

The L^AT_EX 2_ε Sources

Johannes Braams
David Carlisle
Alan Jeffrey
Leslie Lamport
Frank Mittelbach
Chris Rowley
Rainer Schöpf

2025-06-01 Patch level -2

This file is maintained by the L^AT_EX Project team.
Bug reports can be opened (category `latex`) at
<https://latex-project.org/bugs.html>.

Contents

01	ltdirchk.dtx	1
1	L^AT_EX System Dependent Initializations	1
2	Initialization	2
2.1	INITEX	3
2.2	Some bits of 2e	4
3	texsys.cfg	5
3.1	texsys.cfg	5
3.2	UNIX (web2c)	6
3.3	UNIX (other)	7
3.4	MSDOS (emtex)	7
3.5	MSDOS (other)	7
3.6	VMS (DECUS T _E X, PD VMS 3.6)	7
3.7	VMS (???)	7
3.8	MACINTOSH (OzT _E X 1.6)	8
3.9	MACINTOSH (other)	8
3.10	FAKE EXAMPLE	8
4	Setting \@currdir	9
5	Setting \input@path	10

6	Filename Parsing	11
7	TeX Versions	13
8	ltxcheck.tex	13
02	lplain.dtx	14
1	Plain TeX	14
03	ltxvers.dtx	37
1	Version Identification	37
1.1	Declaring an all-new module	40
04	lualatex.dtx	42
1	Overview	42
2	Core TeX functionality	42
3	Plain TeX interface	43
4	Lua functionality	43
4.1	Allocators in Lua	43
4.2	Lua access to TeX register numbers	44
4.3	Module utilities	45
4.4	Callback management	45
5	Implementation	46
5.1	Minimum LuaTeX version	46
5.2	Older L ^A TeX/Plain TeX setup	47
5.2.1	Fixes to <code>etex.src/etex.sty</code>	47
5.2.2	luatex specific settings	48
5.3	Attributes	49
5.4	Category code tables	49
5.5	Named Lua functions	51
5.6	Custom whatsits	51
5.7	Lua bytecode registers	52
5.8	Lua chunk registers	52
5.9	Lua loader	52
5.10	Lua module preliminaries	54
5.11	Lua module utilities	54
5.11.1	Module tracking	54
5.11.2	Module messages	55
5.12	Accessing register numbers from Lua	56
5.13	Attribute allocation	57
5.14	Custom whatsit allocation	58

5.15	Bytecode register allocation	58
5.16	Lua chunk name allocation	58
5.17	Lua function allocation	59
5.18	Lua callback management	59
5.18.1	Housekeeping	59
5.18.2	Handlers	64
5.18.3	Public functions for callback management	67
05	ltxpl.dtx	73
1	expl3-dependent code	73
1.1	Loader	73
1.2	Using expl3 code	76
2	Document-level command names for expl3 functions	77
06	ltxdefns.dtx	80
1	Definitions	80
1.1	Initex initializations	80
1.2	Saved versions of T _E X primitives	80
1.3	Command definitions	81
1.4	Robust commands and protect	90
1.5	Acting on robust commands	96
1.5.1	Copying robust commands	98
1.5.2	Showing robust commands	100
1.5.3	Commands defined with <code>\DeclareRobustCommand</code>	101
1.5.4	Commands defined with <code>\newcommand</code> (with optional argument)	103
1.5.5	Showing environments	105
1.6	Internal defining commands	106
2	Discretionary Hyphenation	110
07	ltxcmd.dtx	113
1	Creating document commands	113
1.1	Variables and constants	113
1.2	Declaring commands and environments	117
1.3	Structure of xparse commands	122
1.4	Normalizing the argument specifications	127
1.5	Preparing the signature: general mechanism	135
1.6	Setting up a standard signature	136
1.7	Setting up expandable types	142
1.7.1	Copying a command and its internal structure	145
1.7.2	Showing the definition of a command	151
1.8	Grabbing arguments	156
1.9	Grabbing arguments expandably	174
1.10	Argument processors	179

1.11	Conversion to key–value form	181
1.12	Utilities	184
1.13	Messages	188
1.14	User functions	193
08	lthooks.dtx	199
1	Introduction	199
2	Package writer interface	199
2.1	L ^A T _Ε X 2 _ε interfaces	199
2.1.1	Declaring hooks	199
2.1.2	Special declarations for generic hooks	200
2.1.3	Using hooks in code	201
2.1.4	Updating code for hooks	202
2.1.5	Hook names and default labels	205
2.1.6	The <code>top-level</code> label	207
2.1.7	Defining relations between hook code	207
2.1.8	Querying hooks	209
2.1.9	Displaying hook code	210
2.1.10	Debugging hook code	211
2.2	L ³ programming layer (<code>expl3</code>) interfaces	211
2.3	On the order of hook code execution	214
2.4	The use of “reversed” hooks	216
2.5	Difference between “normal” and “one-time” hooks	217
2.6	Generic hooks provided by packages	217
2.7	Hooks with arguments	218
2.8	Private L ^A T _Ε X kernel hooks	220
2.9	Legacy L ^A T _Ε X 2 _ε interfaces	220
3	L^AT_ΕX 2_ε commands and environments augmented by hooks	221
3.1	Generic hooks	221
3.1.1	Generic hooks for all environments	222
3.1.2	Generic hooks for commands	223
3.1.3	Generic hooks provided by file loading operations	223
3.2	Hooks provided by <code>\begin{document}</code>	224
3.3	Hooks provided by <code>\end{document}</code>	224
3.4	Hooks provided by <code>\shipout</code> operations	226
3.5	Hooks provided for paragraphs	226
3.6	Hooks provided in NFSS commands	226
3.7	Hook provided by the mark mechanism	227

4	The Implementation	227
4.1	Debugging	227
4.2	Borrowing from internals of other kernel modules	228
4.3	Declarations	228
4.4	Providing new hooks	230
4.4.1	The data structures of a hook	230
4.4.2	On the existence of hooks	231
4.4.3	Setting hooks up	232
4.4.4	Disabling and providing hooks	238
4.5	Parsing a label	240
4.6	Adding or removing hook code	244
4.7	Setting rules for hooks code	265
4.8	Specifying code for next invocation	286
4.9	Using the hook	289
4.10	Querying a hook	294
4.11	Messages	298
4.12	L ^A T _ε X package interface commands	301
4.13	Deprecated that needs cleanup at some point	305
4.14	Internal commands needed elsewhere	306
09	ltxcmdhooks.dtx	309
1	Introduction	309
2	Restrictions and Operational details	310
2.1	Patching	311
2.1.1	Timing	311
2.2	Commands that look ahead	311
3	Package Author Interface	312
3.1	Arguments and redefining commands	313
4	The Implementation	313
4.1	Execution plan	313
4.2	Variables	314
4.3	Variants	315
4.4	Patching or delaying	315
4.5	Patching commands	317
4.5.1	Patching by expansion and redefinition	318
4.5.2	Patching by retokenization	326
4.6	Messages	333
10	ltsockets.dtx	335
1	Introduction	335

2	Configuration of the transformation process	335
2.1	The template mechanism	335
2.2	The hook mechanism	336
2.3	The socket mechanism	336
2.3.1	Examples	338
2.3.2	Details and semantics	339
2.3.3	Command syntax	341
2.3.4	Rationale for error handling	343
3	The Implementation	343
3.1	Debugging the socket structures	343
3.2	The L3 layer commands	344
3.3	Error messages	348
3.4	The L ^A T _E X 2 _ε interface commands	348
11	ltemplates.dtx	350
1	Introduction	350
2	What is a document?	350
3	Types, templates, and instances	351
4	Template types	351
5	Templates	351
6	Multiple choices	355
7	Instances	356
8	Document interface	357
9	Changing existing definitions	358
10	Getting information about templates and instances	358
11	The implementation	359
11.1	Variables and constants	360
11.2	Testing existence and validity	362
11.3	Saving and recovering property lists	363
11.4	Creating new template types	365
11.5	Design part of template declaration	366
11.5.1	Storing values	369
11.6	Implementation part of template declaration	370
11.7	Editing template defaults	375
11.8	Creating instances of templates	377
11.9	Using templates directly	379
11.10	Assigning values to variables	380
11.11	Using instances	383
11.12	Assignment manipulation	384

11.13	Showing templates and instances	384
11.14	Messages	385
11.15	User functions	389
12	lalloc.dtx	391
1	Counters	391
13	lcntrl.dtx	393
1	Program control structure	393
14	lerror.dtx	397
1	Error handling and tracing	397
1.1	General commands	397
1.2	Specific errors	403
1.3	Tracing	407
15	ltpar.dtx	408
1	Paragraphs	408
1.1	Implementation	408
16	ltpara.dtx	410
1	Introduction	410
1.1	The default processing done by the engine	410
2	The new mechanism implemented for \LaTeX	412
2.1	The provided hooks	413
2.2	Altered and newly provided commands	414
2.3	Examples	415
2.3.1	Testing the mechanism	415
2.3.2	Mark the first paragraph of each <code>itemize</code>	417
2.4	Some technical notes	417
2.4.1	Glue items between paragraphs (found with <code>fancy par</code>)	417
3	The Implementation	418
3.1	Providing hooks for paragraphs	418
3.2	The error messages	425
17	ltmeta.dtx	427
1	Introduction	427
1.1	<code>\DocumentMetadata</code>	427

2	The Implementation	427
18	ltspace.dtx	430
1	Spacing	430
1.1	User Commands	430
1.2	Chris' comments	430
1.3	Some immediate actions	432
1.4	The code	433
1.5	Vertical spacing	440
1.6	Horizontal space (and breaks)	446
19	ltlogos.dtx	450
1	Logos	450
20	ltfiles.dtx	451
1	File Handling	451
1.1	Safe Input Macros	464
1.2	Listing files	472
21	ltoutenc.dtx	475
1	Font encodings	475
1.1	Removing encoding-specific commands	477
1.2	The order of declarations	478
1.3	Docstrip modules	478
1.4	Definitions for the kernel	479
1.4.1	Declaration commands	479
1.4.2	Hyphenation	487
1.4.3	Miscellania	488
1.4.4	Default encodings	488
1.4.5	Math material	490
1.5	Definitions for the OT1 encoding	491
1.6	Definitions for the T1 encoding	494
1.7	Definitions for the OMS encoding	499
1.8	Definitions for the OML encoding	500
1.9	Definitions for the OT4 encoding	500
1.10	Definitions for the TS1 encoding	502
1.11	Definitions for the TU encoding	507
2	Package files	518
2.1	The fontenc package	518
22	ltspace.dtx	521

1	Counters and Lengths	521
1.1	Environment Counter Macros	521
23	ltnlength.dtx	531
1	Lengths	531
24	ltnssbas.dtx	533
1	Preliminary macros	533
2	Macros for setting up the tables	534
3	Selecting a new font	543
3.1	Macros for the user	543
3.2	Macros for loading fonts	549
4	Assigning math fonts to <i>versions</i>	556
25	ltnssaxes.dtx	563
1	Changing the font series	563
1.1	The series lookup table	563
1.2	Mapping rules for series changes	564
1.3	Changing to a new series	617
2	Changing the shape	622
2.1	Mapping rules for shape combinations	624
2.2	Changing to a new shape	628
3	Make sure we win . . .	630
26	ltnsstrc.dtx	633
1	Introduction	633
2	A driver for this document	633
3	The Implementation	634
4	Handling Options	634
5	Macros common to <code>fam.tex</code> and <code>tracefnt.sty</code>	636
5.1	General font loading	636
5.2	Math fonts setup	642
5.2.1	Outline of algorithm for math font sizes	642
5.2.2	Code for math font size setting	643
5.2.3	Other code for math	644

6	Scaled font extraction	646
6.1	Sizefunctions	654
27	lffscmp.dtx	658
28	lffsdcl.dtx	663
1	Interface Commands	663
29	lffssini.dtx	695
1	NFSS Initialization	695
1.1	Providing math <i>versions</i>	695
2	Custom series settings for main document families	696
3	Supporting nested emphasis	713
3.1	Legacy	717
3.2	Miscellaneous	717
30	fontdef.dtx	723
1	Introduction	723
2	Customization	723
3	The docstrip modules	724
4	A driver for this document	724
5	The fonttext.ltx file	724
5.1	Encodings	725
5.2	Defaults	727
6	The fontmath.ltx file	729
6.1	The font encodings used	729
6.1.1	Symbolfont and Alphabet declarations	730
6.2	Math font sizes	730
6.3	The math symbol assignments	731
6.3.1	The letters	731
6.3.2	The digits	732
6.3.3	Punctuation, brace, etc. keys	732
6.3.4	Delimitercodes for characters	733
6.4	Symbols accessed via control sequences	733
6.4.1	Greek letters	733
6.4.2	Ordinary symbols	734
6.4.3	Large Operators	735
6.4.4	Binary symbols	735

6.4.5	Relations	736
6.4.6	Arrows	738
6.4.7	Punctuation symbols	738
6.4.8	Math accents	739
6.4.9	Radicals	739
6.4.10	Over and under something, etc	739
6.4.11	Delimiters	740
6.5	Math versions of text commands	741
6.6	Other special functions and parameters	742
6.6.1	Biggggg	742
6.6.2	The log-like functions	742
6.6.3	Parameters	742
7	Default cfg files	742
31	preload.dtx	744
1	Overview	744
2	Customization	744
3	Module switches for the DOCSTRIP program	744
4	A driver for this document	745
5	The code	745
32	lftntcmd.dtx	747
1	Introduction	747
2	The implementation	749
3	Initialization	755
33	ltxtextcomp.dtx	756
1	Sub-encodings	758
1.1	Unavailable in sub-encoding 1 and higher (drop symbols not working in Latin Modern)	760
1.2	Unavailable in sub-encoding 2 (majority of new OTF fonts via autoinst) and higher	760
1.3	Unavailable in sub-encoding 3 and higher	763
1.4	Unavailable in sub-encoding 4 and higher	763
1.5	Unavailable in sub-encoding 5 (most older PS fonts) and higher	764
1.6	Unavailable in sub-encoding 6 and higher	764
1.7	Unavailable in sub-encoding 7 and higher	764
1.8	Unavailable in sub-encoding 8 and higher	764
1.9	Unavailable in Sub-encoding 9 (most missing)	765

2	Unicode engine specials	765
3	Font family sub-encodings setup	766
4	Legacy symbol support for lists and footnote symbols	770
5	The <code>textcomp</code> package	775
5.1	The old <code>textcomp</code> package code	776
5.1.1	Supporting oldstyle digits	785
5.1.2	Subset encoding defaults	785
6	The <code>checkencodingssubset.tex</code> file	787
34	<code>ltpageno.dtx</code>	796
1	Page Numbering	796
35	<code>ltxref.dtx</code>	797
1	Cross Referencing	797
1.1	Cross Referencing	797
36	<code>ltproperties.dtx</code>	805
1	Introduction	805
2	Design discussion	805
3	Handling unknown labels and properties	806
4	Rerun messages	806
5	Open points	806
6	Code interfaces	806
7	Auxiliary file interfaces	808
8	\LaTeX 2 ϵ interface	808
9	Pre-declared properties	809
10	The Implementation	810
10.1	Reference commands	812
10.2	Tests and warnings	814
10.3	Predeclared properties	816
10.4	Messages	818
37	<code>ltmiscen.dtx</code>	819

1	Miscellaneous Environments	819
1.1	Environments	819
1.2	Center, Flushright, Flushleft	833
1.3	Verbatim	836
38	l_tmath.dtx	844
1	Math setup	844
1.1	Math commands based on plain T _E X	844
1.1.1	The log-like functions	844
1.1.2	Bigggg	845
1.1.3	The UNSORTED Rest	845
1.2	Math Environments	852
1.3	External options to the standard document classes	857
1.3.1	Left equation numbering	857
1.3.2	Flush left equations	857
39	l_tlists.dtx	860
1	List, and related environments	860
1.1	List and Trivlist	861
1.2	Vertical Spacing (skips)	862
1.3	Penalties	862
1.4	Horizontal Spacing (dimens)	862
1.5	Default Values	862
1.6	Itemize and Enumerate	875
40	l_tboxes.dtx	878
1	L^AT_EX Box commands	878
1.1	Some low-level constructs	896
41	l_ttab.dtx	897
1	Tabbing, Tabular and Array Environments	897
1.1	tabbing	897
1.2	array and tabular environments	906
42	l_tpictur.dtx	922
1	Picture Mode	922
1.1	Curves	949
43	l_tthm.dtx	954

1	Theorem Environments	954
44	ltsect.dtx	958
1	Sectioning Commands	958
1.1	The Title	958
1.2	Sectioning	959
1.2.1	Initializations	966
1.3	Table of Contents etc.	966
1.3.1	Convention	966
1.3.2	Commands	966
45	lfloat.dtx	972
1	Floats	972
1.1	Floating Environments	972
1.2	Footnotes	986
46	ltxglo.dtx	995
1	Index and Glossary Generation	995
47	ltxbibl.dtx	998
1	Bibliography Generation	998
1.1	Default definitions	1002
48	ltxmarks.dtx	1003
1	Introduction	1003
2	Design-level and code-level interfaces	1004
2.1	Use cases for conditionals	1006
2.2	Understanding regions	1006
2.3	Debugging mark code	1008
3	Application examples	1008
4	Legacy L^AT_EX 2_ε interface	1008
4.1	Legacy design-level and document-level interfaces	1009
4.2	Legacy interface extensions	1009
5	Notes on the mechanism	1010
6	Public interfaces for packages such as multicol	1011
7	Internal functions for the standard output routine of L^AT_EX	1012

8	The Implementation	1013
8.1	Allocating new mark classes	1013
8.2	Updating mark structures	1015
8.3	Placing and retrieving marks	1023
8.4	Comparing mark values	1025
8.5	Messages	1025
8.6	Debugging the mark structures	1026
8.7	Designer-level interfaces	1028
9	L^AT_EX 2_ε integration	1029
9.1	Core L ^A T _E X 2 _ε integration	1029
9.2	Other L ^A T _E X 2 _ε output routines	1032
9.3	Rollback information	1032
49	ltpage.dtx	1033
1	Page styles and related commands	1033
1.1	Page Style Commands	1033
1.2	How a page style makes running heads and feet	1033
1.3	marking conventions	1033
50	ltclass.dtx	1038
1	Introduction	1038
2	User interface	1038
2.1	Option processing	1039
3	Class and Package interface	1040
3.1	Class name and version	1040
3.2	Package name and version	1040
3.3	Requiring other packages	1040
3.4	Declaring new options	1041
3.5	Safe Input Macros	1042
4	Implementation	1042
4.1	Hooks	1071
4.2	Providing shipment	1074
5	Package/class rollback mechanism	1082
6	After Preamble	1090
51	ltkeys.dtx	1091

1	Creating and using keyval options	1091
1.1	Implementation of <code>ltkeys</code>	1092
1.2	Key properties	1092
1.3	Main mechanism	1093
1.4	The document interfaces	1097
1.5	Option usage scope	1098
1.6	General key setting	1099
52	ltfilehook.dtx	1100
1	Introduction	1100
1.1	Provided hooks	1100
1.2	General hooks for file reading	1100
1.3	Hooks for package and class files	1101
1.4	Hooks for <code>\include</code> files	1102
1.5	High-level interfaces for \LaTeX	1103
1.6	Kernel, class, and package interfaces for \LaTeX	1104
1.7	A sample package for structuring the log output	1104
2	The Implementation	1105
2.1	Document and package-level commands	1105
2.2	<code>expl3</code> helpers	1106
2.3	Declaring the file-related hooks	1109
2.4	Patching \LaTeX 's <code>\InputIfFileExists</code> command	1109
2.5	Declaring a file substitution	1111
2.6	Selecting a file (<code>\set@curr@file</code>)	1113
2.7	Replacing a file and detecting loops	1116
2.7.1	The Tortoise and Hare algorithm	1117
2.8	Preventing a package from loading	1119
2.9	High-level interfaces for \LaTeX	1120
2.10	Internal commands needed elsewhere	1120
3	A sample package for structuring the log output	1121
4	Package emulations	1122
4.1	Package <code>atveryend</code> emulation	1122
53	ltshipout.dtx	1124
1	Introduction	1124
1.1	Overloading the <code>\shipout</code> primitive	1124
1.2	Provided hooks	1125
1.3	Legacy \LaTeX commands	1127
1.4	Special commands for use inside the hooks	1128
1.5	Provided $\text{Lua}\TeX$ callbacks	1128
1.6	Information counters	1129
1.7	Debugging shipout code	1129

2	Emulating commands from other packages	1130
2.1	Emulating atbegshi	1130
2.2	Emulating everyshi	1131
2.3	Emulating atenddvi	1131
2.4	Emulating everypage	1131
3	The Implementation	1132
3.1	Debugging	1132
3.2	Handling the end of job hook	1144
4	Legacy L^AT_EX 2_ε interfaces	1147
5	Internal commands needed elsewhere	1148
6	Package emulation for compatibility	1149
6.1	Package atenddvi emulation	1149
6.2	Package atbegshi emulation	1150
6.3	Package everyshi emulation	1151
54	ltoutput.dtx	1152
1	Output Routine and float handling	1152
1.1	Historical notes on the algorithm and commands	1152
1.2	Core definitions	1162
1.2.1	Definition of float boxes	1162
1.2.2	Page layout parameters	1163
1.2.3	Internal registers	1165
1.2.4	Page break commands	1165
2	The L^AT_EX output routine	1169
2.1	Hooks and replaceable code blocks	1169
2.1.1	Output routine hooks	1169
2.1.2	Replaceable code blocks (sockets)	1170
2.1.3	Tagging sockets	1171
2.1.4	Output routine commands	1171
2.2	The output routine configuration components	1182
2.2.1	Configuration sockets	1184
2.2.2	Dealing with floats	1194
2.2.3	Kludgeins	1223
2.2.4	Float control	1225
2.2.5	Float placement parameters	1239
55	ltagging.dtx	1243
1	General support for tagged output	1243

2	Implementation	1244
2.1	Math collection	1245
2.2	Tagging sockets	1245
2.2.1	Tagging support for paragraph setup	1245
2.2.2	Tagging socket for targets	1246
2.2.3	Tagging sockets for toc	1246
2.2.4	Tagging support for marginpar	1246
2.2.5	Tagging support for table/tabular packages	1247
2.2.6	Tagging Support for floats	1248
2.3	Tagging support for output routines	1248
2.4	Tagging support for math	1249
2.4.1	General sockets	1249
2.4.2	Sockets specific for luamml	1249
3	For lttab.dtx parked here for now	1251
3.1	Variables for row, column and span counting	1251
3.2	Tracing/debugging	1252
3.3	Interface commands	1253
56	lthyphen.dtx	1258
57	ltfinal.dtx	1260
1	Final settings	1260
1.1	Debugging	1260
1.2	Typesetting parameters	1260
1.3	Lccodes for hyphenation	1264
1.4	Hyphenation	1266
1.5	Font loading	1267
1.6	Input encoding	1268
1.7	Lccodes and uccodes	1273
1.8	Case changing	1274
1.9	Applying Patch files	1276
1.10	Freeing Memory	1277
1.11	Initialise file list	1278
1.12	Preparation for supporting PDF in backends	1278
1.13	Do some temporary work for pre-release	1278
1.14	Some last minute initializations	1278
1.15	Dumping the format	1279
	Change History	1280
	Index	1361

File 01

ltdirchk.dtx

1 L^AT_EX System Dependent Initializations

This file implements the semi-automatic determination of various system dependent parts of the initialization. The actual definitions may be placed in a file `texsys.cfg`. Thus for operating systems for which the tests here do not result in acceptable settings, a ‘hand written’ `texsys.cfg` may be produced.

The macros that must be defined are:

`\@currdir` `\@currdir{filename}<space>` should expand to a form of the filename that uniquely refers to the ‘current directory’ if this is possible. (The expansion should also end with a space.) on UNIX, this is `\def\@currdir{./}`. For more exotic operating systems you may want to make `\@currdir` a macro with arguments delimited by `.` and/or `<space>`. If the operating system has no concept of directory structure, this macro should be defined to be empty.

`\input@path` If the primitive `\openin` searches the same directories as the primitive `\input`, then it is possible to tell (using `\ifeof`) whether a file exists before trying to input it. For systems like this, `\input@path` should be left undefined.

If `\openin` does not ‘follow’ `\input` then `\input@path` must be defined to be a list of directories to search for input files. The format for each directory is as for `\@currdir`, normally just a prefix is required, but it may be a macro with space-delimited argument. That is, if `<dir>` is an entry in the input path, T_EX will try to load the expansion of `<dir><filename><space>`

So either `<dir>` should be defined as a macro with argument delimited by space, or it should just expand to a directory name, including the final directory separator, so that it may be concatenated with the `<filename>`. This means that for UNIX-like syntax, each `<dir>` should end with a slash, `/`.

`\input@path` should expand to a list of such directories, each in a `{}` group.

`\filename@parse` After a call of the form: `\filename@parse{<filename>}`, the three macros `\filename@area`, `\filename@base` and `\filename@ext` should be defined to be the ‘area’ (or directory), basename and extension respectively. If there was no extension specified in `<filename>`, `\filename@ext` should be `\let` to `\relax` (so this case may be tested with `\@ifundefined{filename@ext}` and, perhaps a default extension substituted).

Normally one would not need to define this macro in `texsys.cfg` as the automatic tests can supply parsers that work with UNIX and VMS and Macintosh syntax, as well as a basic parser that will cover many other cases. However some operating systems may need a ‘hand produced’ parser in which case it should be defined in this file.

The UNIX parser also works for most MSDOS T_EX versions. Currently if the UNIX, VMS or Macintosh parser is not used, `\filename@parse` is defined to always return an empty area, and to split the argument into basename and extension at the first ‘.’ that occurs in the name. Parsers for other formats may be defined in `texsys.cfg`, in which case they will be used in preference to the default definitions.

`\@TeXversion` `\@TeXversion` is now set automatically by the initialization tests in this file. You should not need to set it in `texsys.cfg`, however the following documentation is left for information. L^AT_EX does not set this variable exactly, the automatic tests set it to:

2 for any version, $v, v < 3.0$

3 for any version, $v, 3.0 \leq v \leq 3.14$

`<undefined>` otherwise.

However these values are accurate enough for L^AT_EX to take appropriate action for these old T_EXs.

If your T_EX is older than version 3.141, then you should define `\@TeXversion` (using `\def`) to be the version number. If you do not do this¹, L^AT_EX will not work around a bug in old T_EX versions, and so error messages will appear in a very strange format, with `^^J` appearing instead of line breaks:

```
LaTeX Error: \rubbish undefined.^^J^^JSee the LaTeX manual or LaTeX=
Companion
for explanation.^^JType H <return> for immediate help.
...

.3 \renewcommand{\rubbish}
      {}
```

However if you put `\def\@TeXversion{3.14}` in `texsys.cfg` the following format will be used:

```
LaTeX Error: \rubbish undefined.

ee the LaTeX manual or LaTeX Companion for explanation.
ype H <return> for immediate help.
.
...

.3 \renewcommand{\rubbish}
      {}
```

Note that this has an extra line `!` which does not appear in error messages that use the default settings with a current version of T_EX, but this should not cause any confusion we hope.

2 Initialization

As this file is read at a very early stage, some definitions that are normally considered to be part of the format must be made here.

Most such definitions are repeated later in the “right” place, usually (but not always) with different implementations. To be able to spot this more easily if you look into the file `latex.ltx` (which is stripped of comments) we add some comment lines to that effect that survive the stripping process by `docstrip`.

```
1 <*dircheck>
2 %% ---- START temporary definitions for bootstrapping; later overwritten ----
3 </dircheck>
```

¹Actually if your T_EX is really old, version 2, L^AT_EX can detect this, and sets `\@TeXversion` to 2 if it is not set in the `cfg` file.

2.1 INITEX

```
4 <*dircheck>
5 <*initex>
6 <initex> \ifnum\catcode'\{=1
7 <initex> \errmessage
8 <initex> {LaTeX must be made using an initex with no format preloaded}
9 <initex> \fi
10 \catcode'\{=1
11 \catcode'\}=2
```

If LuaTeX is in use the extensions and other new primitives have to be activated: this is done as early as possible. Older versions of LuaTeX do not hide the primitives: a version check is not needed as the version itself will be missing in the case where action is needed!

```
12 \ifx\directlua\undefined
13 \else
14 \ifx\luatexversion\undefined
```

Enable e-TeX/pdfTeX/Umath primitives with their natural names

```
15 \directlua{tex.enableprimitives("",%
16 tex.extraprimitives('etex', 'pdftex', 'umath'))}
```

In current formats enable primitives with unprefix names. the latexrelease guards allow the primitives to be defined with a \luatex prefix if older formats are specified.

The unprefix forms are *not* undefined for improved compatibility with external packages when rolling back the format.

```
17 </initex>
18 </dircheck>
19 <*initex, latexrelease>
20 <latexrelease> \ifx\directlua\undefined\else
21 <latexrelease> \IncludeInRelease{2015/10/01}{\luatexluafunction}
22 <latexrelease> {LuaTeX (prefixed names)}%
23 \directlua{tex.enableprimitives("",%
24 tex.extraprimitives("omega", "aleph", "luatex"))}
25 <latexrelease> \EndIncludeInRelease
26 <latexrelease> \IncludeInRelease{0000/00/00}{\luatexluafunction}
27 <latexrelease> {LuaTeX (prefixed names)}%
28 <latexrelease> \directlua{
29 <latexrelease> tex.enableprimitives(
30 <latexrelease> "luatex",
31 <latexrelease> tex.extraprimitives("core", "omega", "aleph", "luatex")
32 <latexrelease> )
33 <latexrelease> }
34 <latexrelease> \EndIncludeInRelease
35 <latexrelease> \fi
36 </initex, latexrelease>
37 <*dircheck>
38 <*initex>
39 \fi
40 \fi
```

A test can now be made for eTeX.

```
41 <initex> \ifx\eTeXversion\undefined
42 <initex> \errmessage
43 <initex> {LaTeX requires e-TeX}
44 <initex> \expandafter\endinput
```

```

45 <initex>\fi
    That distraction over, back to the basics of a format.
46 \catcode'\#=6
47 \catcode'\^=7
48 \chardef\active=13
49 \catcode'\@=11
50 \countdef\count@=255
51 \let\bgroup={ \let\egroup=}
52 \ifx\@input\undefined\let\@input\input\fi
53 \ifx\@end\undefined\let\@end\end\fi
54 \chardef\@inputcheck0
55 \chardef\sixt@n=16
56 \newlinechar'\^^J
57 \def\typeout{\immediate\write17}
58 \def\dospecials{\do\ \do\|\do\{\do\}\do\$\do\&%
59 \do\#\do\^\do\_ \do\% \do\~ \do\^^I}
60 \def\makeother#1{\catcode'#1=12\relax}
61 \def\space{ }
62 \def\tempswafalse{\let\if@tempswa\iffalse}
63 \def\tempswatrue{\let\if@tempswa\iftrue}
64 \let\if@tempswa\iffalse
65 \def\loop#1\repeat{\def\iterate{#1\relax\expandafter\iterate\fi}%
66 \iterate \let\iterate\relax}
67 \let\repeat\fi
68 </initex>

```

2.2 Some bits of 2e

```

69 <*2kernel>
70 \def\two@digits#1{\ifnum#1<10 0\fi\number#1}
71 \long\def\@firstoftwo#1#2{#1}
72 \long\def\@secondoftwo#1#2{#2}

```

This is a special version of \ProvidesFile for initex use.

```

73 \def\ProvidesFile#1{%
74 \begingroup
75 \catcode'\ 10 %
76 \ifnum \endlinechar<256 %
77 \ifnum \endlinechar>\m@ne
78 \catcode\endlinechar 10 %
79 \fi
80 \fi
81 \@makeother\/%
82 \@ifnextchar[{\@providesfile{#1}}{\@providesfile{#1}[]}]
83 \def\@providesfile#1[#2]{%
84 \wlog{File: #1 #2}%
85 \@addtofilelist{ #2}%
86 \endgroup}
87 \long\def\@addtofilelist#1{}
88 \def\@empty{}
89 \catcode'\%=12
90 \def\@percentchar{%}
91 \catcode'\%=14
92 \let\@currdir\undefined
93 \let\input@path\undefined

```

```

94 \let\filename@parse \@undefined
\strip@prefix

```

```

95 \def\strip@prefix#1>{}
96 </2ekernel>

```

(End of definition for \strip@prefix.)

3 texsys.cfg

As mentioned above, any site specific definitions required to describe the filename handling must be entered into a file `texsys.cfg`. If `texsys.cfg` can not be located by `\openin`, we write a default version out. The default version only contains comments, so we do not actually input the file in that case. The automatic tests later will, hopefully, correctly define the required macros.

The tricky code below checks to see if `texsys.cfg` exists. If it does not, all the text in this file between `START` and `END` is copied verbatim to a new file `texsys.cfg`. If `texsys.cfg` is found, then it is simply input. This is only done when this file is being used unstripped.

```

97 <docstrip>
98 \openin15=texsys.cfg
99 \ifeof15
100 \typeout{** Writing a default texsys.cfg}
101 \immediate\openout15=texsys.cfg
102 \begingroup
103 \catcode'\^M\active%
104 \let^M\par%
105 \def\reserved@a#1^M{%
106   \def\reserved@b{#1}%
107   \ifx\reserved@b\reserved@c\endgroup\else%
108     \immediate\write15{#1}%
109     \expandafter\reserved@a{fi}%
110 \def\reserved@d#1START^M{\let\do\makeother\dospecials\reserved@a}%
111 \catcode'\%=12
112 \def\reserved@c{END}
113 \reserved@d

```

START

3.1 texsys.cfg

This file contains the site specific definitions of the four macros `\@currdir`, `\input@path`, `\filename@parse` and `\TeXversion`.

As distributed it only contains comments, however this ‘empty’ file will work on many systems because of the automatic tests built into `ltdirchk.dtx`. You *are* allowed to edit this file to add definitions of these macros appropriate to your system.

The macros that must be defined are:

```

\@currdir   \@currdir{filename}<space> should expand to a form of the filename that uniquely
refers to the ‘current directory’ if this is possible. (The expansion should also end with
a space.) on UNIX, this is \def\@currdir{./}. For more exotic operating systems you
may want to make \@currdir a macro with arguments delimited by . and/or <space>. If

```

the operating system has no concept of directory structure, this macro should be defined to be empty.

`\input@path` If the primitive `\openin` searches the same directories as the primitive `\input`, then it is possible to tell (using `\ifeof`) whether a file exists before trying to input it. For systems like this, `\input@path` should be left undefined.

If `\openin` does not ‘follow’ `\input` then `\input@path` must be defined to be a list of directories to search for input files. The format for each directory is as for `\@currdir`, normally just a prefix is required, but it may be a macro with space-delimited argument. That is, if `<dir>` is an entry in the input path, \TeX will try to load the expansion of

`<dir><filename><space>`

So either `<dir>` should be defined as a macro with argument delimited by space, or it should just expand to a directory name, including the final directory separator, so that it may be concatenated with the `<filename>`. This means that for UNIX-like syntax, each `<dir>` should end with a slash, `/`. One exception to this rule is that the input path should *always* contain the empty directory `{}` as this will allow ‘full pathnames’ to be used, and the ‘current directory’ to be searched.

`\input@path` should expand to a list of such directories, each in a `{}` group.

`\filename@parse` After a call of the form: `\filename@parse{<filename>}`, the three macros `\filename@area`, `\filename@base`, `\filename@ext` should be defined to be the ‘area’ (or directory), basename and extension respectively. If there was no extension specified in `<filename>`, `\filename@ext` should be `\let` to `\relax` (so this case may be tested with `\@ifundefined{filename@ext}` and, perhaps a default extension substituted).

Normally one would not need to define this macro in `texsys.cfg` as the automatic tests can supply parsers that work with UNIX and VMS syntax, as well as a basic parser that will cover many other cases. However some operating systems may need a ‘hand produced’ parser in which case it should be defined in this file.

The UNIX parser also works for most MSDOS \TeX versions. Currently if the UNIX or VMS parser is not used, `\filename@parse` is defined to always return an empty area, and to split the argument into basename and extension at the first ‘.’ that occurs in the name. Parsers for other formats may be defined in `texsys.cfg`, in which case they will be used in preference to the default definitions.

`\@TeXversion` You should not need to set this macro in `texsys.cfg`. \LaTeX tests to set this automatically. See the comments in the opening section of `ltdirchk.dtx`.

The following sections give examples of definitions which might work on various systems. These are currently mainly untested as I only have access to a few systems, all of which do not need this file as the automatic tests work. All the code is commented out.

3.2 UNIX (web2c)

This implementation does make `\openin` and `\input` look in the same places. Acceptable settings are made by `ltdirchk.dtx`, and so this file may be empty. The definitions below are therefore just for information.

```
114 %\def\@currdir{./}
```

```
115 %\let\input@path\@undefined
```


3.3 UNIX (other)

Apparently some commercial UNIX implementations have different paths for `\openin` and `\input`. For these one could use definitions like the following (with whatever directories are used at your site): note that the directory names should end with `/`.

```
116 % \def\currdir{./}
117 % \def\input@path{%
118 %   {/usr/local/lib/tex/inputs/distrib/}%
119 %   {/usr/local/lib/tex/inputs/contrib/}%
120 %   {/usr/local/lib/tex/inputs/local/}%
121 % }
```

3.4 MSDOS (emtex)

This implementation does make `\openin` and `\input` look in the same places. Acceptable settings are made by `ltdirchk.dtx`, and so this file may be empty. The definitions below are therefore just for information.

```
122 % \def\currdir{./}
123 % \let\input@path\undefined
```

3.5 MSDOS (other)

Some PC implementations have different paths for `\openin` and `\input`. For these one could use definitions like the following (with whatever directories are used at your site): note that the directory names should end with `/`. This assumes the implementation uses UNIX style `/` as the directory separator.

```
124 % \def\currdir{./}
125 % \def\input@path{%
126 %   {c:/tex/inputs/distrib/}%
127 %   {c:/tex/inputs/contrib/}%
128 %   {c:/tex/inputs/local/}%
129 % }
```

3.6 VMS (DECUS T_EX, PD VMS 3.6)

This implementation does make `\openin` and `\input` look in the same places. Acceptable settings are made by `ltdirchk.dtx`, and so this file may be empty. The definitions below are therefore just for information.

```
130 % \def\currdir{[]}
131 % \let\input@path\undefined
```

3.7 VMS (???)

Some VMS implementations have different paths for `\openin` and `\input`. For these one could use definitions like the following:

```
132 % \def\currdir{[]}
133 % \def\input@path{%
134 %   {tex_inputs:}%
135 %   {SOMEDISK:[SOME.TEX.DIRECTORY]}%
136 % }
```

3.8 MACINTOSH (OzTeX 1.6)

This implementation does make `\openin` and `\input` look in the same places. Acceptable settings are made by `ltdirchk.dtx`, and so this file may be empty. The definitions below are therefore just for information.

```
137 % \def\currdir{:}
138 % \let\input@path\undefined
```

3.9 MACINTOSH (other)

Some Macintosh implementations have different paths for `\openin` and `\input`. For these one could use definitions like the following (with whatever folders are used on your machine): note that the directory names should end with `:`, and they should contain *no* spaces.

```
139 % \def\currdir{:}
140 % \def\input@path{%
141 %   {Hard-Disk:Applications:TeX:TeX-inputs:}%
142 %   {Hard-Disk:Applications:TeX:My-inputs:}%
143 % }
```

3.10 FAKE EXAMPLE

This example is for an operating system that has filenames of the form `<area>name`. For maximum compatibility with macro sets, you want `name.ext` to be mapped to `<ext>name`, and `<area>name.ext` to be mapped to `<area.ext>name`. `\input` does this mapping automatically, but `\openin` does not, and does not look in the same places as `\input`. `<>name` is the desired ‘current directory’ syntax.

the following code would possibly work:

```
144 % \def\@dir#1#2 {%
145 %   \@d@r{#1}#2..\@nil}
146 % \def\@d@r#1#2.#3.#4\@nil{%
147 %   <\ifx\@dir#1\@dir\else#1\ifx\@dir#3\@dir\else.\fi\fi#3>#2 }
148 %
149 % \def\currdir{\@dir{}}
150 % \def\input@path{%
151 %   {\@dir{area.one}}%
152 %   {\@dir{area.two}}%
153 % }
```

END

```
154 \immediate\closeout15
```

If `texsys.cfg` did exist, then input it.

```
155 \else
156 \typeout{** Using the existing texsys.cfg}
157 \closein15
158 \input texsys.cfg
159 \fi
160 </docstrip>
```

If the stripped version of this file is being used (in `latex2e.ltx`) then `texsys.cfg` should be there, so just input it.

```
161 <dircheck>\input texsys.cfg
```

4 Setting \@currdir

`\@currdir` This is a local definition of `\IfFileExists`. It tries to relocate `texsys.aux`. If it succeeds, then the `\@currdir` syntax has been determined. If all the tests fail then `\@currdir` will be set to `\@empty`, and `ltxcheck` will warn of this when it checks the format.

```
162 \begingroup
163 \count@ \time
164 \divide \count@ 60
165 \count2 = - \count@
166 \multiply \count2 60
167 \advance \count2 \time
```

The current date and time stamp.

```
168 \edef \today {%
\today 169   \the \year / \two@digits { \the \month } / \two@digits { \the \day } : %
170   \two@digits { \the \count@ } : \two@digits { \the \count2 } }
```

Create a file `texsys.aux` (hopefully in the current directory), then try to locate it again.

```
171 \immediate \openout 15 = texsys.aux
172 \immediate \write 15 { \today ^^J }
173 \immediate \closeout 15 %
```

`#1` is the file to try, `#2` is what to do on success, `#3` on failure. Note that this definition is overwritten later on again!

```
174 \def \IfFileExists #1 #2 #3 { %
175   \openin \@inputcheck #1 %
176   \ifeof \@inputcheck
177     #3 \relax
178   \else
179     \read \@inputcheck to \reserved@a
180     \ifx \reserved@a \today
181       \typeout { #1 found } #2 \relax
182     \else
183       \typeout { BAD: old file \reserved@a (should be \today) } %
184       #3 \relax
185     \fi
186   \fi
187   \closein \@inputcheck }
188 \endlinechar = -1
```

If `\@currdir` has not been pre-defined in `texsys.cfg` then test for UNIX, VMS and Oz-TeX-Mac. syntax.

```
189 \ifx \@currdir \@undefined
190   \IfFileExists { ./texsys.aux } { \gdef \@currdir { ./ } } %
191   { \IfFileExists { [] texsys.aux } { \gdef \@currdir { [] } } %
192   { \IfFileExists { : texsys.aux } { \gdef \@currdir { : } } { } }
```

If it is still undefined at this point, all the above tests failed. Earlier versions interactively prompted for a definition at this point, but it seems impossible to reliably obtain information from users at this point in the installation. This version of the file produces

a format with no user-interaction. Later if the format is not suitable for the system, `texsys.cfg` may be edited and the format re-made.

```

193 \ifx\@currdir\undefined
194   \global\let\@currdir\empty
195   \typeout{^^J^^J%
196     !! No syntax for the current directory could be found^^J%
197   }%
198 \fi

```

Otherwise `\@currdir` was defined in `texsys.cfg`. In this case check that the syntax specified works on this system. (In case a complete L^AT_EX system has been copied from one system to another.) If the test fails, give up. The installer should remove or correct the offending `texsys.cfg` and try again.

```

199 \else
200   \IfFileExists{\@currdir texsys.aux}{}{%
201     \edef\reserved@a{\errhelp{%
202       texsys.cfg specifies the current directory syntax to be^^J%
203       \meaning\@currdir^^J%
204       but this does not work on this system.^^J%
205       Remove texsys.cfg and restart.}}\reserved@a
206     \errmessage{Bad texsys.cfg file: \noexpand\@currdir}\@@end}

```

The version of `\@currdir` in `texsys.cfg` looks OK.

```

207 \fi

208 \immediate\closeout15 %
209 \endgroup

210 \typeout{^^J^^J%
211   \noexpand\@currdir set to:
212   \expandafter\strip@prefix\meaning\@currdir.^^J%
213 }

```

(End of definition for \@currdir, \IfFileExists, and \today.)

Stop here if the file is being used unstripped.

```

214 <*\docstrip>
215 \relax\endinput
216 </\docstrip>

```

5 Setting `\input@path`

Earlier versions of this file attempted to automatically test whether `\input@path` was required, and interactively prompt for a path if necessary. This was not found to be very reliable. The first-time installer of L^AT_EX 2_ε can not be expected to have enough information to supply the correct information to the prompts. Now the interaction is omitted. After the format is made the installer can attempt to run the test document `ltxcheck.tex` through L^AT_EX 2_ε. This will check, among other things, whether `texsys.cfg` will need to be edited and the format remade.

`\input@path` Now set up the `\input@path`.

`\input@path` should either be undefined, or a list of directories as described in the introduction.

```

217 \typeout{^^J%

```

```

218     Assuming \noexpand\openin and \noexpand\input^^J%
219     \ifx\input@path\@undefined
\input@path has not been pre-defined.
220     have the same search path.^^J%
221     \else
\input@path has been defined in texsys.cfg.
222     have different search paths.^^J%
223     LaTeX will use the path specified by \noexpand\input@path:^^J%
224     \fi
225     }

```

(End of definition for \input@path.)

6 Filename Parsing

`\filename@parse` Split a filename into its components.

```

226 \ifx\filename@parse\@undefined
227   \def\reserved@a{.}\ifx\currdir\reserved@a
\filename@parse was not specified in texsys.cfg, but \@currdir looks like UNIX...
228   \typeout{^^JDefining UNIX/DOS style filename parser.^^J}
229   \def\filename@parse#1{%
230     \let\filename@area\@empty
231     \expandafter\filename@path#1/\}
Search for the last /.
232   \def\filename@path#1/#2\{%
233     \ifx\#2\%
234       \def\reserved@a{\filename@simple#1.\}%
235     \else
236       \edef\filename@area{\filename@area#1/}%
237       \def\reserved@a{\filename@path#2\}%
238     \fi
239     \reserved@a}
240   \else\def\reserved@a{[]}\ifx\currdir\reserved@a
\filename@parse was not specified in texsys.cfg, but \@currdir looks like VMS...
241   \typeout{^^JDefining VMS style filename parser.^^J}
242   \def\filename@parse#1{%
243     \let\filename@area\@empty
244     \expandafter\filename@path#1]\}
Search for the last ].
245   \def\filename@path#1]#2\{%
246     \ifx\#2\%
247       \def\reserved@a{\filename@simple#1.\}%
248     \else
249       \edef\filename@area{\filename@area#1]}\}
250     \def\reserved@a{\filename@path#2\}%
251     \fi
252     \reserved@a}
253   \else\def\reserved@a{:}\ifx\currdir\reserved@a

```

\filename@parse was not specified in texsys.cfg, but \@currdir looks like Macintosh...

```
254 \typeout{^^JDefining Mac style filename parser.^^J}
255 \def\filename@parse#1{%
256 \let\filename@area\@empty
257 \expandafter\filename@path#1:\}}
```

Search for the last :.

```
258 \def\filename@path#1:#2\{%
259 \ifx\#2\%
260 \def\reserved@a{\filename@simple#1.\}%
261 \else
262 \edef\filename@area{\filename@area#1:%}
263 \def\reserved@a{\filename@path#2\}%
264 \fi
265 \reserved@a}
266 \else
```

\filename@parse was not specified in texsys.cfg. So just make a simple parser that always sets \filename@area to empty.

```
267 \typeout{^^JDefining generic filename parser.^^J}
268 \def\filename@parse#1{%
269 \let\filename@area\@empty
270 \expandafter\filename@simple#1.\}
271 \fi\fi\fi
```

\filename@simple is used by all three versions. Finally we can split off the extension.

```
272 </dircheck>
273 <*dircheck, latexrelease>
274 <latexrelease>\IncludeInRelease{2019/10/01}{\filename@simple}
275 <latexrelease> \Final dot for extension}%
276 \def\filename@simple#1.#2\{%
277 \ifx\#2\%
278 \let\filename@ext\relax
279 \edef\filename@base{#1}%
280 \else
281 \filename@dots{#1}#2\%
282 \fi}
283 \def\filename@dots#1#2.#3\{%
284 \ifx\#3\%
285 \def\filename@ext{#2}%
286 \edef\filename@base{#1}%
287 \else
288 \filename@dots{#1.#2}#3\%
289 \fi}
290 <latexrelease>\EndIncludeInRelease
291 <latexrelease>\IncludeInRelease{0000/00/00}{\filename@simple}
292 <latexrelease> \Final dot for extension}%
293 <latexrelease> \def\filename@simple#1.#2\{%
294 <latexrelease> \ifx\#2\%
295 <latexrelease> \let\filename@ext\relax
296 <latexrelease> \else
297 <latexrelease> \edef\filename@ext{\filename@dot#2\}\}}
```

```

298 <latexrelease> \fi
299 <latexrelease> \edef\filename@base{#1}}
300 <latexrelease>\EndIncludeInRelease
301 </dircheck, latexrelease>
302 <*dircheck>

Remove a final dot, added earlier.
303 \def\filename@dot#1.\{#1}
304 \else
Otherwise, \filename@parse was specified in texsys.cfg.
305 \typeout{^^J^^J%
306 \noexpand\filename@parse was defined in texsys.cfg:^^J%
307 \expandafter\strip@prefix\meaning\filename@parse.^^J%
308 }
309 \fi

(End of definition for \filename@parse.)

```

7 T_EX Versions

`\TeXversion` T_EX versions older than 3.141 require `\TeXversion` to be set. This can be determined automatically due to a trick suggested by Bernd Raichle. Actually this will not always get the correct version number, e.g., T_EX3.14 would be detected as T_EX3, but L^AT_EX only needs to take account of T_EX's older than 3, or between 3 and 3.14.

```

310 \ifx\TeXversion\undefined
311 \ifx\undefined\inputlineno
312 \def\TeXversion{2}
313 \else
314 {\catcode'\^^J=\active
315 \def\reserved@a#1#2\@{\if#1\string^3\fi}
316 \edef\reserved@a{\expandafter\reserved@a\string^^J\@}
317 \ifx\reserved@a\empty\else\gdef\TeXversion{3}\fi}
318 \fi
319 \fi

(End of definition for \TeXversion.)

320 %% ---- END temporary definitions for bootstrapping ----
321 </dircheck>

```

8 ltxcheck.tex

After the format has been made, and `article.cls` moved with the other files to the 'standard input directory' as specified in `install.txt`, the format may be checked by running the file `ltxcheck.tex`.

File 02

lplain.dtx

1 Plain T_EX

L^AT_EX includes almost all of the functionality of Knuth's original 'Basic Macros' That is, the plain T_EX format described in Appendix B of the T_EXBook. However, some of the user commands are not much use so, in order to save memory, we may remove them from the kernel into a package. Here is a list of the commands that may be removed (PROBABLY NOT COMPLETE).

```
\magstep \magstephalf
\mathhexbox
\vglue \vgl@
\hglue \hgl@
```

This file is by now very small as most of it has been moved to more appropriate kernel files: it may disappear completely one day.

L^AT_EX font definitions are done using NFSS2 so none of PLAIN's font definitions are in L^AT_EX.

L^AT_EX has its own tabbing environment, so PLAIN's is disabled.

L^AT_EX uses its own output routine, so most of the plain one was removed.

```
1 <*2kernel>
2 \catcode'\{=1 % left brace is begin-group character
3 \catcode'\}=2 % right brace is end-group character
4 \catcode'\$=3 % dollar sign is math shift
5 \catcode'\&=4 % ampersand is alignment tab
6 \catcode'\#=6 % hash mark is macro parameter character
7 \catcode'\^=7 % circumflex and uparrow are for superscripts
8 \catcode'\_ =8 % underline and downarrow are for subscripts
9 \catcode'\^I=10 % ascii tab is a blank space
10 \chardef\active=13 \catcode'\~=\active % tilde is active
11 \catcode'\^L=\active \def^L{\par}% ascii form-feed is \par
12 \message{catcodes,}
```

We had to define the `\catcodes` right away, before the message line, since `\message` uses the `{` and `}` characters. When INT_EX (the T_EX initializer) starts up, it has defined the following `\catcode` values:

```
\catcode'\^@=9 % ascii null is ignored
\catcode'\^M=5 % ascii return is end-line
\catcode'\ =0 % backslash is TeX escape character
\catcode'\%=14 % percent sign is comment character
\catcode'\ =10 % ascii space is blank space
\catcode'\^?=15 % ascii delete is invalid
\catcode'\A=11 ... \catcode'\Z=11 % uppercase letters
\catcode'\a=11 ... \catcode'\z=11 % lowercase letters
all others are type 12 (other)
```

Here is a list of the characters that have been specially catcoded:

```
13 \def\dospecials{\do\ \do\\\do\{\do\}\do\$\do\&%
14 \do\#\do\^\do\_ \do\% \do\~ \do\^I}
```


(not counting ascii null, linefeed, formfeed, return, delete) Each symbol in the list is preceded by , which can be defined if you want to do something to every item in the list.

We make @ signs act like letters, temporarily, to avoid conflict between user names and internal control sequences of plain format.

```
15 \catcode'@=11
```

To make the plain macros more efficient in time and space, several constant values are declared here as control sequences. If they were changed, anything could happen; so they are private symbols.

```
\@ne Small constants are defined using \chardef.
\tw@ 16 \chardef\@ne=1
\thr@@ 17 \chardef\tw@=2
\sixt@@n 18 \chardef\thr@@=3
\@cclv 19 \chardef\sixt@@n=16
       20 \chardef\@cclv=255
```

(End of definition for \@ne and others.)

```
\@cclvi Constants above 255 defined using \mathchardef.
\@m 21 \mathchardef\@cclvi=256
\@M 22 \mathchardef\@m=1000
\@MM 23 \mathchardef\@M=10000
      24 \mathchardef\@MM=20000
```

(End of definition for \@cclvi and others.)

Allocation of registers

Here are macros for the automatic allocation of \count, \box, \dimen, \skip, \muskip, and \toks registers, as well as \read and \write stream numbers, \fam codes, \language codes, and \insert numbers.

```
25 \message{registers,}
```

When a register is used only temporarily, it need not be allocated; grouping can be used, making the value previously in the register return after the close of the group. The main use of these macros is for registers that are defined by one macro and used by others, possibly at different nesting levels. All such registers should be defined through these macros; otherwise conflicts may occur, especially when two or more macro packages are being used at the same time.

Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

The following counters are reserved:

- 0 to 9 page numbering
- 10 count allocation
- 11 dimen allocation
- 12 skip allocation
- 13 muskip allocation
- 14 box allocation
- 15 toks allocation
- 16 read file allocation
- 17 write file allocation
- 18 math family allocation
- 19 language allocation
- 20 insert allocation
- 21 the most recently allocated number

22 constant -1

End of historical L^AT_EX 2.09 comments.

New counters are allocated starting with 23, 24, etc. Other registers are allocated starting with 10. This leaves 0 through 9 for the user to play with safely, except that counts 0 to 9 are considered to be the page and subpage numbers (since they are displayed during output). In this scheme, `\count 10` always contains the number of the highest-numbered counter that has been allocated, `\count 14` the highest-numbered box, etc. Inserts are given numbers 254, 253, etc., since they require a `\count`, `\dimen`, `\skip`, and `\box` all with the same number; `\count 20` contains the lowest-numbered insert that has been allocated. Of course, `\box255` is reserved for `\output`; `\count255`, `\dimen255`, and `\skip255` can be used freely.

It is recommended that macro designers always use `\global` assignments with respect to registers numbered

1, 3, 5, 7, 9,

and always non-`\global` assignments with respect to registers

0, 2, 4, 6, 8, 255.

This will prevent “save stack buildup” that might otherwise occur.

```
26 \count10=22 % allocates \count registers 23, 24, ...
27 \count11=9 % allocates \dimen registers 10, 11, ...
28 \count12=9 % allocates \skip registers 10, 11, ...
29 \count13=9 % allocates \muskip registers 10, 11, ...
30 \count14=9 % allocates \box registers 10, 11, ...
31 \count15=9 % allocates \toks registers 10, 11, ...
32 \count16=-1 % allocates input streams 0, 1, ...
33 \count17=-1 % allocates output streams 0, 1, ...
34 \count18=3 % allocates math families 4, 5, ...
35 \count19=0 % allocates \language codes 1, 2, ...
36 \count20=255 % allocates insertions 254, 253, ...
```

`\insc@unt` The insertion counter and most recent allocation.

```
\allocationnumber 37 \countdef\insc@unt=20
38 \countdef\allocationnumber=21
```

(End of definition for \insc@unt and \allocationnumber.)

`\m@ne` The constant -1 .

```
39 \countdef\m@ne=22 \m@ne=-1
```

(End of definition for \m@ne.)

`\wlog` Write on log file (only)

```
40 \def\wlog{\immediate\write\m@ne}
```

(End of definition for \wlog.)

`\count@` Here are abbreviations for the names of scratch registers that don't need to be allocated.

```
\dimen@ 41 \countdef\count@=255
```

```
\dimen@i 42 \dimendef\dimen@=0
```

```
\dimen@ii 43 \dimendef\dimen@ii=1 % global only
```

```
\skip@ 44 \dimendef\dimen@ii=2
```

```
\toks@ 45 \skipdef\skip@=0
```

```
46 \toksdef\toks@=0
```

(End of definition for \count@ and others.)

\newcount Now, we define \newcount, \newbox, etc. so that you can say \newcount\foo and \foo
\newdimen will be defined (with \countdef) to be the next counter.
\newskip To find out which counter \foo is, you can look at \allocationnumber.
\newmuskip Since there's no \boxdef command, \chardef is used to define a \newbox,
\newbox \newinsert, \newfam, and so on.
\newtoks L^AT_EX change: remove \outer from \newcount and \newdimen (FMi) This is necessary
\newread to use \newcount inside \if... later on. Also remove from \newskip, \newbox
\newwrite \newwrite and \newfam (DPC) to save later redefinition.

```
\newfam 47 </2kernel>
\newlanguage 48 <*2kernel | latexrelease>
49 <latexrelease> \IncludeInRelease{2015/01/01}%
50 <latexrelease> {\newcount}{Extended Allocation}%
51 \def\newcount {\e@alloc\count \countdef {\count10}\insecunt\float@count}
52 \def\newdimen {\e@alloc\dimen \dimendef {\count11}\insecunt\float@count}
53 \def\newskip {\e@alloc\skip \skipdef {\count12}\insecunt\float@count}
54 \def\newmuskip
55 {\e@alloc\muskip\muskipdef{\count13}\m@ne\e@alloc@top}
```

For compatibility use \chardef in the classical range.

```
56 \def\newbox {\e@alloc\box
57 {\ifnum\allocationnumber<\cclvi
58 \expandafter\chardef
59 \else
60 \expandafter\e@alloc@chardef
61 \fi}
62 {\count14}\insecunt\float@count}
63 \def\newtoks {\e@alloc\toks \toksdef{\count15}\m@ne\e@alloc@top}
64 \def\newread {\e@alloc\read \chardef{\count16}\m@ne\sixt@n}
Skip \write18 due to its traditional use as a shell-escape.
65 \ifx\directlua\@undefined
66 \def\newwrite {\e@alloc\write \chardef{\count17}\m@ne\sixt@n}
67 \else
68 \def\newwrite {\e@alloc\write
69 {\ifnum\allocationnumber=18
70 \advance\count17\@ne
71 \allocationnumber\count17 %
72 \fi
73 \global\chardef}%
74 {\count17}%
75 \m@ne
76 {128}}
77 \fi
78 \def\new@mathgroup
79 {\e@alloc\mathgroup\chardef{\count18}\m@ne\e@mathgroup@top}
80 \let\newfam\new@mathgroup
81 \ifx\directlua\@undefined
82 \def\newlanguage {\e@alloc\language \chardef{\count19}\m@ne\cclvi}
83 \else
84 \def\newlanguage {\e@alloc\language \chardef{\count19}\m@ne{16384}}
```

```

85 \fi
86 </2ekernel | latexrelease>
87 <latexrelease>\EndIncludeInRelease
88 <latexrelease>\IncludeInRelease{0000/00/00}%
89 <latexrelease>          {\newcount}{Extended Allocation}%
90 <latexrelease>\def\newcount{\alloc@0\count\countdef\insc@unt}
91 <latexrelease>\def\newdimen{\alloc@1\dimen\dimendef\insc@unt}
92 <latexrelease>\def\newskip{\alloc@2\skip\skipdef\insc@unt}
93 <latexrelease>\def\newmuskip{\alloc@3\muskip\muskipdef\@ccclvi}
94 <latexrelease>\def\newbox{\alloc@4\box\chardef\insc@unt}
95 <latexrelease>\def\newtoks{\alloc@5\toks\toksdef\@ccclvi}
96 <latexrelease>\def\newread{\alloc@6\read\chardef\sixt@@n}
97 <latexrelease>\def\newwrite{\alloc@7\write\chardef\sixt@@n}
98 <latexrelease>\def\new@mathgroup{\alloc@8\fam\chardef\sixt@@n}
99 <latexrelease>\def\newlanguage{\alloc@9\language\chardef\@ccclvi}
100 <latexrelease>\let\newfam\new@mathgroup
101 <latexrelease>\EndIncludeInRelease

```

(End of definition for \newcount and others.)

`\e@alloc@chardef` The upper limit of extended registers, which leaves this number (eg `\dimen32767`) always
`\e@alloc@top` unallocated by these macros. cf traditional `\dimen255`.

```

102 <*2ekernel | latexrelease>
103 <latexrelease>\IncludeInRelease{2015/01/01}%
104 <latexrelease>          {\e@alloc@chardef}{Extended Allocation}%

```

```

105 \ifx\directlua\@undefined
106   \ifx\widowpenalties\@undefined

```

classic tex has 2^8 registers.

```

107   \mathchardef\e@alloc@top=255
108   \let\e@alloc@chardef\chardef
109   \else

```

etex and xetex have 2^{15} registers.

```

110   \mathchardef\e@alloc@top=32767
111   \let\e@alloc@chardef\mathchardef
112   \fi
113   \else

```

luatex has 2^{16} registers.

```

114   \chardef\e@alloc@top=65535
115   \let\e@alloc@chardef\chardef
116   \fi

```

```

117 </2ekernel | latexrelease>
118 <latexrelease>\EndIncludeInRelease
119 <latexrelease>\IncludeInRelease{0000/00/00}%
120 <latexrelease>          {\e@alloc@chardef}{Extended Allocation}%
121 <latexrelease>\let\e@alloc@top\@undefined
122 <latexrelease>\let\e@alloc@chardef\@undefined
123 <latexrelease>\EndIncludeInRelease

```

(End of definition for \e@alloc@chardef and \e@alloc@top.)

`\e@mathgroup@top` The upper limit of extended math groups (`\fam`) 16 in classic T_EX and e-T_EX, but 256 in Unicode TeX variants.

```

124 <*2ekernel | latexrelease>
125 <latexrelease>\IncludeInRelease{2015/01/01}%
126 <latexrelease>          {\e@mathgroup@top}{Extended Allocation}%
127 \ifx\Umathcode\@undefined
classic and e tex have 16 fam (0–15).
128   \chardef\e@mathgroup@top=16
129   \else
xetex and luatex have 256 fam (0–255).
130   \chardef\e@mathgroup@top=256
131   \fi
132 </2ekernel | latexrelease>
133 <latexrelease>\EndIncludeInRelease
134 <latexrelease>\IncludeInRelease{0000/00/00}%
135 <latexrelease>          {\e@mathgroup@top}{Extended Allocation}%
136 <latexrelease>\let\e@mathgroup@top\@undefined
137 <latexrelease>\EndIncludeInRelease
(End of definition for \e@mathgroup@top.)

```

`\e@alloc` A modified version of `\alloc@` that takes the count register rather than just the final digit of its number (assuming `\count1x`). It also has an extra argument to give the top of the extended range.

```

          #1 #2      #3      #4      #5      #6
      \e@alloc type defcmd current top extended-top newname
      Note that if just a single allocation range is required (not omitting a range up to
      255 for inserts) then -1 should be used for the first upper bound argument, #4.
138 <*2ekernel | latexrelease>
139 <latexrelease>\IncludeInRelease{2015/01/01}{\e@alloc}{Extended Allocation}%
140 \def\e@alloc#1#2#3#4#5#6{%
141   \global\advance#3\@ne
142   \e@ch@ck{#3}{#4}{#5}#1%
143   \allocationnumber#3\relax
144   \global#2#6\allocationnumber
145   \wlog{\string#6=\string#1\the\allocationnumber}}%
146 </2ekernel | latexrelease>
147 <latexrelease>\EndIncludeInRelease
148 <latexrelease>\IncludeInRelease{0000/00/00}{\e@alloc}{Extended Allocation}%
149 <latexrelease>\let\e@alloc\@undefined
150 <latexrelease>\EndIncludeInRelease
151 <*2ekernel>
(End of definition for \e@alloc.)

```

`\e@ch@ck` Extended check command. If the first range is exceeded, bump to 256 (or 266 for counts) and try again, testing the extended range.

Allocate matching registers from the top of the extended range and add to \@freelist.

```
\extrafloats 152 </2kernel>
153 <*2kernel | latexrelease>
154 <latexrelease> \IncludeInRelease{2015/10/01}
155 <latexrelease>           {\e@ch@ck}{Extended Allocation (checking)}%
156 \gdef\e@ch@ck#1#2#3#4{%
157   \ifnum#1<#2\else
```

If we've reached the classical top limit, bump to 256 or 266 for counts (count 256–265 are reserved by the allocation system).

```
158     \ifnum#1=#2\relax
159       \global#1\@cclvi
160       \ifx\count#4\global\advance#1 10 \fi
161     \fi
```

Check we are below the extended limit.

```
162     \ifnum#1<#3\relax
163     \else
164       \errmessage{No room for a new \string#4}%
165     \fi
166   \fi}%
167 <latexrelease> \EndIncludeInRelease
168 <latexrelease> \IncludeInRelease{2015/01/01}%
169 <latexrelease>           {\e@ch@ck}{Extended Allocation (checking)}%
170 <latexrelease> \gdef\e@ch@ck#1#2#3#4{%
171 <latexrelease>   \ifnum#1<#2\else
172 <latexrelease>     \ifnum#1=#2\relax
173 <latexrelease>       #1\@cclvi
174 <latexrelease>       \ifx\count#4\advance#1 10 \fi
175 <latexrelease>     \fi
176 <latexrelease>     \ifnum#1<#3\relax
177 <latexrelease>     \else
178 <latexrelease>       \errmessage{No room for a new #4}%
179 <latexrelease>     \fi
180 <latexrelease>   \fi}%
181 <latexrelease> \EndIncludeInRelease
182 <latexrelease> \IncludeInRelease{0000/00/00}%
183 <latexrelease>           {\e@ch@ck}{Extended Allocation (checking)}%
184 <latexrelease> \let\e@ch@ck\@undefined
185 <latexrelease> \EndIncludeInRelease
186 <latexrelease> \IncludeInRelease{2015/01/01}%
187 <latexrelease>           {\extrafloats}{Extra floats}%
188 \let\float@count\@alloc@top
```

```
\extrafloats 189 \ifx\numexpr\@undefined
In classic TeX use \newinsert to allocate float boxes.
190 \def\extrafloats#1{%
191   \count@#1\relax
192   \ifnum\count@>\z@
193     \newinsert\reserved@a
194     \global\expandafter\chardef
```

```

195         \csname bx@\the\allocationnumber\endcsname\allocationnumber
196 \@cons\@freelist{\csname bx@\the\allocationnumber\endcsname}%
197 \advance\count@m@ne
198 \expandafter\extrafloats
199 \expandafter\count@
200 \fi
201 }%
202 \else

```

In e-tex take float boxes from the top of the extended range.

```

203 \def\extrafloats#1{%
204 \ifnum#1>\z@
205 \count@\numexpr\float@count-1\relax
206 \ifnum\count@<266 \ch@ck0@m@ne\insert\fi
207 \ch@ck0\count@\count
208 \ch@ck1\count@\dimen
209 \ch@ck2\count@\skip
210 \ch@ck4\count@\box
211 \global\@alloc@chardef\float@count\count@
212 \global\expandafter\@alloc@chardef
213         \csname bx@\the\float@count\endcsname\float@count
214 \@cons\@freelist{\csname bx@\the\float@count\endcsname}%
215 \expandafter\extrafloats\expandafter{\the\numexpr#1-1\expandafter}%
216 \fi}%
217 \fi
218 </2ekernel | latexrelease>
219 <latexrelease>\EndIncludeInRelease
220 <latexrelease>\IncludeInRelease{0000/00/00}%
221 <latexrelease>         {\extrafloats}{Extra floats}%
222 <latexrelease>\let\float@count\@undefined
223 <latexrelease>\let\extrafloats\@undefined
224 <latexrelease>\EndIncludeInRelease
225 <*2ekernel>

```

(End of definition for \e@ch@ck, \extrafloats, and \extrafloats.)

`\alloc@` Since `\e@alloc` was added in 2015, `\@alloc` has not been used, but was left as some legacy code calls it. However the original definition gives spurious errors once the “classic” registers run out, so it is now defined to call `\e@alloc` internally.

```

226 </2ekernel>
227 <*2ekernel | latexrelease>
228 <latexrelease>\IncludeInRelease{2020/10/01}
229 <latexrelease>         {\alloc@}{emulate alloc@}%
230 \def\alloc@#1#2#3#4{\e@alloc#2#3{\count1#1}#4\float@count}
231 </2ekernel | latexrelease>
232 <latexrelease>\EndIncludeInRelease
233 <latexrelease>\IncludeInRelease{0000/00/00}%
234 <latexrelease>         {\alloc@}{emulate alloc@}%
235 <latexrelease>\def\alloc@#1#2#3#4#5{\global\advance\count1#1\@ne
236 <latexrelease> \ch@ck#1#4#2%
237 <latexrelease> \allocationnumber\count1#1%

```

```

238 <latexrelease> \global#3#5\allocationnumber
239 <latexrelease> \wlog{\string#5=\string#2\the\allocationnumber}}
240 <latexrelease>\EndIncludeInRelease
241 <*2ekernel)

```

(End of definition for \alloc@.)

`\newinsert`

```

242 </2ekernel)
243 <*2ekernel | latexrelease)
244 <latexrelease> \IncludeInRelease{2015/10/01}
245 <latexrelease> \newinsert}{Extended \newinsert}%
246 \ifx\numexpr\@undefined

```

If e-TeX is not available use the original plain TeX definition of `\newinsert`.

```

247 \def\newinsert#1{\global\advance\insc@unt \m@ne
248 \ch@ck0\insc@unt\count
249 \ch@ck1\insc@unt\dimen
250 \ch@ck2\insc@unt\skip
251 \ch@ck4\insc@unt\box
252 \allocationnumber\insc@unt
253 \global\chardef#1\allocationnumber
254 \wlog{\string#1=\string\insert\the\allocationnumber}}
255 \else

```

The highest register allowed with `\insert`.

```

256 \ifx\directlua\@undefined
257 \chardef\@insert@top255
258 \else
259 \chardef\@insert@top\@alloc@top
260 \fi

```

If the classic registers are exhausted, take an insert from the free float list and use `\extrafloats` to add a new float to that list.

```

261 \def\newinsert#1{%
262 \@tempswafalse
263 \global\advance\insc@unt\m@ne
264 \ifnum\count10<\insc@unt
265 \ifnum\count11<\insc@unt
266 \ifnum\count12<\insc@unt
267 \ifnum\count14<\insc@unt
268 \@tempswatrue
269 \fi\fi\fi\fi
270 \if@tempswa
271 \allocationnumber\insc@unt
272 \else
273 \global\advance\insc@unt\@ne
274 \extrafloats\@ne
275 \@next\@currbox\@freelist
276 {\ifnum\@currbox<\@insert@top
277 \allocationnumber\@currbox
278 \else
279 \ch@ck0\m@ne\insert
280 \fi}%

```



```

281     {\ch@ck0\m@ne\insert}%
282 \fi
283 \global\chardef#1\allocationnumber
284 \wlog{\string#1=\string\insert\the\allocationnumber}%
285 }

286 \fi
287 </2ekernel | latexrelease>

288 <latexrelease>\EndIncludeInRelease
289 <latexrelease>\IncludeInRelease{0000/00/00}%
290 <latexrelease>      {\newinsert}{Extended \newinsert}%
291 <latexrelease>\let\@insert@top\@undefined
292 <latexrelease>\def\newinsert#1{\global\advance\insc@unt \m@ne
293 <latexrelease> \ch@ck0\insc@unt\count
294 <latexrelease> \ch@ck1\insc@unt\dimen
295 <latexrelease> \ch@ck2\insc@unt\skip
296 <latexrelease> \ch@ck4\insc@unt\box
297 <latexrelease> \allocationnumber\insc@unt
298 <latexrelease> \global\chardef#1\allocationnumber
299 <latexrelease> \wlog{\string#1=\string\insert\the\allocationnumber}}
300 <latexrelease>\EndIncludeInRelease
301 <*2ekernel>

```

(End of definition for \newinsert.)

\ch@ck

```

302 \gdef\ch@ck#1#2#3{%
303   \ifnum\count1#1<#2\else
304     \errmessage{No room for a new #3}%
305   \fi}

```

(End of definition for \ch@ck.)

\newhelp

```

306 \def\newhelp#1#2{\newtoks#1#1\expandafter{\csname#2\endcsname}}

```

(End of definition for \newhelp.)

\@inputcheck Allocate read stream for testing and output stream that is never open and thus writes to the terminal.

\@unused

```

307 \newread\@inputcheck
308 \newwrite\@unused

```

(End of definition for \@inputcheck and \@unused.)

\maxdimen Here are some examples of allocation.

\hideskip

```

309 \newdimen\maxdimen \maxdimen=16383.99999pt % the largest legal <dimen>
310 \newskip\hideskip \hideskip=-1000pt plus 1fill % negative but can grow

```

(End of definition for \maxdimen and \hideskip.)

\p@

\z@

\z@skip

\voidb@x

```

311 \newdimen\p@ \p@=1pt % this saves macro space and time
312 \newdimen\z@ \z@=0pt % can be used both for 0pt and 0
313 \newskip\z@skip \z@skip=0pt plus0pt minus0pt
314 \newbox\voidb@x % permanently void box register

```

(End of definition for `\p@` and others.)

Assign initial values to T_EX's parameters

```
315 \message{parameters,}
```

All of T_EX's numeric parameters are listed here, but the code is commented out if no special value needs to be set. INITEX makes all parameters zero except where noted.

```
316 \pretolerance=100
317 \tolerance=200 % INITEX sets this to 10000
318 \hbadness=1000
319 \vbadness=1000
320 \linepenalty=10
321 \hyphenpenalty=50
322 \exhyphenpenalty=50
323 \binoppenalty=700
324 \relpenalty=500
325 \clubpenalty=150
326 \widowpenalty=150
327 \displaywidowpenalty=50
328 \brokenpenalty=100
329 \predisplaypenalty=10000
330 % \postdisplaypenalty=0
331 % \interlinepenalty=0
332 % \floatingpenalty=0, set during \insert
333 % \outputpenalty=0, set before TeX enters \output
334 \doublehyphendemerits=10000
335 \finalhyphendemerits=5000
336 \adjdemerits=10000
337 % \looseness=0, cleared by TeX after each paragraph
338 % \pausing=0
339 % \holdinginserts=0
340 % \tracingonline=0
341 % \tracingmacros=0
342 % \tracingstats=0
343 % \tracingparagraphs=0
344 % \tracingpages=0
345 % \tracingoutput=0
```

In the past L^AT_EX used the default value of 1 for `\tracinglostchars` because this was the best it could do. This way one would at least get a warning in the `.log` file. e-T_EX improved on that and supported a value of 2 to show the warning on the terminal, so we could have changed the default when we made the e-T_EX extensions required—however, we overlooked that opportunity. In 2021 this parameter was improved on again and now also accepts the value 3 (error on the terminal). This made us realize that we should change the default. Using 3 would really be the best, but for compatibility reasons we only use 2.

```
346 \tracinglostchars=2
347 % \tracingcommands=0
348 % \tracingrestores=0
```

`\tracingstacklevels` For LuaT_EX, the `\tracingstacklevels` functionality was implemented as a callback, so here we just define the count register to hold the value of the parameter.

```

349 </2ekernel>
350 <*2ekernel | latexrelease>
351 <latexrelease>\IncludeInRelease{2021/06/01}{\tracingstacklevels}%
352 <latexrelease>          {tracingstacklevels}%
353 \ifx\directlua\@undefined
354   % \tracingstacklevels=0 % added in 2021
355 \else
356   \newcount\tracingstacklevels
357   % Code for \tracingstacklevels defined in ltfinal.dtx
358 \fi
359 <latexrelease>\EndIncludeInRelease
360 <latexrelease>
361 <latexrelease>\IncludeInRelease{0000/00/00}{\tracingstacklevels}%
362 <latexrelease>          {tracingstacklevels}%
363 <latexrelease>\ifx\directlua\@undefined
364 <latexrelease>\else
365 <latexrelease> \let\tracingstacklevels\@undefined
366 <latexrelease>\fi
367 <latexrelease>\EndIncludeInRelease
368 </2ekernel | latexrelease>
369 <*2ekernel>

(End of definition for \tracingstacklevels.)

370 \uchyph=1
371 % \lefthyphenmin=2 \righthyphenmin=3 set below
372 % \globaldefs=0
373 % \maxdeadcycles=25 % INITEX does this
374 % \hangafter=1 % INITEX does this, also TeX after each paragraph
375 % \fam=0
376 % \mag=1000 % INITEX does this
377 % \escapechar='\ % INITEX does this
378 \defaultshyphenchar='\-
379 \defaultskewchar=-1
380 % \newlinechar='\^M % INITEX does this
381 % \newlinechar=-1 \LaTeX\ sets this in ltdefs.dtx.
382 \delimiterfactor=901
383 % \time=now % TeX does this at beginning of job
384 % \day=now % TeX does this at beginning of job
385 % \month=now % TeX does this at beginning of job
386 % \year=now % TeX does this at beginning of job

    In LATEX we don't want box information in the transcript unless we do a full tracing.

387 \showboxbreadth=-1
388 \showboxdepth=-1
389 \errorcontextlines=-1
390 \hfuzz=0.1pt
391 \vfuzz=0.1pt
392 \overfullrule=5pt
393 \maxdepth=4pt
394 \splitmaxdepth=\maxdimen
395 \boxmaxdepth=\maxdimen

```

```

396 % \lineskiplimit=0pt, changed by \normalbaselines
397 \delimitershortfall=5pt
398 \nulldelimiterspace=1.2pt
399 \scriptspace=0.5pt

400 % \mathsurround=0pt
401 % \predisplaysize=0pt, set before TeX enters $$
402 % \displaywidth=0pt, set before TeX enters $$
403 % \displayindent=0pt, set before TeX enters $$

404 \parindent=20pt

405 % \hangindent=0pt, zeroed by TeX after each paragraph
406 % \hoffset=0pt
407 % \voffset=0pt
408 %
409 % \baselineskip=0pt, changed by \normalbaselines
410 % \lineskip=0pt, changed by \normalbaselines

411 \parskip=0pt plus 1pt
412 \abovedisplayskip=12pt plus 3pt minus 9pt
413 \abovedisplayshortskip=0pt plus 3pt
414 \belowdisplayskip=12pt plus 3pt minus 9pt
415 \belowdisplayshortskip=7pt plus 3pt minus 4pt

416 % \leftskip=0pt
417 % \rightskip=0pt

418 \topskip=10pt
419 \splittopskip=10pt

420 % \tabskip=0pt
421 % \spaceskip=0pt
422 % \xspaceskip=0pt

423 \parfillskip=0pt plus 1fil

```

`\normalbaselineskip` We also define special registers that function like parameters:

```

\normalbaselineskip
\normallineskip 424 \newskip\normalbaselineskip \normalbaselineskip=12pt
\normallineskiplimit 425 \newskip\normallineskip \normallineskip=1pt
426 \newdimen\normallineskiplimit \normallineskiplimit=0pt

```

(End of definition for \normalbaselineskip, \normallineskip, and \normallineskiplimit.)

`\interfootlinepenalty`

```

427 \newcount\interfootnotelinepenalty \interfootnotelinepenalty=100

```

(End of definition for \interfootlinepenalty.)

Definitions for preloaded fonts

`\magstephalf`

`\magstep`

```

428 \def\magstephalf{1095 }
429 \def\magstep#1{\ifcase#1 \@m\or 1200\or 1440\or 1728\or
430 2074\or 2488\fi\relax}

```

(End of definition for \magstephalf and \magstep.)

Macros for setting ordinary text

```

\ frenchspacing
\nonfrenchspacing 431 \def\ frenchspacing{\sfcode'\.\@m \sfcode'\?\@m \sfcode'\!\@m
432 \sfcode'\:\@m \sfcode'\;\@m \sfcode'\,\@m}
433 \def\nonfrenchspacing{\sfcode'\.3000\sfcode'\?3000\sfcode'\!3000%
434 \sfcode'\:2000\sfcode'\;1500\sfcode'\,1250 }

(End of definition for \ frenchspacing and \nonfrenchspacing.)

\normalbaselines

435 \def\normalbaselines{\lineskip\normallineskip
436 \baselineskip\normalbaselineskip \lineskiplimit\normallineskiplimit}

(End of definition for \normalbaselines.)

\M Save a bit of space by using \let here.
\I 437 \def\^M{\ } % control <return> = control <space>
438 \let\^I\^M % same for <tab>

(End of definition for \M and \I.)

\lq
\rq 439 \def\lq{' }
440 \def\rq{' }

(End of definition for \lq and \rq.)

\lbrack
\rbrack 441 \def\lbrack{[ ]}
442 \def\rbrack{[ ]}

(End of definition for \lbrack and \rbrack.)

\aa These are not from plain.tex but they are similar to other commands found here and
\AA nowhere else, being alternate input forms for characters.
443 \def \aa {\r a}
444 \def \AA {\r A}

(End of definition for \aa and \AA.)

\endgraf
\endline 445 \let\endgraf=\par
446 \let\endline=\cr

(End of definition for \endgraf and \endline. These functions are documented on page 414.)

\space

447 \def\space{ }

(End of definition for \space.)

\empty This probably ought to go altogether, but let it to the LATEX version to save space.
448 \let\empty\@empty

(End of definition for \empty.)

```

`\null`

```
449 \def\null{\hbox{}}
```

(End of definition for `\null`.)

`\bgroup`

`\egroup`

```
450 \let\bgroup={
```

```
451 \let\egroup=}
```

(End of definition for `\bgroup` and `\egroup`.)

`\obeylines` In `\obeylines`, we say `\let^^M=\obeyedline` instead of `\def^^M{\obeyedline}` since
`\obeyspaces` this allows, for example, `\let\obeyedline=\cr \obeylines \halign{...`

This is essentially a plain `TEX` trick and in its original version where you had to use to use `\let\par=\cr` not really a safe idea in `LATEX`. If anybody used this trick this now breaks (and one needs to use `\obeyedline` instead).

```
452 </2ekernel>
```

```
453 <*2ekernel | latexrelease>
```

```
454 <latexrelease>\IncludeInRelease{2022/06/01}{\obeylines}%
```

```
455 <latexrelease> {Add a redirection to obeylines and obeyspaces}%
```

If the active `^^M` escapes, e.g., into a `\write` (which is effectively in a different context) then we don't want the definition from `\obeylines` but rather a simple `\par` (in fact even the primitive one, not the `LATEX` version `\para_end`: which is only defined later).

```
456 \begingroup
```

```
457 \catcode'\^^M=active % these lines must end with %
```

```
458 \gdef\obeylines{\catcode'\^^M=active%
```

```
459 \let^^M\obeyedline%
```

The next line ending the definition is rather curious and it took me awhile to understand why rollback fails. The problem is the following: if `latexrelease` is used, then blocks of `\IncludeInRelease ... \EndIncludeInRelease` are bypassed at high speed by grabbing each as a delimited argument. However, in that case `^^M` is seen not as code but as line ending characters and in that mode `TEX` discards everything from that point onwards to the real end of the line so it works like a comment — pretty strange really (and I think due to the fact that the original pascal compiler could have some garbage showing up after the normal line ending character. Thus we really have to make sure that any closing braces is not one the same line as an `^^M`, because otherwise it would get dropped and we end with unbalanced braces and never see the `\EndIncludeInRelease` — weird. In other places it doesn't matter because we aren't using the incomplete result.

```
460 }%
```

```
461 \global\let^^M\par % this is in case ^^M appears in a \write
```

```
462 \endgroup
```

`\obeyedline` The `\obeyedline` expands by default to `\par` with whatever definition `\par` has when it is executed. It can, however, be redefined (before calling `\obeylines!`) to achieve some special effects. If you want to alter this definition when already in the scope of `\obeylines`, it has no effect (because `\let` is used above). In that case simply make another call to `\obeylines` immediately. As you are in a restricted scope all that happens is that your redefinition is applied.

For the default definition we have to use `\def` not `\let` because the meaning of `\par` can change and we want to use the one that is current when `\obeylines` act.

There is a small subtlety here: in an `\edef` the active `^^M` stayed put (because it was equal to the primitive `\par`), now `\obeyedline` expands and you get what it contains, i.e., in that case `\par`, into the `\edef` or `\mark` unless we use `\protected` on it.

```
463 \protected\gdef\obeyedline{\par}
```

The definition of `\obeyspaces` is changed in the same way and now executes `\obeyedspace` for each active space.

```
\obeyedspace 464 \global\let\obeyedspace\space

465 \begingroup
466 \catcode'\ =\active%
467 \gdef\obeyspaces{\catcode'\ \active\let =\obeyedspace}%
```

An active space elsewhere generates `\space` by default (for example in a `\write`).

```
468 \global\let =\space%
469 \endgroup

470 </2ekernel | latexrelease>
471 <latexrelease>\EndIncludeInRelease
472 <latexrelease>\IncludeInRelease{0000/00/00}{\obeylines}%
473 <latexrelease>    {Add a redirection to obeylines and obeyspaces}%
474 <latexrelease>
```

From 2019 onwards the commands are made robust (somewhat later in the kernel sources). So if we roll back they are robust, so when redefining them we have get rid of the robust payload first. Otherwise that is seen by the later rollback below, which then installs a fragile version of the new definition on top of the one we roll back to here, sigh. `\kernel@make@fragile` also changes its definition (later own) so this is done directly.

```
475 <latexrelease>\expandafter\let\csname obeylines \endcsname\@undefined
476 <latexrelease>\expandafter\let\csname obeyspace \endcsname\@undefined
477 <latexrelease>
478 <latexrelease>\begingroup
479 <latexrelease>\catcode'\^^M=\active % these lines must end with %
480 <latexrelease> \gdef\obeylines{\catcode'\^^M\active \let^^M\par %
```

Closing brace on a separate line (see comment above).

```
481 <latexrelease>          }%
```

Another pitfall: if we do a rollback `\par` is no longer the primitive, so the roll back definition needs `\let` to what is new the primitive.

```
482 <latexrelease> \global\let^^M\RawParEnd % this is in case ^^M appears in a \write
483 <latexrelease>\endgroup
484 <latexrelease>\def\obeyspaces{\catcode'\ \active}
485 <latexrelease>
486 <latexrelease>\let\obeyedline\@undefined
487 <latexrelease>\let\obeyedspace\@undefined
488 <latexrelease>\EndIncludeInRelease
489 <*2ekernel>
```

(End of definition for `\obeylines` and others.)

`\loop` We use Kabelschacht's method of doing loops, see TUB 8#2 (1987). (unless that breaks something :-). It turned out to need an extra `\relax`: see pr/642 (`\loop` could do one iteration too much in certain cases).

```
490 \long\def \loop #1\repeat{%
```

```

491 \def\iterate{#1\relax % Extra \relax
492         \expandafter\iterate\fi
493         }%
494 \iterate
495 \let\iterate\relax
496 }

```

This setting of `\repeat` is needed to make `\loop...if...repeat` skippable within another `\if...`

```
497 \let\repeat=\fi
```

(End of definition for `\loop`, `\iterate`, and `\repeat`.)

L^AT_EX defines `\smallskip`, etc. in `ltspace.dtx`.

```

\nointerlineskip
\offinterlineskip

```

```

498 \def\nointerlineskip{\prevdepth-\@m\p@}
499 \def\offinterlineskip{\baselineskip-\@m\p@
500 \lineskip\z@ \lineskiplimit\maxdimen}

```

(End of definition for `\nointerlineskip` and `\offinterlineskip`.)

```

\vglue
\hglue

```

```

501 \def\vglue{\afterassignment\vgl@\skip@=}
502 \def\vgl@{\par \dimen@\prevdepth \hrule \@height\z@
503 \nobreak\vskip\skip@ \prevdepth\dimen@}
504 \def\hglue{\afterassignment\hgl@\skip@=}
505 \def\hgl@{\leavevmode \count@\spacefactor \vrule \@width\z@
506 \nobreak\hskip\skip@ \spacefactor\count@}

```

(End of definition for `\vglue` and `\hglue`.)

L^AT_EX defines `~` in `ltdefns.dtx`.

`\slash` This generates a `/` acting a bit like `-` but still allows hyphenation in the word part preceding it (but not after).

```
507 \def\slash{/\penalty\exhyphenpenalty}
```

(End of definition for `\slash`.)

```

\break
\nobreak
\allowbreak

```

```

508 \def\break{\penalty-\@M}
509 \def\nobreak{\penalty \@M}
510 \def\allowbreak{\penalty \z@}

```

(End of definition for `\break`, `\nobreak`, and `\allowbreak`.)

```

\filbreak
\goodbreak

```

```

511 \def\filbreak{\par\vfil\penalty-200\vfilneg}
512 \def\goodbreak{\par\penalty-500 }

```

(End of definition for `\filbreak` and `\goodbreak`.)

`\eject` Define `\eject` as in plain T_EX but define `\supereject` only in the compatibility file.

```
513 \def\eject{\par\break}
```

(End of definition for `\eject`.)

`\removelastskip`

```
514 \def\removelastskip{\ifdim\lastskip=\z@\else\vskip-\lastskip\fi}
```

(End of definition for \removelastskip.)

`\smallbreak`

`\medbreak`

```
515 \def\smallbreak{\par\ifdim\lastskip<\smallskipamount
```

`\bigbreak`

```
516 \removelastskip\penalty-50\smallskip\fi}
```

```
517 \def\medbreak{\par\ifdim\lastskip<\medskipamount
```

```
518 \removelastskip\penalty-100\medskip\fi}
```

```
519 \def\bigbreak{\par\ifdim\lastskip<\bigskipamount
```

```
520 \removelastskip\penalty-200\bigskip\fi}
```

(End of definition for \smallbreak, \medbreak, and \bigbreak.)

`\m@th`

```
521 \def\m@th{\mathsurround\z@}
```

(End of definition for \m@th.)

`\underbar` Due to L^AT_EX's redefinition of `\underline` plain T_EX's `\underbar` can be done in a simpler fashion (but do we need it at all?).

```
522 \def\underbar#1{\underline{\sbox\tw@{#1}\dp\tw@\z@\box\tw@}}
```

(End of definition for \underbar.)

`\strutbox` L^AT_EX sets `\strutbox` in `\set@fontsize`.

`\strut`

```
523 \newbox\strutbox
```

```
524 \def\strut{\relax\ifmmode\copy\strutbox\else\unhcopy\strutbox\fi}
```

(End of definition for \strutbox and \strut.)

`\hidewidth` For alignment entries that can stick out.

```
525 \def\hidewidth{\hskip\hideskip}
```

(End of definition for \hidewidth.)

`\narrower`

```
526 \def\narrower{%
```

```
527 \advance\leftskip\parindent
```

```
528 \advance\rightskip\parindent}
```

(End of definition for \narrower.)

L^AT_EX defines `\ae` and similar commands elsewhere.

```
529 \chardef\%= '\%
```

```
530 \chardef\&= '\&
```

```
531 \chardef\#= '\#
```

Most text commands are actually encoding specific and therefore defined later, so commented out or removed from this file.

`\leavevmode` begins a paragraph, if necessary

```
532 \def\leavevmode{\unhbox\voidb@x}
```

(End of definition for \leavevmode.)

`\mathhexbox`

```
533 \def\mathhexbox#1#2#3{\mbox{$\m@th \mathchar"#1#2#3$}}
```

(End of definition for `\mathhexbox`.)

`\ialign`

```
534 \def\ialign{\everycr{}}\tabskip\z@skip\halign} % initialized \halign
```

(End of definition for `\ialign`.)

`\oalign`

```
\o@lign 535 \def\oalign#1{\leavevmode\vtop{\baselineskip\z@skip \lineskip.25ex%
```

```
\oalign 536 \ialign{##\crrc#1\crrc}}}
```

```
537 \def\o@lign{\lineskiplimit\z@ \oalign}
```

```
538 \def\ooalign{\lineskiplimit-\maxdimen \oalign}
```

(End of definition for `\oalign`, `\o@lign`, and `\ooalign`.)

`\sh@ft`

The definition of this macro in `plain.tex` was improved in about 1997; but as a result its usage was changed and its new definition is not appropriate for \LaTeX .

Since the version given here has been in use by \LaTeX for many years it does not seem prudent to remove it now. As far as we can tell it has only been used to define `\b` and `\d` but this cannot be certain.

```
539 \def\sh@ft#1{\dimen@.00#1ex\multiply\dimen@\fontdimen1\font
```

```
540 \kern-.0156\dimen@} % compensate for slant in lowered accents
```

(End of definition for `\sh@ft`.)

`\ltx@sh@ft`

This is the \LaTeX version of the second incarnation of the plain macro `\sh@ft`, which takes a dimension as its argument. It shifts a pseudo-accent horizontally by an amount proportional to the product of its argument and the slant-per-point (`fontdimen 1`).

```
541 \def\ltx@sh@ft #1{%
```

```
542 \dimen@ #1%
```

```
543 \kern \strip@pt
```

```
544 \fontdimen1\font \dimen@
```

```
545 } % kern by #1 times the current slant
```

(End of definition for `\ltx@sh@ft`.)

\LaTeX change: the text commands such as `\d`, `\b`, `\c`, `\copyright`, `\TeX` are now defined elsewhere.

\LaTeX change: Make `\t` work in a moving argument. Now defined elsewhere.

`\hrulefill`

\LaTeX change: `\kern\z@` added to end of `\hrulefill` and `\dotfill` to make them work

`\dotfill`

in ‘`tabular`’ and ‘`array`’ environments. (Change made 24 July 1987). \LaTeX change:

`\leavevmode` added at beginning of `\dotfill` and `\hrulefill` so that they work as expected in vertical mode.

```
546 \def\hrulefill{\leavevmode\leaders\hrule\hfill\kern\z@}
```

The box in `\dotfill` originally contained (in `plain.tex`):

```
\mkern 1.5mu .\mkern 1.5mu;
```

the width of `.44em` differs from this by `.04pt` which is probably an acceptable difference within leaders.

```
547 \def\dotfill{%
```

```
548 \leavevmode
```

```
549 \cleaders \hb@xt@ .44em{\hss.\hss}\hfill
```

```
550 \kern\z@}
```

(End of definition for \hrulefill and \dotfill.)

INITEX sets \sfcode x=1000 for all x, except that \sfcode‘X=999 for uppercase letters. The following changes are needed:

```
551 \sfcode‘\)=0 \sfcode‘\’=0 \sfcode‘\]=0
```

The \nonfrenchspacing macro will make further changes to \sfcode values.

Definitions related to output

\magnification doesn’t work in L^AT_EX.

```
def\magnification{\afterassignment\m@g\count@}
```

```
def\m@g{\mag\count@
```

```
\hsize6.5truein\vsize8.9truein\dimen\footins8truein}
```

\showoverfull The following commands are used in debugging:

```
552 \def\showoverfull{\tracingonline\@ne}
```

(End of definition for \showoverfull.)

\showoutput

\loggingoutput

```
553 \gdef\loggingoutput{\tracingoutput\@ne
```

```
554 \showboxbreadth\maxdimen\showboxdepth\maxdimen\errorstopmode}
```

```
555 \gdef\showoutput{\loggingoutput\showoverfull}
```

```
556 </2ekernel)
```

(End of definition for \showoutput and \loggingoutput.)

\tracingall

\loggingall

```
557 <latexrelease>\IncludeInRelease{2021/06/01}{\loggingall}
```

```
558 <latexrelease> \tracingstacklevels and \tracinglostchars=3}%
```

```
559 <*2ekernel | latexrelease>
```

```
560 \edef\loggingall{%
```

```
561 \tracingstats\tw@
```

```
562 \tracingpages\@ne
```

```
563 \tracinglostchars\thr@@
```

```
564 \tracingparagraphs\@ne
```

```
565 \tracinggroups\@ne
```

```
566 \tracingifs\@ne
```

```
567 \tracingscantokens\@ne
```

```
568 \tracingnesting\@ne
```

```
569 \errorcontextlines\maxdimen
```

```
570 \ifdefined\tracingstacklevels \tracingstacklevels\maxdimen \fi
```

```
571 \noexpand \loggingoutput
```

```
572 \tracingmacros\tw@
```

```
573 \tracingcommands\thr@@
```

```
574 \tracingrestores\@ne
```

```
575 \tracingassigns\@ne
```

```
576 }%
```

```
577 \def\tracingall{\showoverfull\loggingall}
```

```
578 </2ekernel | latexrelease>
```

```
579 <latexrelease>\EndIncludeInRelease
```

```
580 <latexrelease>
```

```
581 <latexrelease>\IncludeInRelease{2015/01/01}{\loggingall}{etex tracing}%
```

```
582 <latexrelease>\ifx\tracingscantokens\undefined
```

```
583 <latexrelease>\gdef\loggingall{%
```

```

584 <latexrelease> \tracingstats\tw@
585 <latexrelease> \tracingpages\@ne
586 <latexrelease> \tracinglostchars\@ne
587 <latexrelease> \tracingparagraphs\@ne
588 <latexrelease> \errorcontextlines\maxdimen
589 <latexrelease> \loggingoutput
590 <latexrelease> \tracingmacros\tw@
591 <latexrelease> \tracingcommands\tw@
592 <latexrelease> \tracingrestores\@ne
593 <latexrelease> }%
594 <latexrelease>\else
595 <latexrelease>\gdef\loggingall{%
596 <latexrelease> \tracingstats\tw@
597 <latexrelease> \tracingpages\@ne
598 <latexrelease> \tracinglostchars\tw@
599 <latexrelease> \tracingparagraphs\@ne
600 <latexrelease> \tracinggroups\@ne
601 <latexrelease> \tracingifs\@ne
602 <latexrelease> \tracingscantokens\@ne
603 <latexrelease> \tracingnesting\@ne
604 <latexrelease> \errorcontextlines\maxdimen
605 <latexrelease> \loggingoutput
606 <latexrelease> \tracingmacros\tw@
607 <latexrelease> \tracingcommands\thr@@
608 <latexrelease> \tracingrestores\@ne
609 <latexrelease> \tracingassigns\@ne
610 <latexrelease>}%
611 <latexrelease>\fi
612 <latexrelease>\gdef\tracingall{\showoverfull\loggingall}
613 <latexrelease>\EndIncludeInRelease
614 <latexrelease>
615 <latexrelease>\IncludeInRelease{0000/00/00}{\loggingall}{etex tracing}%
616 <latexrelease>\gdef\loggingall{\tracingcommands\tw@\tracingstats\tw@
617 <latexrelease> \tracingpages\@ne\tracinglostchars\@ne
618 <latexrelease> \tracingmacros\tw@\tracingparagraphs\@ne\tracingrestores\@ne
619 <latexrelease> \errorcontextlines\maxdimen\loggingoutput}
620 <latexrelease> \gdef\tracingall{\loggingall\showoverfull}
621 <latexrelease>\EndIncludeInRelease

```

(End of definition for \tracingall and \loggingall.)

`\tracingnone`

```

622 <latexrelease>\IncludeInRelease{2015/01/01}{\tracingnone}%
623 <latexrelease> \turn off etex tracing}%
624 <*2ekernel | latexrelease>
625 \edef\tracingnone{%
626 \tracingassigns\z@
627 \tracingrestores\z@
628 \tracingonline\z@
629 \tracingcommands\z@
630 \showboxdepth\m@ne
631 \showboxbreadth\m@ne
632 \tracingoutput\z@
633 \errorcontextlines\m@ne

```

```

634 \ifdefined\tracingstacklevels \tracingstacklevels\z@ \fi
635 \tracingnesting\z@
636 \tracingscantokens\z@
637 \tracingifs\z@
638 \tracinggroups\z@
639 \tracingparagraphs\z@
640 \tracingmacros\z@

```

None really means go back to the L^AT_EX “default” and for `\tracinglostchars` this should therefore be 2 these days.

```

641 \tracinglostchars\tw@
642 \tracingpages\z@
643 \tracingstats\z@
644 }%
645 </2ekernel | latexrelease>
646 <latexrelease>\EndIncludeInRelease
647 <latexrelease>
648 <latexrelease>\IncludeInRelease{2015/01/01}{\tracingnone}%
649 <latexrelease> {turn off etex tracing}%
650 <latexrelease>\ifx\tracingscantokens\@undefined
651 <latexrelease>\def\tracingnone{%
652 <latexrelease> \tracingonline\z@
653 <latexrelease> \tracingcommands\z@
654 <latexrelease> \showboxdepth\m@ne
655 <latexrelease> \showboxbreadth\m@ne
656 <latexrelease> \tracingoutput\z@
657 <latexrelease> \errorcontextlines\m@ne
658 <latexrelease> \tracingrestores\z@
659 <latexrelease> \tracingparagraphs\z@
660 <latexrelease> \tracingmacros\z@
661 <latexrelease> \tracinglostchars\@ne
662 <latexrelease> \tracingpages\z@
663 <latexrelease> \tracingstats\z@
664 <latexrelease>}%
665 <latexrelease>\else
666 <latexrelease>\def\tracingnone{%
667 <latexrelease> \tracingassigns\z@
668 <latexrelease> \tracingrestores\z@
669 <latexrelease> \tracingonline\z@
670 <latexrelease> \tracingcommands\z@
671 <latexrelease> \showboxdepth\m@ne
672 <latexrelease> \showboxbreadth\m@ne
673 <latexrelease> \tracingoutput\z@
674 <latexrelease> \errorcontextlines\m@ne
675 <latexrelease> \tracingnesting\z@
676 <latexrelease> \tracingscantokens\z@
677 <latexrelease> \tracingifs\z@
678 <latexrelease> \tracinggroups\z@
679 <latexrelease> \tracingparagraphs\z@
680 <latexrelease> \tracingmacros\z@
681 <latexrelease> \tracinglostchars\@ne
682 <latexrelease> \tracingpages\z@
683 <latexrelease> \tracingstats\z@
684 <latexrelease>}%

```

```

685 <latexrelease>\fi
686 <latexrelease>\EndIncludeInRelease
687 <latexrelease>
688 <latexrelease>\IncludeInRelease{0000/00/00}{\tracingnone}%
689 <latexrelease>                                {turn off etex tracing}%
690 <latexrelease>\let\tracingnone\@undefined
691 <latexrelease>\EndIncludeInRelease

```

(End of definition for \tracingnone.)

\hideoutput

```

692 <*2ekernel | latexrelease>
693 <latexrelease>\IncludeInRelease{2015/01/01}{\hideoutput}%
694 <latexrelease>                                {hide output from tracing}%
695 \def\hideoutput{%
696   \tracingoutput\z@
697   \showboxbreadth\m@ne
698   \showboxdepth\m@ne
699   \tracingonline\m@ne
700 }%
701 <latexrelease>\EndIncludeInRelease
702 <latexrelease>
703 <latexrelease>\IncludeInRelease{0000/00/00}{\hideoutput}%
704 <latexrelease>                                {hide output from tracing}%
705 <latexrelease>\let\hideoutput\@undefined
706 <latexrelease>\EndIncludeInRelease
707 </2ekernel | latexrelease>

```

(End of definition for \hideoutput.)

\LaTeX change: \showhyphens Defined later.
 Punctuation affects the spacing.

```

708 <*2ekernel>
709 \nonfrenchspacing
710 </2ekernel>

```

File 03

ltxvers.dtx

1 Version Identification

First we identify the date and version number of this release of L^AT_EX, and set `\everyjob` so that it is printed at the start of every L^AT_EX run.

`\fmtname` A `\patch@level` of 0 or higher denotes an official public release. A negative value
`\fmtversion` indicates a candidate release that is not distributed.
`\latexreleaseversion` If we put code updates into the kernel that are supposed to go into the next release
`\patch@level` we set the `\patch@level` to -1 and the `\fmtversion` / `\latexreleaseversion` to the
dated of the next release (guessed, the real value is not so important and will get corrected
when we make the release official).

If the `\patch@level` is already at -1 we do nothing here and use the `\fmtversion`
date for any new `\IncludeInRelease` line when we add further code.

Finally, if we do make a public release we either just set the `\patch@level` to zero
(if our initial guess was good) or we also change the date and then have to additionally
change to that date on all the `\IncludeInRelease` statements that used the “guessed”
date.

```
1 <*2ekernel>
2 \def\fmtname{LaTeX2e}
3 \edef\fmtversion
4 </2ekernel>
5 (latexrelease)\edef\latexreleaseversion
6 <*2ekernel | latexrelease>
7     {2025-06-01}
8 </2ekernel | latexrelease>
9 <*2ekernel>
10 \def\patch@level{-2}
```

For more fine grain control there is the possibility to name the current development
branch. This is only used when the `\patch@level` is negative (i.e., a pre-release for-
mat) and is intended to help us internally when we locally install a format out of some
development branch.

```
\development@branch@name 11 \edef\development@branch@name{develop \the\year-\the\month-\the\day}
```

(End of definition for `\fmtname` and others.)

Check that the format being made is not too old. The error message complains
about ‘more than 5 years’ but in fact the error is not triggered until 65 months.

This code is currently not activated as we don’t know if we already got to the last
official 2e version (due to staff shortage or due to a successor (think positive:-)).

```
12 \iffalse
13 \def\reserved@a#1/#2/#3@nil{%
14     \count@\year
15     \advance\count@-#1\relax
16     \multiply\count@ by 12\relax
17     \advance\count@\month
18     \advance\count@-#2\relax}
19 \expandafter\reserved@a\fmtversion@nil
```

\count@ is now the age of this file in months. Take a generous definition of 'year' so this message is not generated too often.

```
20 \ifnum\count@>65
21   \typeout{^^J%
22   !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!^^J%
23   ! You are attempting to make a LaTeX format from a source file^^J%
24   ! That is more than five years old.^^J%
25   !^^J%
26   ! If you enter <return> to scroll past this message then the format^^J%
27   ! will be built, but please consider obtaining newer source files^^J%
28   ! before continuing to build LaTeX.^^J%
29   !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!^^J%
30 }
31   \errhelp{To avoid this error message, obtain new LaTeX sources.}
32   \errmsg{LaTeX source files more than 5 years old!}
33 \fi
34 \let\reserved@a\relax
35 \fi
36 \newtoks\LaTeXReleaseInfo
37 \everyjob\expandafter{\the\everyjob\the\LaTeXReleaseInfo}
38 \let\show@release@info\typeout
39 \ifnum0\ifnum\patch@level=0 \ifx\development@branch@name\@empty 1\fi\fi>0 %
40   \LaTeXReleaseInfo\expandafter{\the\LaTeXReleaseInfo
41     \show@release@info{\fmtname\space <\fmtversion>}}
42   \immediate
43     \write16{\fmtname \space<\fmtversion>}
44 \else\ifnum\patch@level>0
45   \LaTeXReleaseInfo\expandafter{\the\LaTeXReleaseInfo
46     \show@release@info{\fmtname\space <\fmtversion> patch level \patch@level}}
47   \immediate
48     \write16{\fmtname\space <\fmtversion> patch level \patch@level}
49 \else
50   \LaTeXReleaseInfo\expandafter{\the\LaTeXReleaseInfo
51     \show@release@info{\fmtname\space <\fmtversion>
52       pre-release-\number-\patch@level\space
53       \ifx\development@branch@name\@undefined \else
54         \ifx\development@branch@name\@empty \else
55           \space (\development@branch@name\space branch)%
56         \fi
57       \fi
58     }}
59   \immediate
60     \write16{\fmtname\space <\fmtversion>
61       pre-release-\number-\patch@level\space
62       \ifx\development@branch@name\@undefined \else
63         \ifx\development@branch@name\@empty \else
64           \space (\development@branch@name\space branch)%
65         \fi
66       \fi
67   }
```



```

68     \fi
69     \fi
70 \</2ekernel>

\IncludeInRelease
\EndIncludeInRelease
\@IncludeInRelease
\@IncludeInRele@se
\@gobble@IncludeInRelease
\@check@IncludeInRelease
71 <2ekernel>\let\@currname\@empty
72 <*2ekernel | latexrelease>
73 <latexrelease>\newif\if@includeinrelease
74 <latexrelease>\@includeinreleasefalse
75 \def\IncludeInRelease#1{%
76   \if@includeinrelease
77     \PackageError{latexrelease}{mis-matched IncludeInRelease}%
78       {There is an \string\EndIncludeRelease\space missing}%
79   \@includeinreleasefalse
80   \fi
81   \ifnum0%
82     \ifx\new@moduledate\@empty\else 1\fi
83     \ifnum \expandafter\@parse@version#1//00\@nil=0 1\fi
84     =1
85     \expandafter\@firstoftwo
86   \else
87     \expandafter\@secondoftwo
88   \fi
89   {\finish@module@release{#1}}%
90   {\kernel@ifnextchar[%
91     {\@IncludeInRelease{#1}}
92     {\@IncludeInRelease{#1}[#1]}}%
93 \def\finish@module@release#1#2#3{%
94   \toks@{[#1] #3}%
95   \begingroup
96     \edef\x{\detokenize\expandafter{\new@modulename}}%
97     \edef\y{\detokenize{#2}}%
98   \expandafter\endgroup
99   \ifx\x\y \else
100     \@latex@error{\noexpand\IncludeInRelease dated #1 in a module is not
101       allowed.\MessageBreak Use a date at least equal to \new@moduledate
102       \space for complete rollback}\@ehd
103   \fi
104   \ifnum\expandafter\@parse@version\new@moduledate//00\@nil
105     >\expandafter\@parse@version\fmtversion//00\@nil
106     \GenericInfo{]{Applying: \the\toks@}%
107   \else
108     \GenericInfo{]{Skipping: \the\toks@}%
109     \expandafter\gobble@finish@module@release
110   \fi}
111 \long\def\gobble@finish@module@release#1\EndModuleRelease{%
112   \EndModuleRelease}

    If a specific date has not been specified in latexrelease use ‘#1’.
113 \def\@IncludeInRelease#1[#2]{\@IncludeInRele@se{#2}}
114 \def\@IncludeInRele@se#1#2#3{%
115   \toks@{[#1] #3}%
116   \expandafter\ifx\curname\string#2+\@currname+IIR\endcurname\relax

```

If we roll back and the first patch already match then applying that is actually reapplying what is already in the format, i.e., it is useless and possibly allocating new registers. However, it makes the logic simpler so this is the way it is for now. In theory we could always jump over the first patch because that is only really needed for rolling forward. So maybe one day ...

```

117 \ifnum\expandafter\@parse@version#1//00\@nil
118 >\expandafter\@parse@version\fmtversion//00\@nil
119 \GenericInfo{}{Skipping: \the\toks@}%
120 \expandafter\expandafter\expandafter@gobble@IncludeInRelease
121 \else
122 \GenericInfo{}{Applying: \the\toks@}%
123 \@includeinreleasetrue
124 \expandafter\let\csname\string#2+\@currname+IIR\endcsname\@empty
125 \fi
126 \else
127 \GenericInfo{}{Already applied: \the\toks@}%
128 \expandafter@gobble@IncludeInRelease
129 \fi
130 }

131 \def\EndIncludeInRelease{%
132 \if@includeinrelease
133 \@includeinreleasefalse
134 \else
135 \PackageError{latexrelease}{mis-matched EndIncludeInRelease}{}%
136 \fi
137 \if@skipping@module
138 \expandafter\new@module@skip
139 \fi}

140 \long\def@gobble@IncludeInRelease#1\EndIncludeInRelease{%
141 \@includeinreleasefalse
142 \@check@IncludeInRelease#1\IncludeInRelease\@check@IncludeInRelease
143 \@end@check@IncludeInRelease}

144 \long\def@check@IncludeInRelease#1\IncludeInRelease
145 #2#3\@end@check@IncludeInRelease{%
146 \ifx\@check@IncludeInRelease#2\else
147 \PackageError{latexrelease}{skipped IncludeInRelease for tag \string#2}{}%
148 \fi
149 \if@skipping@module
150 \expandafter\new@module@skip
151 \fi}

```

(End of definition for `\IncludeInRelease` and others.)

1.1 Declaring an all-new module

```

\if@skipping@module \NewModuleRelease \EndModuleRelease
\new@module@skip \new@modulename \new@moduledate
152 \let\if@skipping@module\iffalse
153 \def\@skipping@moduletrue{\let\if@skipping@module\iftrue}
154 \def\@skipping@modulefalse{\let\if@skipping@module\iffalse}

```

When we have a whole new module, we can't roll back to a date where such module exists, otherwise hundreds of "command already defined" errors will pop up. But we can't skip it altogether either, because the module might have changes we still want applied, so a more detailed cherry-picking of code chunks have to be done.

```

155 \let\new@modulename\@empty
156 \let\new@moduledate\@empty
157 \def\NewModuleRelease#1#2#3{%
158   \ifx\new@modulename\@empty \else
159     \@latex@error{Nested \noexpand\NewModuleRelease forbidden.}\@ehd \fi
160   \edef\new@moduledate{#1}%
161   \edef\new@modulename{#2}%
162   \GenericInfo{}{BEGIN module: \new@modulename\space (\new@moduledate)}%
163   \GenericInfo{}{ \spaces\@spaces\@spaces\space#3\@gobble}%
164   \ifnum\sourceLaTeXdate<%
165     \expandafter\@parse@version\new@moduledate//00\@nil\relax
166     \ifnum\expandafter\@parse@version\fmtversion//00\@nil<%
167       \expandafter\@parse@version\new@moduledate//00\@nil\relax
168       \GenericInfo{}{Skipping module \new@modulename}%
169       \expandafter\expandafter
170       \expandafter\gobble@finish@module@release
171     \else
172       \GenericInfo{}{Applying module \new@modulename}
173       \@skipping@modulefalse
174     \fi
175   \else
176     \GenericInfo{}{Skipping module \new@modulename}
177     \@skipping@moduletrue
178     \expandafter\new@module@skip
179   \fi}
180 \long\def\new@module@skip#1\IncludeInRelease{%
181 \long\def\reserved@a##1\EndModuleRelease{}%
182 \ifrelax\detokenize\expandafter{\reserved@a#1{}}\EndModuleRelease}\relax
183 \else
184   \@latex@error{Missing mandatory \string\IncludeInRelease{0000/00/00}}\@ehc
185   \expandafter\@secondoftwo
186 \fi
187 \@gobble
188   {\@expandtwoargs\IncludeInRelease
189     {0000/00/00}{\new@modulename}%
190     {ERROR! Emergency recovery}%
191     #1}%
192 \IncludeInRelease}
193 \def\EndModuleRelease{%
194   \ifx\new@modulename\@empty
195     \@latex@error{Extra \string\EndModuleRelease.}\@eha
196   \else
197     \GenericInfo{}{END module: \new@modulename\space (\new@moduledate)}%
198     \let\new@modulename\@empty
199     \let\new@moduledate\@empty
200     \@skipping@modulefalse
201   \fi}

```

(End of definition for \if@skipping@module and others.)

```

202 </2ekernel | latexrelease>

```

File 04

ltxuatex.dtx

1 Overview

LuaTeX adds a number of engine-specific functions to TeX. Several of these require set up that is best done in the kernel or need related support functions. This file provides *basic* support for LuaTeX at the L^ATeX 2_ε kernel level plus as a loadable file which can be used with plain TeX and L^ATeX.

This file contains code for both TeX (to be stored as part of the format) and Lua (to be loaded at the start of each job). In the Lua code, the kernel uses the namespace `luatexbase`.

The following `\count` registers are used here for register allocation:

```
\e@alloc@attribute@count Attributes (default 258)
\e@alloc@ccodetable@count Category code tables (default 259)
\e@alloc@luafunction@count Lua functions (default 260)
  \e@alloc@whatsit@count User whatsits (default 261)
  \e@alloc@bytecode@count Lua bytecodes (default 262)
  \e@alloc@luachunk@count Lua chunks (default 263)
```

(`\count 256` is used for `\newmarks` allocation and `\count 257` is used for `\newXeTeXintercharclass` with XeTeX, with code defined in `ltxfinal.dtx`). With any L^ATeX 2_ε kernel from 2015 onward these registers are part of the block in the extended area reserved by the kernel (prior to 2015 the L^ATeX 2_ε kernel did not provide any functionality for the extended allocation area).

2 Core TeX functionality

The commands defined here are defined for possible inclusion in a future L^ATeX format, however also extracted to the file `ltxuatex.tex` which may be used with older L^ATeX formats, and with plain TeX.

```
\newattribute \newattribute{<attribute>}
  Defines a named \attribute, indexed from 1 (i.e. \attribute0 is never defined). Attributes initially have the marker value -"7FFFFFFF ('unset') set by the engine.
\newcatcodetable \newcatcodetable{<catcodetable>}
  Defines a named \catcodetable, indexed from 1 (\catcodetable0 is never assigned). A new catcode table will be populated with exactly those values assigned by IniTeX (as described in the LuaTeX manual).
\newluafunction \newluafunction{<function>}
  Defines a named \luafunction, indexed from 1. (Lua indexes tables from 1 so \luafunction0 is not available).
\newluacmd \newluacmd{<function>}
  Like \newluafunction, but defines the command using \luadef instead of just assigning an integer.
```

<code>\newprotectedluacmd</code>	<code>\newluadef{⟨function⟩}</code>	Like <code>\newluacmd</code> , but the defined command is not expandable.
<code>\newwhatsit</code>	<code>\newwhatsit{⟨whatsit⟩}</code>	Defines a custom <code>\whatsit</code> , indexed from 1.
<code>\newluabytecode</code>	<code>\newluabytecode{⟨bytecode⟩}</code>	Allocates a number for Lua bytecode register, indexed from 1.
<code>\newluachunkname</code>	<code>newluachunkname{⟨chunkname⟩}</code>	Allocates a number for Lua chunk register, indexed from 1. Also enters the name of the register (without backslash) into the <code>lua.name</code> table to be used in stack traces.
<code>\catcodetable@initex</code>		Predefined category code tables with the obvious assignments. Note that the <code>latex</code> and
<code>\catcodetable@string</code>		<code>atletter</code> tables set the full Unicode range to the codes predefined by the kernel.
<code>\catcodetable@latex</code>	<code>\setattribute{⟨attribute⟩}{⟨value⟩}</code>	
<code>\catcodetable@atletter</code>	<code>\unsetattribute{⟨attribute⟩}</code>	
<code>\setattribute</code>		Set and unset attributes in a manner analogous to <code>\setlength</code> . Note that attributes
<code>\unsetattribute</code>		take a marker value when unset so this operation is distinct from setting the value to zero.

3 Plain T_EX interface

The `ltluatex` interface may be used with plain T_EX using `\input{ltluatex}`. This inputs `ltluatex.tex` which inputs `etex.src` (or `etex.sty` if used with L^AT_EX) if it is not already input, and then defines some internal commands to allow the `ltluatex` interface to be defined.

The `luatexbase` package interface may also be used in plain T_EX, as before, by inputting the package `\input luatexbase.sty`. The new version of `luatexbase` is based on this `ltluatex` code but implements a compatibility layer providing the interface of the original package.

4 Lua functionality

4.1 Allocators in Lua

<code>new_attribute</code>	<code>luatexbase.new_attribute(⟨attribute⟩)</code>	Returns an allocation number for the <code>⟨attribute⟩</code> , indexed from 1. The attribute will be initialised with the marker value <code>-"7FFFFFFF</code> ('unset'). The attribute allocation sequence is shared with the T _E X code but this function does <i>not</i> define a token using <code>\attributedef</code> . The attribute name is recorded in the <code>attributes</code> table. A metatable is provided so that the table syntax can be used consistently for attributes declared in T _E X or Lua.
<code>new_whatsit</code>	<code>luatexbase.new_whatsit(⟨whatsit⟩)</code>	Returns an allocation number for the custom <code>⟨whatsit⟩</code> , indexed from 1.
<code>new_bytecode</code>	<code>luatexbase.new_bytecode(⟨bytecode⟩)</code>	Returns an allocation number for a bytecode register, indexed from 1. The optional <code>⟨name⟩</code> argument is just used for logging.
<code>new_chunkname</code>	<code>luatexbase.new_chunkname(⟨chunkname⟩)</code>	Returns an allocation number for a Lua chunk name for use with <code>\directlua</code> and <code>\latelua</code> , indexed from 1. The number is returned and also <code>⟨name⟩</code> argument is added to the <code>lua.name</code> array at that index.
<code>new_luafunction</code>	<code>luatexbase.new_luafunction(⟨functionname⟩)</code>	

Returns an allocation number for a lua function for use with `\luafunction`, `\lataluafunction`, and `\luadef`, indexed from 1. The optional `<functionname>` argument is just used for logging.

These functions all require access to a named T_EX count register to manage their allocations. The standard names are those defined above for access from T_EX, e.g. `\e@alloc@attribute@count`, but these can be adjusted by defining the variable `<type>_count_name` before loading `ltluatex.lua`, for example

```
local attribute_count_name = "attributetracker"
require("ltluatex")
```

would use a T_EX `\count` (`\countdef`'d token) called `attributetracker` in place of `\e@alloc@attribute@count`.

4.2 Lua access to T_EX register numbers

`registernumber` `luatexbase.registernumber(<name>)`

Sometimes (notably in the case of Lua attributes) it is necessary to access a register *by number* that has been allocated by T_EX. This package provides a function to look up the relevant number using LuaT_EX's internal tables. After for example `\newattribute\myattrib, \myattrib` would be defined by (say) `\myattrib=\attribute15`. `luatexbase.registernumber("myattrib")` would then return the register number, 15 in this case. If the string passed as argument does not correspond to a token defined by `\attributedef`, `\countdef` or similar commands, the Lua value `false` is returned.

As an example, consider the input:

```
\newcommand\test[1]{%
\typeout{#1: \expandafter\meaning\csname#1\endcsname^^J
\space\space\space\space
\directlua{tex.write(luatexbase.registernumber("#1") or "bad input")}%
}}

\test{undefinedrubbish}

\test{space}

\test{hbox}

\test{@MM}

\test{@tempdima}
\test{@tempdimb}

\test{strutbox}

\test{sixt@@n}

\attributedef\myattr=12
\myattr=200
\test{myattr}
```

If the demonstration code is processed with Lua \LaTeX then the following would be produced in the log and terminal output.

```

undefinedrubbish: \relax
    bad input
space: macro:->
    bad input
hbox: \hbox
    bad input
@MM: \mathchar"4E20
    20000
@tempdima: \dimen14
    14
@tempdimb: \dimen15
    15
strutbox: \char"B
    11
sxt@@n: \char"10
    16
myattr: \attribute12
    12

```

Notice how undefined commands, or commands unrelated to registers do not produce an error, just return `false` and so print `bad input` here. Note also that commands defined by `\newbox` work and return the number of the box register even though the actual command holding this number is a `\chardef` defined token (there is no `\boxdef`).

4.3 Module utilities

`provides_module` `luatexbase.provides_module(<info>)`

This function is used by modules to identify themselves; the `info` should be a table containing information about the module. The required field `name` must contain the name of the module. It is recommended to provide a field `date` in the usual \LaTeX format `yyyy/mm/dd`. Optional fields `version` (a string) and `description` may be used if present. This information will be recorded in the log. Other fields are ignored. If the `version` begins with a digit, a `v` will be added at the start in the log.

`module_info` `luatexbase.module_info(<module>, <text>)`

`module_warning` `luatexbase.module_warning(<module>, <text>)`

`module_error` `luatexbase.module_error(<module>, <text>)`

These functions are similar to \LaTeX 's `\PackageError`, `\PackageWarning` and `\PackageInfo` in the way they format the output. No automatic line breaking is done, you may still use `\n` as usual for that, and the name of the package will be prepended to each output line.

Note that `luatexbase.module_error` raises an actual Lua error with `error()`, which currently means a call stack will be dumped. While this may not look pretty, at least it provides useful information for tracking the error down.

4.4 Callback management

`add_to_callback` `luatexbase.add_to_callback(<callback>, <function>, <description>)` Registers the `<function>` into the `<callback>` with a textual `<description>` of the function. Functions are inserted into the callback in the order loaded.

`remove_from_callback` `luatexbase.remove_from_callback(<callback>, <description>)` Removes the callback function with *<description>* from the *<callback>*. The removed function and its description are returned as the results of this function.

`in_callback` `luatexbase.in_callback(<callback>, <description>)` Checks if the *<description>* matches one of the functions added to the list for the *<callback>*, returning a boolean value.

`disable_callback` `luatexbase.disable_callback(<callback>)` Sets the *<callback>* to `false` as described in the LuaTeX manual for the underlying `callback.register` built-in. Callbacks will only be set to `false` (and thus be skipped entirely) if there are no functions registered using the callback.

`callback_descriptions` A list of the descriptions of functions registered to the specified callback is returned. `{}` is returned if there are no functions registered.

`create_callback` `luatexbase.create_callback(<name>,<type>,<default>)` Defines a user defined callback. The last argument is a default function or `false`.

`call_callback` `luatexbase.call_callback(<name>,...)` Calls a user defined callback with the supplied arguments.

`declare_callback_rule` `luatexbase.declare_callback_rule(<name>, <first>, <relation>, <second>)` Adds an ordering constraint between two callback functions for callback *<name>*.
The kind of constraint added depends on *<relation>*:

before The callback function with description *<first>* will be executed before the function with description *<second>*.

after The callback function with description *<first>* will be executed after the function with description *<second>*.

incompatible-warning When both a callback function with description *<first>* and with description *<second>* is registered, then a warning is printed when the callback is executed.

incompatible-error When both a callback function with description *<first>* and with description *<second>* is registered, then an error is printed when the callback is executed.

unrelated Any previously declared callback rule between *<first>* and *<second>* gets disabled.

Every call to `declare_callback_rule` with a specific callback *<name>* and descriptions *<first>* and *<second>* overwrites all previous calls with same callback and descriptions.

The callback functions do not have to be registered yet when the functions is called. Only the constraints for which both callback descriptions refer to callbacks registered at the time the callback is called will have an effect.

5 Implementation

```

1 <*2ekernel | tex | latexrelease>
2 <2ekernel | latexrelease>\ifx\directlua\@undefined\else

```

5.1 Minimum LuaTeX version

LuaTeX has changed a lot over time. In the kernel support for ancient versions is not provided: trying to build a format with a very old binary therefore gives some information

in the log and loading stops. The cut-off selected here relates to the tree-searching behaviour of `require()`: from version 0.60, LuaTeX will correctly find Lua files in the `texmf` tree without ‘help’.

```

3 <latexrelease>\IncludeInRelease{2015/10/01}
4 <latexrelease>          {\newluafunction}{LuaTeX}%
5 \ifnum\luatexversion<60 %
6   \wlog{*****}
7   \wlog{* LuaTeX version too old for ltuatex support *}
8   \wlog{*****}
9   \expandafter\endinput
10 \fi

```

Two simple L^AT_EX macros from `ltdfn.dtx` have to be defined here because `ltdfn.dtx` is not loaded yet when `ltluatex.dtx` is executed.

```

11 \long\def@gobble#1{}
12 \long\def@firstofone#1{#1}

```

5.2 Older L^AT_EX/Plain T_EX setup

```

13 <*tex>

```

Older L^AT_EX formats don’t have the primitives with ‘native’ names: sort that out. If they already exist this will still be safe.

```

14 \directlua{tex.enableprimitives("", tex.extraprimitives("luatex"))}
15 \ifx\@alloc\@undefined

```

In pre-2014 L^AT_EX, or plain T_EX, load `etex.{sty,src}`.

```

16 \ifx\documentclass\@undefined
17   \ifx\loccount\@undefined
18     \input{etex.src}%
19   \fi
20   \catcode'\@=11 %
21   \outer\expandafter\def\csname newfam\endcsname
22     {\alloc@8\fam\chardef\et@xmaxfam}
23 \else
24   \RequirePackage{etex}
25   \expandafter\def\csname newfam\endcsname
26     {\alloc@8\fam\chardef\et@xmaxfam}
27   \expandafter\let\expandafter\new@mathgroup\csname newfam\endcsname
28 \fi

```

5.2.1 Fixes to `etex.src/etex.sty`

These could and probably should be made directly in an update to `etex.src` which already has some LuaTeX-specific code, but does not define the correct range for LuaTeX.

2015-07-13 higher range in `luatex`.

```

29 \edef \et@xmaxregs {\ifx\directlua\@undefined 32768\else 65536\fi}

```

`luatex/xetex` also allow more math fam.

```

30 \edef \et@xmaxfam {\ifx\Umathcode\@undefined\sixt@@n\else@cclvi\fi}
31 \count 270=\et@xmaxregs % locally allocates \count registers
32 \count 271=\et@xmaxregs % ditto for \dimen registers
33 \count 272=\et@xmaxregs % ditto for \skip registers
34 \count 273=\et@xmaxregs % ditto for \muskip registers
35 \count 274=\et@xmaxregs % ditto for \box registers

```

```

36 \count 275=\et@xmaxregs % ditto for \toks registers
37 \count 276=\et@xmaxregs % ditto for \marks classes
    and 256 or 16 fam. (Done above due to plain/LATEX differences in ltuatex.)
38 % \outer\def\newfam{\alloc@8\fam\chardef\et@xmaxfam}
    End of proposed changes to etex.src

```

5.2.2 luatex specific settings

Switch to global cf `luatex.sty` to leave room for inserts not really needed for `luatex` but possibly most compatible with existing use.

```

39 \expandafter\let\csname newcount\expandafter\expandafter\endcsname
    \csname globcount\endcsname
40
41 \expandafter\let\csname newdimen\expandafter\expandafter\endcsname
    \csname globdimen\endcsname
42
43 \expandafter\let\csname newskip\expandafter\expandafter\endcsname
    \csname globskip\endcsname
44
45 \expandafter\let\csname newbox\expandafter\expandafter\endcsname
    \csname globbox\endcsname
46

```

Define `\e@alloc` as in L^AT_EX (the existing macros in `etex.src` are hard to extend to further register types as they assume specific 26x and 27x count range). For compatibility the existing register allocation is not changed.

```

47 \chardef\e@alloc@top=65535
48 \let\e@alloc@chardef\chardef
49 \def\e@alloc#1#2#3#4#5#6{%
50   \global\advance#3\@ne
51   \e@ch@ck{#3}{#4}{#5}#1%
52   \allocationnumber#3\relax
53   \global#2#6\allocationnumber
54   \wlog{\string#6=\string#1\the\allocationnumber}}%
55 \gdef\e@ch@ck#1#2#3#4{%
56   \ifnum#1<#2\else
57     \ifnum#1=#2\relax
58       #1\@cclvi
59       \ifx\count#4\advance#1 10 \fi
60     \fi
61     \ifnum#1<#3\relax
62     \else
63       \errmessage{No room for a new \string#4}%
64     \fi
65   \fi}%

```

Fix up allocations not to clash with `etex.src`.

```

66 \expandafter\csname newcount\endcsname\e@alloc@attribute@count
67 \expandafter\csname newcount\endcsname\e@alloc@ccodetable@count
68 \expandafter\csname newcount\endcsname\e@alloc@luafunction@count
69 \expandafter\csname newcount\endcsname\e@alloc@whatsit@count
70 \expandafter\csname newcount\endcsname\e@alloc@bytecode@count
71 \expandafter\csname newcount\endcsname\e@alloc@luachunk@count

```

End of conditional setup for plain T_EX / old L^AT_EX.

```

72 \fi
73 </tex>

```

5.3 Attributes

`\newattribute` As is generally the case for the LuaTeX registers we start here from 1. Notably, some code assumes that `\attribute0` is never used so this is important in this case.

```
74 \ifx\@alloc@attribute@count\undefined
75   \countdef\@alloc@attribute@count=258
76   \@alloc@attribute@count=\z@
77 \fi
78 \def\newattribute#1{%
79   \@alloc@attribute@attributedef
80   \@alloc@attribute@count\m@ne\@alloc@top#1%
81 }
```

(End of definition for `\newattribute`.)

`\setattribute` Handy utilities.
`\unsetattribute`

```
82 \def\setattribute#1#2{#1=\numexpr#2\relax}
83 \def\unsetattribute#1{#1=-"7FFFFFFF\relax}
```

(End of definition for `\setattribute` and `\unsetattribute`.)

5.4 Category code tables

`\newcatcodetable` Category code tables are allocated with a limit half of that used by LuaTeX for everything else. At the end of allocation there needs to be an initialization step. Table 0 is already taken (it's the global one for current use) so the allocation starts at 1.

```
84 \ifx\@alloc@ccodetable@count\undefined
85   \countdef\@alloc@ccodetable@count=259
86   \@alloc@ccodetable@count=\z@
87 \fi
88 \def\newcatcodetable#1{%
89   \@alloc@catcodetable\chardef
90   \@alloc@ccodetable@count\m@ne{"8000}#1%
91   \initcatcodetable\allocationnumber
92 }
```

(End of definition for `\newcatcodetable`.)

`\catcodetable@initex` Save a small set of standard tables. The Unicode data is read here in using a parser simplified from that in `load-unicode-data`: only the nature of letters needs to be detected.
`\catcodetable@string`
`\catcodetable@latex`
`\catcodetable@atletter`

```
93 \newcatcodetable\catcodetable@initex
94 \newcatcodetable\catcodetable@string
95 \begingroup
96   \def\setstrangeatcode#1#2#3{%
97     \ifnum#1>#2 %
98       \expandafter\@gobble
99     \else
100      \expandafter\@firstofone
101     \fi
102     {%
103       \catcode#1=#3 %
104       \expandafter\setstrangeatcode\expandafter
105       {\number\numexpr#1 + 1\relax}{#2}{#3}
106     }%
```

```

107 }
108 \@firstofone{%
109   \catcodetable\catcodetable@initex
110     \catcode0=12 %
111     \catcode13=12 %
112     \catcode37=12 %
113     \setrangecatcode{65}{90}{12}%
114     \setrangecatcode{97}{122}{12}%
115     \catcode92=12 %
116     \catcode127=12 %
117     \savecatcodetable\catcodetable@string
118   \endgroup
119 }%
120 \newcatcodetable\catcodetable@latex
121 \newcatcodetable\catcodetable@atletter
122 \beginingroup
123   \def\parseunicodedataI#1;#2;#3;#4\relax{%
124     \parseunicodedataII#1;#3;#2 First>\relax
125   }%
126   \def\parseunicodedataII#1;#2;#3 First>#4\relax{%
127     \ifx\relax#4\relax
128       \expandafter\parseunicodedataIII
129     \else
130       \expandafter\parseunicodedataIV
131     \fi
132     {#1}#2\relax%
133   }%
134   \def\parseunicodedataIII#1#2#3\relax{%
135     \ifnum 0%
136       \if L#21\fi
137       \if M#21\fi
138       >0 %
139       \catcode"#1=11 %
140     \fi
141   }%
142   \def\parseunicodedataIV#1#2#3\relax{%
143     \read\unicoderead to \unicodedataline
144     \if L#2%
145       \count0="#1 %
146       \expandafter\parseunicodedataV\unicodedataline\relax
147     \fi
148   }%
149   \def\parseunicodedataV#1;#2\relax{%
150     \loop
151       \unless\ifnum\count0>"#1 %
152         \catcode\count0=11 %
153         \advance\count0 by 1 %
154     \repeat
155   }%
156   \def\storedpar{\par}%
157   \chardef\unicoderead=\numexpr\count16 + 1\relax
158   \openin\unicoderead=UnicodeData.txt %
159   \loop\unless\ifeof\unicoderead %
160     \read\unicoderead to \unicodedataline

```

```

161 \unless\ifx\unicodedataline\storedpar
162 \expandafter\parseunicodedataI\unicodedataline\relax
163 \fi
164 \repeat
165 \closein\unicoderead
166 \@firstofone{%
167 \catcode64=12 %
168 \savecatcodetable\catcodetable@latex
169 \catcode64=11 %
170 \savecatcodetable\catcodetable@atletter
171 }
172 \endgroup

```

(End of definition for \catcodetable@initex and others.)

5.5 Named Lua functions

`\newluafunction` Much the same story for allocating Lua_{TEX} functions except here they are just numbers so they are allocated in the same way as boxes. Lua indexes from 1 so once again slot 0 is skipped.

```

173 \ifx\e@alloc@luafunction@count\@undefined
174 \countdef\e@alloc@luafunction@count=260
175 \e@alloc@luafunction@count=\z@
176 \fi
177 \def\newluafunction{%
178 \e@alloc\luafunction\e@alloc@chardef
179 \e@alloc@luafunction@count\m@ne\e@alloc@top
180 }

```

(End of definition for \newluafunction.)

`\newluacmd` Additionally two variants are provided to make the passed control sequence call the function directly.

`\newprotectedluacmd`

```

181 \def\newluacmd{%
182 \e@alloc\luafunction\luadef
183 \e@alloc@luafunction@count\m@ne\e@alloc@top
184 }
185 \def\newprotectedluacmd{%
186 \e@alloc\luafunction{\protected\luadef}
187 \e@alloc@luafunction@count\m@ne\e@alloc@top
188 }

```

(End of definition for \newluacmd and \newprotectedluacmd.)

5.6 Custom whatsits

`\newwhatsit` These are only settable from Lua but for consistency are definable here.

```

189 \ifx\e@alloc@whatsit@count\@undefined
190 \countdef\e@alloc@whatsit@count=261
191 \e@alloc@whatsit@count=\z@
192 \fi
193 \def\newwhatsit#1{%
194 \e@alloc\whatsit\e@alloc@chardef
195 \e@alloc@whatsit@count\m@ne\e@alloc@top#1%
196 }

```

(End of definition for `\newwhatsit`.)

5.7 Lua bytecode registers

`\newluabytocode` These are only settable from Lua but for consistency are definable here.

```
197 \ifx\@alloc@bytecode@count\@undefined
198   \countdef\@alloc@bytecode@count=262
199   \@alloc@bytecode@count=\z@
200 \fi
201 \def\newluabytocode#1{%
202   \@alloc\luabytocode\@alloc@chardef
203   \@alloc@bytecode@count\m@ne\@alloc@top#1%
204 }
```

(End of definition for `\newluabytocode`.)

5.8 Lua chunk registers

`\newluachunkname` As for bytecode registers, but in addition we need to add a string to the `lua.name` table to use in stack tracing. We use the name of the command passed to the allocator, with no backslash.

```
205 \ifx\@alloc@luachunk@count\@undefined
206   \countdef\@alloc@luachunk@count=263
207   \@alloc@luachunk@count=\z@
208 \fi
209 \def\newluachunkname#1{%
210   \@alloc\luachunk\@alloc@chardef
211   \@alloc@luachunk@count\m@ne\@alloc@top#1%
212   {\escapechar\m@ne
213    \directlua{lua.name[\the\allocationnumber]="\string#1"}}%
214 }
```

(End of definition for `\newluachunkname`.)

5.9 Lua loader

Lua code loaded in the format often has to be loaded again at the beginning of every job, so we define a helper which allows us to avoid duplicated code:

```
215 \def\now@and@everyjob#1{%
216   \everyjob\expandafter{\the\everyjob
217     #1%
218   }%
219   #1%
220 }
```

Load the Lua code at the start of every job. For the conversion of \TeX into numbers at the Lua side we need some known registers: for convenience we use a set of systematic names, which means using a group around the Lua loader.

```
221 \langle 2kernel \rangle\now@and@everyjob{%
222   \begingroup
223   \attributedef\attributezero=0 %
224   \chardef \charzero =0 %
```

Note name change required on older luatex, for hash table access.

```

225 \countdef \CountZero =0 %
226 \dimendef \dimenzero =0 %
227 \mathchardef \mathcharzero =0 %
228 \muskipdef \muskipzero =0 %
229 \skipdef \skipzero =0 %
230 \toksdef \tokszero =0 %
231 \directlua{require("ltxlua")}
232 \endgroup
233 <2kernel>}
234 <latexrelease>\EndIncludeInRelease

235 <latexrelease>\IncludeInRelease{0000/00/00}
236 <latexrelease> {\newluafunction}{LuaTeX}%
237 <latexrelease>\let\@alloc@attribute@count\@undefined
238 <latexrelease>\let\newattribute\@undefined
239 <latexrelease>\let\setattribute\@undefined
240 <latexrelease>\let\unsetattribute\@undefined
241 <latexrelease>\let\@alloc@ccodetable@count\@undefined
242 <latexrelease>\let\newcatcodetable\@undefined
243 <latexrelease>\let\catcodetable@initex\@undefined
244 <latexrelease>\let\catcodetable@string\@undefined
245 <latexrelease>\let\catcodetable@latex\@undefined
246 <latexrelease>\let\catcodetable@atletter\@undefined
247 <latexrelease>\let\@alloc@luafunction@count\@undefined
248 <latexrelease>\let\newluafunction\@undefined
249 <latexrelease>\let\@alloc@luafunction@count\@undefined
250 <latexrelease>\let\newwhatsit\@undefined
251 <latexrelease>\let\@alloc@whatsit@count\@undefined
252 <latexrelease>\let\newluabytecode\@undefined
253 <latexrelease>\let\@alloc@bytecode@count\@undefined
254 <latexrelease>\let\newluachunkname\@undefined
255 <latexrelease>\let\@alloc@luachunk@count\@undefined
256 <latexrelease>\directlua{luaotfload.uninstall()}
257 <latexrelease>\EndIncludeInRelease

```

In \everyjob, if luaotfload is available, load it and switch to TU.

```

258 <latexrelease>\IncludeInRelease{2017/01/01}%
259 <latexrelease> {\fontencoding}{TU in everyjob}%
260 <latexrelease>\fontencoding{TU}\let\encodingdefault\f@encoding
261 <latexrelease>\ifx\directlua\@undefined\else
262 <2kernel>\everyjob\expandafter{%
263 <2kernel> \the\everyjob
264 <*2kernel, latexrelease>
265 \directlua{%
266 if xpcall(function ()%
267 require('luaotfload-main')%
268 end, texio.write_nl) then %
269 local _void = luaotfload.main ()%
270 else %
271 texio.write_nl('Error in luaotfload: reverting to OT1')%
272 tex.print('\string\def\string\encodingdefault{OT1}')%
273 end %
274 }%

```

```

275 \let\f@encoding\encodingdefault
276 \expandafter\let\csname ver@luaotfload.sty\endcsname\fmtversion
277 </2ekernel, latexrelease>
278 <latexrelease>\fi
279 <2ekernel> }
280 <latexrelease>\EndIncludeInRelease
281 <latexrelease>\IncludeInRelease{0000/00/00}%
282 <latexrelease>          {\fontencoding}{TU in everyjob}%
283 <latexrelease>\fontencoding{OT1}\let\encodingdefault\f@encoding
284 <latexrelease>\EndIncludeInRelease

285 <2ekernel | latexrelease>\fi
286 </2ekernel | tex | latexrelease>

```

5.10 Lua module preliminaries

```
287 <*lua>
```

Some set up for the Lua module which is needed for all of the Lua functionality added here.

luatexbase Set up the table for the returned functions. This is used to expose all of the public functions.

```
288 luatexbase      = luatexbase or { }
289 local luatexbase = luatexbase
```

(End of definition for luatexbase.)

Some Lua best practice: use local versions of functions where possible.

```
290 local string_gsub      = string.gsub
291 local tex_count       = tex.count
292 local tex_setcount    = tex.setcount
293 local texio_write_nl  = texio.write_nl
294 local flush_list     = node.flush_list

295 local luatexbase_warning
296 local luatexbase_error
```

5.11 Lua module utilities

5.11.1 Module tracking

modules To allow tracking of module usage, a structure is provided to store information and to return it.

```
297 local modules = modules or { }
```

(End of definition for modules.)

provides_module Local function to write to the log.

```
298 local function luatexbase_log(text)
299   texio_write_nl("log", text)
300 end
```


Modelled on `\ProvidesPackage`, we store much the same information but with a little more structure.

```

301 local function provides_module(info)
302   if not (info and info.name) then
303     luatexbase_error("Missing module name for provides_module")
304   end
305   local function spaced(text)
306     return text and (" " .. text) or ""
307   end
308   luatexbase_log(
309     "Lua module: " .. info.name
310     .. spaced(info.date)
311     .. spaced(info.version and string_gsub(info.version or "", "^(%d)", "v%1"))
312     .. spaced(info.description)
313   )
314   modules[info.name] = info
315 end
316 luatexbase.provides_module = provides_module

```

(End of definition for provides_module.)

5.11.2 Module messages

There are various warnings and errors that need to be given. For warnings we can get exactly the same formatting as from `TEX`. For errors we have to make some changes. Here we give the text of the error in the `LATEX` format then force an error from Lua to halt the run. Splitting the message text is done using `\n` which takes the place of `\MessageBreak`.

First an auxiliary for the formatting: this measures up the message leader so we always get the correct indent.

```

317 local function msg_format(mod, msg_type, text)
318   local leader = ""
319   local cont
320   local first_head
321   if mod == "LaTeX" then
322     cont = string_gsub(leader, ".", " ")
323     first_head = leader .. "LaTeX: "
324   else
325     first_head = leader .. "Module " .. msg_type
326     cont = "(" .. mod .. ")"
327     .. string_gsub(first_head, ".", " ")
328     first_head = leader .. "Module " .. mod .. " " .. msg_type .. ":"
329   end
330   if msg_type == "Error" then
331     first_head = "\n" .. first_head
332   end
333   if string.sub(text,-1) ~= "\n" then
334     text = text .. " "
335   end
336   return first_head .. " "
337     .. string_gsub(
338       text
339       .. "on input line "

```

```

340         .. tex.inputlineno, "\n", "\n" .. cont .. " "
341     )
342     .. "\n"
343 end

```

```

module_info Write messages.
module_warning
module_error
344 local function module_info(mod, text)
345     texio_write_nl("log", msg_format(mod, "Info", text))
346 end
347 luatexbase.module_info = module_info
348 local function module_warning(mod, text)
349     texio_write_nl("term and log",msg_format(mod, "Warning", text))
350 end
351 luatexbase.module_warning = module_warning
352 local function module_error(mod, text)
353     error(msg_format(mod, "Error", text))
354 end
355 luatexbase.module_error = module_error

(End of definition for module_info, module_warning, and module_error.)
    Dedicated versions for the rest of the code here.

356 function luatexbase_warning(text)
357     module_warning("luatexbase", text)
358 end
359 function luatexbase_error(text)
360     module_error("luatexbase", text)
361 end

```

5.12 Accessing register numbers from Lua

Collect up the data from the T_EX level into a Lua table: from version 0.80, LuaT_EX makes that easy.

```

362 local luaregisterbasetable = { }
363 local registermap = {
364     attributezero = "assign_attr"    ,
365     charzero      = "char_given"     ,
366     CountZero     = "assign_int"     ,
367     dimenzero     = "assign_dimen"   ,
368     mathcharzero  = "math_given"     ,
369     muskipzero    = "assign_mu_skip" ,
370     skipzero      = "assign_skip"    ,
371     tokzero       = "assign_toks"    ,
372 }
373 local createtoken
374 if tex.luatexversion > 81 then
375     createtoken = token.create
376 elseif tex.luatexversion > 79 then
377     createtoken = newtoken.create
378 end
379 local hashtokens = tex.hashtokens()
380 local luatexversion = tex.luatexversion
381 for i,j in pairs (registermap) do
382     if luatexversion < 80 then

```

```

383     luaregisterbasetable[hashtokens[i][1]] =
384     hashtokens[i][2]
385   else
386     luaregisterbasetable[j] = createtoken(i).mode
387   end
388 end

```

registernumber Working out the correct return value can be done in two ways. For older LuaTeX releases it has to be extracted from the `hashtokens`. On the other hand, newer LuaTeX's have `newtoken`, and whilst `.mode` isn't currently documented, Hans Hagen pointed to this approach so we should be OK.

```

389 local registernumber
390 if luatexversion < 80 then
391   function registernumber(name)
392     local nt = hashtokens[name]
393     if(nt and luaregisterbasetable[nt[1]]) then
394       return nt[2] - luaregisterbasetable[nt[1]]
395     else
396       return false
397     end
398   end
399 else
400   function registernumber(name)
401     local nt = createtoken(name)
402     if(luaregisterbasetable[nt.cmdname]) then
403       return nt.mode - luaregisterbasetable[nt.cmdname]
404     else
405       return false
406     end
407   end
408 end
409 luatexbase.registernumber = registernumber

```

(End of definition for registernumber.)

5.13 Attribute allocation

new_attribute As attributes are used for Lua manipulations its useful to be able to assign from this end.

```

410 local attributes=setmetatable(
411   {},
412   {
413     __index = function(t,key)
414       return registernumber(key) or nil
415     end}
416 )
417 luatexbase.attributes = attributes
418 local attribute_count_name =
419     attribute_count_name or "e@alloc@attribute@count"
420 local function new_attribute(name)
421   tex_setcount("global", attribute_count_name,
422               tex_count[attribute_count_name] + 1)
423   if tex_count[attribute_count_name] > 65534 then
424     luatexbase_error("No room for a new \\attribute")

```

```

425 end
426 attributes[name]= tex_count[attribute_count_name]
427 luatexbase_log("Lua-only attribute " .. name .. " = " ..
428               tex_count[attribute_count_name])
429 return tex_count[attribute_count_name]
430 end
431 luatexbase.new_attribute = new_attribute

```

(End of definition for new_attribute.)

5.14 Custom whatsit allocation

`new_whatsit` Much the same as for attribute allocation in Lua.

```

432 local whatsit_count_name = whatsit_count_name or "e@alloc@whatsit@count"
433 local function new_whatsit(name)
434   tex_setcount("global", whatsit_count_name,
435               tex_count[whatsit_count_name] + 1)
436   if tex_count[whatsit_count_name] > 65534 then
437     luatexbase_error("No room for a new custom whatsit")
438   end
439   luatexbase_log("Custom whatsit " .. (name or "") .. " = " ..
440                 tex_count[whatsit_count_name])
441   return tex_count[whatsit_count_name]
442 end
443 luatexbase.new_whatsit = new_whatsit

```

(End of definition for new_whatsit.)

5.15 Bytecode register allocation

`new_bytecode` Much the same as for attribute allocation in Lua. The optional *(name)* argument is used in the log if given.

```

444 local bytecode_count_name =
445       bytecode_count_name or "e@alloc@bytecode@count"
446 local function new_bytecode(name)
447   tex_setcount("global", bytecode_count_name,
448               tex_count[bytecode_count_name] + 1)
449   if tex_count[bytecode_count_name] > 65534 then
450     luatexbase_error("No room for a new bytecode register")
451   end
452   luatexbase_log("Lua bytecode " .. (name or "") .. " = " ..
453                 tex_count[bytecode_count_name])
454   return tex_count[bytecode_count_name]
455 end
456 luatexbase.new_bytecode = new_bytecode

```

(End of definition for new_bytecode.)

5.16 Lua chunk name allocation

`new_chunkname` As for bytecode registers but also store the name in the `lua.name` table.

```

457 local chunkname_count_name =
458       chunkname_count_name or "e@alloc@luachunk@count"
459 local function new_chunkname(name)

```

```

460 tex_setcount("global", chunkname_count_name,
461               tex_count[chunkname_count_name] + 1)
462 local chunkname_count = tex_count[chunkname_count_name]
463 chunkname_count = chunkname_count + 1
464 if chunkname_count > 65534 then
465   luatexbase_error("No room for a new chunkname")
466 end
467 lua.name[chunkname_count]=name
468 luatexbase_log("Lua chunkname " .. (name or "") .. " = " ..
469               chunkname_count .. "\n")
470 return chunkname_count
471 end
472 luatexbase.new_chunkname = new_chunkname

```

(End of definition for new_chunkname.)

5.17 Lua function allocation

`new_luafunction` Much the same as for attribute allocation in Lua. The optional `<name>` argument is used in the log if given.

```

473 local luafunction_count_name =
474       luafunction_count_name or "e@alloc@luafunction@count"
475 local function new_luafunction(name)
476   tex_setcount("global", luafunction_count_name,
477               math.max(
478                   #(lua.get_functions_table()),
479                   tex_count[luafunction_count_name]
480                   + 1)
481   lua.get_functions_table()[tex_count[luafunction_count_name]] = false
482   if tex_count[luafunction_count_name] > 65534 then
483     luatexbase_error("No room for a new luafunction register")
484   end
485   luatexbase_log("Lua function " .. (name or "") .. " = " ..
486                 tex_count[luafunction_count_name])
487   return tex_count[luafunction_count_name]
488 end
489 luatexbase.new_luafunction = new_luafunction

```

(End of definition for new_luafunction.)

5.18 Lua callback management

The native mechanism for callbacks in LuaTeX allows only one per function. That is extremely restrictive and so a mechanism is needed to add and remove callbacks from the appropriate hooks.

5.18.1 Housekeeping

The main table: keys are callback names, and values are the associated lists of functions. More precisely, the entries in the list are tables holding the actual function as `func` and the identifying description as `description`. Only callbacks with a non-empty list of functions have an entry in this list.

Actually there are two tables: `realcallbacklist` directly contains the entries as described above while `callbacklist` only directly contains the already sorted entries. Other entries can be queried through `callbacklist` too which triggers a resort.

Additionally `callbackrules` describes the ordering constraints: It contains two element tables with the descriptions of the constrained callback implementations. It can additionally contain a `type` entry indicating the kind of rule. A missing value indicates a normal ordering constraint.

```

490 local realcallbacklist = {}
491 local callbackrules = {}
492 local callbacklist = setmetatable({}, {
493   __index = function(t, name)
494     local list = realcallbacklist[name]
495     local rules = callbackrules[name]
496     if list and rules then
497       local meta = {}
498       for i, entry in ipairs(list) do
499         local t = {value = entry, count = 0, pos = i}
500         meta[entry.description], list[i] = t, t
501       end
502       local count = #list
503       local pos = count
504       for i, rule in ipairs(rules) do
505         local rule = rules[i]
506         local pre, post = meta[rule[1]], meta[rule[2]]
507         if pre and post then
508           if rule.type then
509             if not rule.hidden then
510               assert(rule.type == 'incompatible-warning' and luatexbase_warning
511                 or rule.type == 'incompatible-error' and luatexbase_error)(
512                 "Incompatible functions \"" .. rule[1] .. "\" and \"" .. rule[2]
513                 .. "\" specified for callback \"" .. name .. "\".")
514               rule.hidden = true
515             end
516           else
517             local post_count = post.count
518             post.count = post_count+1
519             if post_count == 0 then
520               local post_pos = post.pos
521               if post_pos ~= pos then
522                 local new_post_pos = list[pos]
523                 new_post_pos.pos = post_pos
524                 list[post_pos] = new_post_pos
525               end
526               list[pos] = nil
527               pos = pos - 1
528             end
529             pre[#pre+1] = post
530           end
531         end
532       end
533       for i=1, count do -- The actual sort begins
534         local current = list[i]
535         if current then

```

```

536     meta[current.value.description] = nil
537   for j, cur in ipairs(current) do
538     local count = cur.count
539     if count == 1 then
540       pos = pos + 1
541       list[pos] = cur
542     else
543       cur.count = count - 1
544     end
545   end
546   list[i] = current.value
547 else
548   -- Cycle occurred. TODO: Show cycle for debugging
549   -- list[i] = ...
550   local remaining = {}
551   for name, entry in next, meta do
552     local value = entry.value
553     list[#list + 1] = entry.value
554     remaining[#remaining + 1] = name
555   end
556   table.sort(remaining)
557   local first_name = remaining[1]
558   for j, name in ipairs(remaining) do
559     local entry = meta[name]
560     list[i + j - 1] = entry.value
561     for _, post_entry in ipairs(entry) do
562       local post_name = post_entry.value.description
563       if not remaining[post_name] then
564         remaining[post_name] = name
565       end
566     end
567   end
568   local cycle = {first_name}
569   local index = 1
570   local last_name = first_name
571   repeat
572     cycle[last_name] = index
573     last_name = remaining[last_name]
574     index = index + 1
575     cycle[index] = last_name
576   until cycle[last_name]
577   local length = index - cycle[last_name] + 1
578   table.move(cycle, cycle[last_name], index, 1)
579   for i=2, length//2 do
580     cycle[i], cycle[length + 1 - i] = cycle[length + 1 - i], cycle[i]
581   end
582   error('Cycle occurred at ' .. table.concat(cycle, ' -> ', 1, length))
583 end
584 end
585 end
586 realcallbacklist[name] = list
587 t[name] = list
588 return list
589 end

```

```
590 })
```

Numerical codes for callback types, and name-to-value association (the table keys are strings, the values are numbers).

```
591 local list, data, exclusive, simple, reverselist = 1, 2, 3, 4, 5
592 local types = {
593   list      = list,
594   data      = data,
595   exclusive = exclusive,
596   simple    = simple,
597   reverselist = reverselist,
598 }
```

Now, list all predefined callbacks with their current type, based on the LuaTeX manual version 1.01. A full list of the currently-available callbacks can be obtained using

```
\directlua{
  for i,_ in pairs(callback.list()) do
    texio.write_nl("- " .. i)
  end
}
```

```
\bye
```

in plain LuaTeX. (Some undocumented callbacks are omitted as they are to be removed.)

```
599 local callbacktypes = callbacktypes or {
```

Section 8.2: file discovery callbacks.

```
600   find_read_file      = exclusive,
601   find_write_file     = exclusive,
602   find_font_file      = data,
603   find_output_file    = data,
604   find_format_file    = data,
605   find_vf_file        = data,
606   find_map_file       = data,
607   find_enc_file       = data,
608   find_pk_file        = data,
609   find_data_file      = data,
610   find_opentype_file  = data,
611   find_truetype_file  = data,
612   find_type1_file     = data,
613   find_image_file     = data,
614   open_read_file      = exclusive,
615   read_font_file      = exclusive,
616   read_vf_file        = exclusive,
617   read_map_file       = exclusive,
618   read_enc_file       = exclusive,
619   read_pk_file        = exclusive,
620   read_data_file      = exclusive,
621   read_truetype_file  = exclusive,
622   read_type1_file     = exclusive,
623   read_opentype_file  = exclusive,
```

Not currently used by luatex but included for completeness. may be used by a font handler.


```
624 find_cidmap_file = data,  
625 read_cidmap_file = exclusive,
```

Section 8.3: data processing callbacks.

```
626 process_input_buffer = data,  
627 process_output_buffer = data,  
628 process_jobname = data,
```

Section 8.4: node list processing callbacks.

```
629 contribute_filter = simple,  
630 buildpage_filter = simple,  
631 build_page_insert = exclusive,  
632 pre_linebreak_filter = list,  
633 linebreak_filter = exclusive,  
634 append_to_vlist_filter = exclusive,  
635 post_linebreak_filter = reverselist,  
636 hpack_filter = list,  
637 vpack_filter = list,  
638 hpack_quality = exclusive,  
639 vpack_quality = exclusive,  
640 pre_output_filter = list,  
641 process_rule = exclusive,  
642 hyphenate = simple,  
643 ligaturing = simple,  
644 kerning = simple,  
645 insert_local_par = simple,  
646 % mlist_to_hlist = exclusive,  
647 new_graf = exclusive,
```

Section 8.5: information reporting callbacks.

```
648 pre_dump = simple,  
649 start_run = simple,  
650 stop_run = simple,  
651 start_page_number = simple,  
652 stop_page_number = simple,  
653 show_error_hook = simple,  
654 show_warning_message = simple,  
655 show_error_message = simple,  
656 show_lua_error_hook = simple,  
657 start_file = simple,  
658 stop_file = simple,  
659 call_edit = simple,  
660 finish_synctex = simple,  
661 wrapup_run = simple,
```

Section 8.6: PDF-related callbacks.

```
662 finish_pdffile = data,  
663 finish_pdfpage = data,  
664 page_objnum_provider = data,  
665 page_order_index = data,  
666 process_pdf_image_content = data,
```

Section 8.7: font-related callbacks.

```
667 define_font = exclusive,  
668 glyph_info = exclusive,  
669 glyph_not_found = exclusive,
```

```

670 glyph_stream_provider      = exclusive,
671 make_extensible            = exclusive,
672 font_descriptor_objnum_provider = exclusive,
673 input_level_string          = exclusive,
674 provide_charproc_data       = exclusive,
675 }
676 luatexbase.callbacktypes=callbacktypes

```

Sometimes multiple callbacks correspond to a single underlying engine level callback. Then the engine level callback should be registered as long as at least one of these callbacks is in use. This is implemented through a shared table which counts how many of the involved callbacks are currently in use. The engine level callback is registered iff this count is not 0.

We add `mlist_to_hlist` directly to the list to demonstrate this, but the handler gets added later when it is actually defined.

All callbacks in this list are treated as user defined callbacks.

```

677 local shared_callbacks = {
678   mlist_to_hlist = {
679     callback = "mlist_to_hlist",
680     count = 0,
681     handler = nil,
682   },
683 }
684 shared_callbacks.pre_mlist_to_hlist_filter = shared_callbacks.mlist_to_hlist
685 shared_callbacks.post_mlist_to_hlist_filter = shared_callbacks.mlist_to_hlist

```

`callback.register` Save the original function for registering callbacks and prevent the original being used. The original is saved in a place that remains available so other more sophisticated code can override the approach taken by the kernel if desired.

```

686 local callback_register = callback_register or callback.register
687 function callback.register()
688   luatexbase_error("Attempt to use callback.register() directly\n")
689 end

```

(End of definition for `callback.register`.)

5.18.2 Handlers

The handler function is registered into the callback when the first function is added to this callback's list. Then, when the callback is called, the handler takes care of running all functions in the list. When the last function is removed from the callback's list, the handler is unregistered.

More precisely, the functions below are used to generate a specialized function (closure) for a given callback, which is the actual handler.

The way the functions are combined together depends on the type of the callback. There are currently 4 types of callback, depending on the calling convention of the functions the callback can hold:

simple is for functions that don't return anything: they are called in order, all with the same argument;

data is for functions receiving a piece of data of any type except node list head (and possibly other arguments) and returning it (possibly modified): the functions are called in order, and each is passed the return value of the previous (and the other arguments untouched, if any). The return value is that of the last function;

list is a specialized variant of *data* for functions filtering node lists. Such functions are called with a node list head as the first argument and may return either the head of a modified node list, or the boolean values **true** or **false**. The functions are chained the same way as for *data* except for the following cases. If a function returns **false**, then **false** is immediately returned and the following functions are *not* called. If a function returns **true**, then the same head is passed to the next function. If all functions return **true**, then the original head is returned, otherwise the return value of the last function not returning **true** is used.

reverselist is a specialized variant of *list* which executes functions in inverse order.

exclusive is for functions with more complex signatures; functions in this type of callback are *not* combined: An error is raised if a second callback is registered.

Handler for **data** callbacks.

```

690 local function data_handler(name)
691   return function(data, ...)
692     for _,i in ipairs(callbacklist[name]) do
693       data = i.func(data,...)
694     end
695     return data
696   end
697 end

```

Default for user-defined **data** callbacks without explicit default.

```

698 local function data_handler_default(value)
699   return value
700 end

```

Handler for **exclusive** callbacks. We can assume `callbacklist[name]` is not empty: otherwise, the function wouldn't be registered in the callback any more.

```

701 local function exclusive_handler(name)
702   return function(...)
703     return callbacklist[name][1].func(...)
704   end
705 end

```

Handler for **list** callbacks.

```

706 local function list_handler(name)
707   return function(head, ...)
708     local ret
709     for _,i in ipairs(callbacklist[name]) do
710       ret = i.func(head, ...)
711       if ret == false then
712         luatexbase_warning(
713           "Function '" .. i.description .. "' returned false\n"
714           .. "in callback '" .. name .. "'")
715       )
716       return false
717     end

```

```

718     if ret ~= true then
719         head = ret
720     end
721 end
722 return head
723 end
724 end

```

Default for user-defined `list` and `reverselist` callbacks without explicit default.

```

725 local function list_handler_default(head)
726 return head
727 end

```

Handler for `reverselist` callbacks.

```

728 local function reverselist_handler(name)
729 return function(head, ...)
730     local ret
731     local callbacks = callbacklist[name]
732     for i = #callbacks, 1, -1 do
733         local cb = callbacks[i]
734         ret = cb.func(head, ...)
735         if ret == false then
736             luatexbase_warning(
737                 "Function '" .. cb.description .. "' returned false\n"
738                 .. "in callback '" .. name .. "'")
739         )
740         return false
741     end
742     if ret ~= true then
743         head = ret
744     end
745 end
746 return head
747 end
748 end

```

Handler for simple callbacks.

```

749 local function simple_handler(name)
750 return function(...)
751     for _,i in ipairs(callbacklist[name]) do
752         i.func(...)
753     end
754 end
755 end

```

Default for user-defined `simple` callbacks without explicit default.

```

756 local function simple_handler_default()
757 end

```

Keep a handlers table for indexed access and a table with the corresponding default functions.

```

758 local handlers = {
759     [data] = data_handler,
760     [exclusive] = exclusive_handler,
761     [list] = list_handler,
762     [reverselist] = reverselist_handler,

```

```

763 [simple]      = simple_handler,
764 }
765 local defaults = {
766 [data]      = data_handler_default,
767 [exclusive] = nil,
768 [list]      = list_handler_default,
769 [reverselist] = list_handler_default,
770 [simple]     = simple_handler_default,
771 }

```

5.18.3 Public functions for callback management

Defining user callbacks perhaps should be in package code, but impacts on `add_to_callback`. If a default function is not required, it may be declared as `false`. First we need a list of user callbacks.

```

772 local user_callbacks_defaults = {}

```

`create_callback` The allocator itself.

```

773 local function create_callback(name, ctype, default)
774   local ctype_id = types[ctype]
775   if not name or name == ""
776   or not ctype_id
777   then
778     luatexbase_error("Unable to create callback:\n" ..
779                       "valid callback name and type required")
780   end
781   if callbacktypes[name] then
782     luatexbase_error("Unable to create callback '" .. name ..
783                       "':\ncallback is already defined")
784   end
785   default = default or defaults[ctype_id]
786   if not default then
787     luatexbase_error("Unable to create callback '" .. name ..
788                       "':\ndefault is required for '" .. ctype ..
789                       "' callbacks")
790   elseif type (default) ~= "function" then
791     luatexbase_error("Unable to create callback '" .. name ..
792                       "':\ndefault is not a function")
793   end
794   user_callbacks_defaults[name] = default
795   callbacktypes[name] = ctype_id
796 end
797 luatexbase.create_callback = create_callback

```

(End of definition for create_callback.)

`call_callback` Call a user defined callback. First check arguments.

```

798 local function call_callback(name,...)
799   if not name or name == "" then
800     luatexbase_error("Unable to create callback:\n" ..
801                       "valid callback name required")
802   end
803   if user_callbacks_defaults[name] == nil then
804     luatexbase_error("Unable to call callback '" .. name

```

```

805         .. "':\nunknown or empty")
806     end
807     local l = callbacklist[name]
808     local f
809     if not l then
810         f = user_callbacks_defaults[name]
811     else
812         f = handlers[callbacktypes[name]](name)
813     end
814     return f(...)
815 end
816 luatexbase.call_callback=call_callback

```

(End of definition for call_callback.)

`add_to_callback` Add a function to a callback. First check arguments.

```

817 local function add_to_callback(name, func, description)
818     if not name or name == "" then
819         luatexbase_error("Unable to register callback:\n" ..
820             "valid callback name required")
821     end
822     if not callbacktypes[name] or
823         type(func) ~= "function" or
824         not description or
825         description == "" then
826         luatexbase_error(
827             "Unable to register callback.\n\n"
828             .. "Correct usage:\n"
829             .. "add_to_callback(<callback>, <function>, <description>)"
830         )
831     end

```

Then test if this callback is already in use. If not, initialise its list and register the proper handler.

```

832     local l = realcallbacklist[name]
833     if l == nil then
834         l = { }
835         realcallbacklist[name] = l

```

Handle count for shared engine callbacks.

```

836     local shared = shared_callbacks[name]
837     if shared then
838         shared.count = shared.count + 1
839         if shared.count == 1 then
840             callback_register(shared.callback, shared.handler)
841         end

```

If it is not a user defined callback use the primitive callback register.

```

842     elseif user_callbacks_defaults[name] == nil then
843         callback_register(name, handlers[callbacktypes[name]](name))
844     end
845 end

```

Actually register the function and give an error if more than one exclusive one is registered.

```

846     local f = {

```

```

847     func          = func,
848     description = description,
849   }
850   if callbacktypes[name] == exclusive then
851     if #l == 1 then
852       luatexbase_error(
853         "Cannot add second callback to exclusive function\n'" ..
854         name .. "'")
855     end
856   end
857   table.insert(l, f)
858   callbacklist[name] = nil

```

Keep user informed.

```

859   luatexbase_log(
860     "Inserting '" .. description .. "' in '" .. name .. "'."
861   )
862 end
863 luatexbase.add_to_callback = add_to_callback

```

(End of definition for add_to_callback.)

declare_callback_rule Add an ordering constraint between two callback implementations

```

864 local function declare_callback_rule(name, desc1, relation, desc2)
865   if not callbacktypes[name] or
866     not desc1 or not desc2 or
867     desc1 == "" or desc2 == "" then
868     luatexbase_error(
869       "Unable to create ordering constraint. "
870       .. "Correct usage:\n"
871       .. "declare_callback_rule(<callback>, <description_a>, <description_b>)"
872     )
873   end
874   if relation == 'before' then
875     relation = nil
876   elseif relation == 'after' then
877     desc2, desc1 = desc1, desc2
878     relation = nil
879   elseif relation == 'incompatible-warning' or relation == 'incompatible-error' then
880   elseif relation == 'unrelated' then
881   else
882     luatexbase_error(
883       "Unknown relation type in declare_callback_rule"
884     )
885   end
886   callbacklist[name] = nil
887   local rules = callbackrules[name]
888   if rules then
889     for i, rule in ipairs(rules) do
890       if rule[1] == desc1 and rule[2] == desc2 or rule[1] == desc2 and rule[2] == desc1 then
891         if relation == 'unrelated' then
892           table.remove(rules, i)
893         else
894           rule[1], rule[2], rule.type = desc1, desc2, relation
895         end
896       end
897     end
898   end

```

```

896         return
897     end
898 end
899 if relation ~= 'unrelated' then
900     rules[#rules + 1] = {desc1, desc2, type = relation}
901 end
902 elseif relation ~= 'unrelated' then
903     callbackrules[name] = {{desc1, desc2, type = relation}}
904 end
905 end
906 luatexbase.declare_callback_rule = declare_callback_rule

```

(End of definition for declare_callback_rule.)

`remove_from_callback` Remove a function from a callback. First check arguments.

```

907 local function remove_from_callback(name, description)
908     if not name or name == "" then
909         luatexbase_error("Unable to remove function from callback:\n" ..
910             "valid callback name required")
911     end
912     if not callbacktypes[name] or
913         not description or
914         description == "" then
915         luatexbase_error(
916             "Unable to remove function from callback.\n\n"
917             .. "Correct usage:\n"
918             .. "remove_from_callback(<callback>, <description>)"
919         )
920     end
921     local l = realcallbacklist[name]
922     if not l then
923         luatexbase_error(
924             "No callback list for '" .. name .. "'\n")
925     end

```

Loop over the callback's function list until we find a matching entry. Remove it and check if the list is empty: if so, unregister the callback handler.

```

926     local index = false
927     for i,j in ipairs(l) do
928         if j.description == description then
929             index = i
930             break
931         end
932     end
933     if not index then
934         luatexbase_error(
935             "No callback '" .. description .. "' registered for '" ..
936             name .. "'\n")
937     end
938     local cb = l[index]
939     table.remove(l, index)
940     luatexbase_log(
941         "Removing '" .. description .. "' from '" .. name .. "'."
942     )
943     if #l == 0 then

```



```

944     realcallbacklist[name] = nil
945     callbacklist[name] = nil
946     local shared = shared_callbacks[name]
947     if shared then
948         shared.count = shared.count - 1
949         if shared.count == 0 then
950             callback_register(shared.callback, nil)
951         end
952     elseif user_callbacks_defaults[name] == nil then
953         callback_register(name, nil)
954     end
955 end
956 return cb.func,cb.description
957 end
958 luatexbase.remove_from_callback = remove_from_callback

```

(End of definition for remove_from_callback.)

in_callback Look for a function description in a callback.

```

959 local function in_callback(name, description)
960     if not name
961         or name == ""
962         or not realcallbacklist[name]
963         or not callbacktypes[name]
964         or not description then
965         return false
966     end
967     for _, i in pairs(realcallbacklist[name]) do
968         if i.description == description then
969             return true
970         end
971     end
972     return false
973 end
974 luatexbase.in_callback = in_callback

```

(End of definition for in_callback.)

disable_callback As we subvert the engine interface we need to provide a way to access this functionality.

```

975 local function disable_callback(name)
976     if(realcallbacklist[name] == nil) then
977         callback_register(name, false)
978     else
979         luatexbase_error("Callback list for " .. name .. " not empty")
980     end
981 end
982 luatexbase.disable_callback = disable_callback

```

(End of definition for disable_callback.)

callback_descriptions List the descriptions of functions registered for the given callback. This will sort the list if necessary.

```

983 local function callback_descriptions (name)
984     local d = {}
985     if not name

```

```

986     or name == ""
987     or not realcallbacklist[name]
988     or not callbacktypes[name]
989     then
990     return d
991 else
992 for k, i in pairs(callbacklist[name]) do
993     d[k]= i.description
994 end
995 end
996 return d
997 end
998 luatexbase.callback_descriptions =callback_descriptions

```

(End of definition for callback_descriptions.)

uninstall Unlike at the TeX level, we have to provide a back-out mechanism here at the same time as the rest of the code. This is not meant for use by anything other than latexrelease: as such this is *deliberately* not documented for users!

```

999 local function uninstall()
1000     module_info(
1001         "luatexbase",
1002         "Uninstalling kernel luatexbase code"
1003     )
1004     callback.register = callback_register
1005     luatexbase = nil
1006 end
1007 luatexbase.uninstall = uninstall

```

(End of definition for uninstall.)

mlist_to_hlist To emulate these callbacks, the “real” mlist_to_hlist is replaced by a wrapper calling the wrappers before and after.

```

1008 create_callback('pre_mlist_to_hlist_filter', 'list')
1009 create_callback('mlist_to_hlist', 'exclusive', node.mlist_to_hlist)
1010 create_callback('post_mlist_to_hlist_filter', 'reverselist')
1011 function shared_callbacks.mlist_to_hlist.handler(head, display_type, need_penalties)
1012     local current = call_callback("pre_mlist_to_hlist_filter", head, display_type, need_penalties)
1013     if current == false then
1014         flush_list(head)
1015         return nil
1016     end
1017     current = call_callback("mlist_to_hlist", current, display_type, need_penalties)
1018     local post = call_callback("post_mlist_to_hlist_filter", current, display_type, need_penalties)
1019     if post == false then
1020         flush_list(current)
1021         return nil
1022     end
1023     return post
1024 end

```

(End of definition for mlist_to_hlist.)

```

1025 </lua>

```

Reset the catcode of @.

```

1026 <tex>\catcode'\@=\etacatcode\relax

```

File 05

ltxexpl.dtx

1 expl3-dependent code

1.1 Loader

`\@kernel@after@enddocument` These two kernel hooks are used by the shipout code. They are defined earlier here
`\@kernel@after@enddocument@afterlastpage` because the `lthooks` code adds material to them.

```
1 <*2kernel | latexrelease>  
2 <latexrelease> \IncludeInRelease{2020/10/01}%  
3 <latexrelease> {kernel@enddocument hooks}{Define several kernel hooks}
```

We only initialize these kernel hooks if they are not already existing. Otherwise they would be set to `\@empty` on rollback which would be wrong because code that has been added to them may still have to be executed in the rollback situation. Instead code that writes to them needs to handle the rollback as needed. It is likely that we have to change that approach in the future, but for now it should do. (It is enough to test only for the existence of one hook, as all got added at the same time.)

```
4 \ifx\@kernel@after@enddocument\undefined  
5   \let \@kernel@after@enddocument \@empty  
6   \let \@kernel@after@enddocument@afterlastpage \@empty
```

For the similar reasons we also define those that are used in `\document` because they too get material added to in early modules.

```
\@kernel@before@begindocument  
\@kernel@after@begindocument  
7 \let \@kernel@before@begindocument \@empty  
8 \let \@kernel@after@begindocument \@empty  
9 \fi  
10 <latexrelease> \EndIncludeInRelease  
11 <latexrelease> \IncludeInRelease{0000/00/00}%  
12 <latexrelease> {kernel@enddocument hooks}{Define several kernel hooks}  
13 <latexrelease> \let \@kernel@after@enddocument\undefined  
14 <latexrelease> \let \@kernel@after@enddocument@afterlastpage\undefined  
15 <latexrelease> \let \@kernel@before@begindocument\undefined  
16 <latexrelease> \let \@kernel@after@begindocument\undefined  
17 </2kernel | latexrelease>  
18 <latexrelease> \EndIncludeInRelease
```

(End of definition for `\@kernel@after@enddocument` and others.)

First define some blank commands, so that in case something goes wrong while loading `expl3`, we won't get strange `Undefined control sequence` errors.

```
19 <*2kernel | latexrelease>  
20 <latexrelease> \IncludeInRelease{2020/10/01}%  
21 <latexrelease> {\@expl@sys@load@backend@}{Roll forward support}%  
22 \def\reserved@a#1{\ifdefined#1\else\def#1{}\fi}  
23 \reserved@a\@expl@sys@load@backend@@  
24 \reserved@a\@expl@push@filename@@  
25 \reserved@a\@expl@push@filename@aux@@  
26 \reserved@a\@expl@pop@filename@@  
27 <latexrelease> \EndIncludeInRelease  
28 </2kernel | latexrelease>
```

Create a hook for last-minute expl3 material.

```
29 <*2kernel>
30 \def\@expl@finalise@setup@{
31 </2kernel>
```

Now define some basics to support loading expl3. These macros can be defined here safely, because they are redefined later on by the kernel, so we define simpler versions just to suit our needs.

```
32 <*2kernel>
33 \long\def\@gobble#1{}
34 \long\def\@firstofone#1{#1}
35 \long\def\@firstoftwo#1#2{#1}
36 \long\def\@secondoftwo#1#2{#2}
37 \long\def\IfFileExists#1{%
38   \openin\@inputcheck"#1" %
39   \ifeof\@inputcheck
40     \expandafter\@secondoftwo
41   \else
42     \closein\@inputcheck
43     \expandafter\@firstoftwo
44   \fi}
45 \long\def\@ifnextchar#1#2#3{%
46   \let\reserved@d=#1%
47   \def\reserved@a{#2}%
48   \def\reserved@b{#3}%
49   \futurelet\@let@token\@ifnch}
50 \def\@ifnch{%
51   \ifx\@let@token\reserved@d
52     \expandafter\reserved@a
53   \else
54     \expandafter\reserved@b
55   \fi}
56 </2kernel>
```

If we are doing a rollback with a format containing expl3 we aren't reloading it as that creates havoc. This may need a refined version!

```
57 <*2kernel | latexrelease>
58 <latexrelease> \IncludeInRelease{2020/10/01}%
59 <latexrelease> {expl3}{Pre-load expl3}%
60 \expandafter\ifx\csname tex\string _let:D\endcsname\relax
61   \expandafter\@firstofone
62 \else
63   \GenericInfo{}{Skipping: expl3 code already part of the format}%
64 <2kernel> \expandafter\endinput
65 <latexrelease> \expandafter\@gobble
66 \fi
```

Check for the required primitive/engine support and the existence of a loader.

```
67 {%
68   \IfFileExists{expl3.ltx}
69   {%
70     \ifnum0%
71       \ifdefined\pdffilesize 1\fi
72       \ifdefined\filesize 1\fi
73       \ifdefined\luatexversion\ifnum\luatexversion>94 1\fi\fi
```

```

74         \ifdefined\kanjiskip 1\fi
75         >0 %
76         \expandafter\@firstofone
77     \else

```

In 2ekernel mode, an error is fatal and building the format is aborted. Use `\batchmode \read -1 to \tokenlist`, which errors with

```
! Emergency stop. (cannot \read from terminal in nonstop modes)
```

and aborts the T_EX run. In latexrelease mode, raise an error and do nothing. Both ways, the error message shows the minimum expl3 engine requirements.

```

78 <2ekernel>         \def~{ }\def\MessageBreak{^^J~~~~~}%
79 <2ekernel>         \errmessage{LaTeX Error:
80 <latexrelease>     \@latex@error{%
81                   LaTeX requires the e-TeX primitives and additional\MessageBreak
82                   functionality available in the engines:\MessageBreak
83                   - pdfTeX v1.40\MessageBreak
84                   - XeTeX v0.99992\MessageBreak
85                   - LuaTeX v0.95\MessageBreak
86                   - e-(u)pTeX mid-2012\MessageBreak
87                   or later}%
88 <latexrelease>     } \@ehd \expandafter\@gobble
89 <2ekernel>         } \batchmode \read -1 to \reserved@a
90     \fi
91   }
92   {%
93 <*2ekernel>
94   \errmessage{LaTeX requires expl3}%
95   \batchmode \read -1 to \reserved@a
96 </2ekernel>

```

We do not support a roll forward across 2019. You need to start with 2019 if you want to get to 2020 or beyond.

```

97 <*latexrelease>
98   \@latex@warning@no@line
99   {You need a format that already contains a recent\MessageBreak
100   expl3 as part of the kernel, e.g. at least a kernel\MessageBreak
101   from 2019 to roll forward to that date!\MessageBreak
102   --- I'm giving up!\MessageBreak\MessageBreak
103   Note that manually loading the expl3 package\MessageBreak
104   from your distribution is not enough}%
105   \batchmode \read -1 to \reserved@a
106 </latexrelease>
107   }%
108   {\input expl3.ltx }%
109 }
110 <latexrelease> \EndIncludeInRelease
111 <latexrelease>

```

To support roll-forward for the case where xparse is fully integrated into the kernel, we do not need to repeat the complex test above as we can simply look for the marker command.

```

112 <latexrelease> \IncludeInRelease{2020/02/02}%
113 <latexrelease>     {expl3}{Pre-load expl3}%

```

```

114 <latexrelease> \IfFileExists{expl3.ltx}
115 <latexrelease> {%
116 <latexrelease>     \ifnum0%
117 <latexrelease>         \ifdefined\pdffilesize 1\fi
118 <latexrelease>         \ifdefined\filesize 1\fi
119 <latexrelease>         \ifdefined\luatexversion\ifnum\luatexversion>94 1\fi\fi
120 <latexrelease>         >0 %
121 <latexrelease>     \else
122 <latexrelease>         \message{Skipping expl3-dependent extensions}
123 <latexrelease>         \expandafter\@gobbletwo
124 <latexrelease>     \fi
125 <latexrelease> }
126 <latexrelease> {%
127 <latexrelease>     \message{Skipping expl3-dependent extensions}%
128 <latexrelease>     \@gobbletwo
129 <latexrelease> }%
130 <latexrelease> \input{expl3.ltx}
131 <latexrelease> \EndIncludeInRelease

```

Now in `latexrelease` mode, redefine a few commands to avoid “already defined” errors.

```

132 <latexrelease> \@ifundefined{ExplSyntaxOff}{-}{\latexrelease@postltxpl}

```

1.2 Using `expl3` code

In order to ease the implementation of some new features in $\text{\LaTeX} 2_{\epsilon}$ we may (temporarily) use some coding based on the `expl3`-code. Such macros will eventually vanish and may be changed unannounced. They are there for internal use in the $\text{\LaTeX} 2_{\epsilon}$ kernel and are not meant to be used in third-party packages. These macros will always have the `@expl@` prefix in their name.

The rest of the name matches the `expl3` name but with all underscores replaced by `@` and the `:` replaced by `@@`, e.g.,

```

\cs_new_eq:NN \@expl@tl@trim@spaces@apply@@nN \tl_trim_spaces_apply:nN

```

if that `expl3` command is needed in places that are others coded in $\text{\LaTeX} 2_{\epsilon}$ conventions.

In this file, each release of `LaTeX` adds an `\IncludeInRelease` block, in which the macros copied for that release were defined. In case a rollback is requested, the entire block is changed.

Each macro copied has a `\changes` entry to explain when and why it was copied, so that further to that may spot it easily.

Here `\cs_gset_eq:NN` is used, instead of the `new` variant because if different releases use that same name for different purposes, each can copy the macro without worrying about redefinitions.

```

133 <latexrelease> \IncludeInRelease{2020/10/01}{\@expl@cs@to@str@@N}%
134 <latexrelease>     {expl3 macros added for the 2020-10-01 release}%

```

The `expl3` activation needs to be inside the release guards as otherwise rolling forward is broken in old kernels that do not have `expl3` loaded.

```

135 \ExplSyntaxOn

136 \cs_gset_eq:NN \@expl@cs@to@str@@N \cs_to_str:N
137 \cs_gset_eq:NN \@expl@str@if@eq@@nnTF \str_if_eq:nnTF

```

```

138 \cs_gset_eq:NN \@expl@cs@prefix@spec@N \cs_prefix_spec:N
139 \cs_if_exist:NTF \cs_parameter_spec:N
140 { \cs_gset_eq:NN \@expl@cs@parameter@spec@N \cs_parameter_spec:N }
141 { \cs_gset_eq:NN \@expl@cs@parameter@spec@N \cs_argument_spec:N }
142 \cs_gset_eq:NN \_kernel_cs_parameter_spec:N \@expl@cs@parameter@spec@N
143 \cs_gset_eq:NN \@expl@cs@replacement@spec@N \cs_replacement_spec:N

144 \cs_gset_eq:NN \@expl@str@map@function@N \str_map_function:NN
145 \cs_gset_eq:NN \@expl@char@generate@N \char_generate:nn

146 \ExplSyntaxOff

```

Here we can't assume that `expl3` is available. It will be if we roll back but if this code is executed rolling forward it needs to be pure 2e.

```

147 <latexrelease> \EndIncludeInRelease
148 <latexrelease> \IncludeInRelease{0000/00/00}{\@expl@cs@to@str@N}%
149 <latexrelease> {expl3 macros added for the 2020-10-01 release}%
150 <latexrelease> \let \@expl@cs@to@str@N \@undefined
151 <latexrelease> \let \@expl@str@if@eq@NnTF \@undefined
152 <latexrelease> \let \@expl@cs@prefix@spec@N \@undefined
153 <latexrelease> \let \@expl@cs@parameter@spec@N \@undefined
154 <latexrelease> \let \@expl@cs@replacement@spec@N \@undefined
155 <latexrelease> \let \@expl@str@map@function@N \@undefined
156 <latexrelease> \EndIncludeInRelease
157 </2ekernel | latexrelease>

```

2 Document-level command names for `expl3` functions

Current home for L3 programming layer functions that we make directly available at the document level. This section may need to be moved later (after `\NewDocumentCommand` is defined in case we want to use that in the setup).

`\fpeval` The expandable command `\fpeval` takes as its argument a floating point expression and produces a result using the normal rules of mathematics. As this command is expandable it can be used where `TEX` requires a number and for example within a low-level `\edef` operation to give a purely numerical result. See `usrguide3` for further explanation.

`\inteval` The expandable command `\inteval` takes as its argument an integer expression and produces a result using the normal rules of mathematics. The operations recognised are `\skipeval` `+`, `-`, `*` and `/` plus parentheses. Division occurs with *rounding*, and ties are rounded away from zero. As this command is expandable it can be used where `TEX` requires a number and for example within a low-level `\edef` operation to give a purely numerical result. See `usrguide3` for further explanation. `\dimeval` and `\skipeval` are similar, but generate fixed and rubber length values, respectively.

`\fpeval` A document level wrapper around the code level function for floating point calculations.
`\inteval`
`\dimeval` 158 <*2ekernel | latexrelease>
`\skipeval` 159 <latexrelease> \IncludeInRelease{2022/06/01}%
160 <latexrelease> {\fpeval}{fp and int calculations}%
161 \ExplSyntaxOn
162 \cs_new_eq:NN \fpeval \fp_eval:n

And a few more, this time wrappers around the eTeX primitives.

```

163 \cs_new_eq:NN \inteval \int_eval:n
164 \cs_new_eq:NN \dimeval \dim_eval:n
165 \cs_new_eq:NN \skipeval \skip_eval:n
166 \ExplSyntaxOff

(End of definition for \fpeval and others.)

167 </2ekernel | latexrelease>
168 <latexrelease>\EndIncludeInRelease
169 <latexrelease>\IncludeInRelease{0000/00/00}%
170 <latexrelease>          {\fpeval}{fp and int calculations}%
171 <latexrelease>
172 <latexrelease>\let\fpeval\@undefined
173 <latexrelease>\let\inteval\@undefined
174 <latexrelease>\let\dimeval\@undefined
175 <latexrelease>\let\skipeval\@undefined
176 <latexrelease>\EndIncludeInRelease

```

\UseName When declaring new commands with `\NewDocumentCommand` or `\NewCommandCopy`
\ExpandArgs or similar, it is sometimes necessary to “construct” the csname. As a general mechanism the L3 programming layer has `\exp_args:N...` for this, but there is no mechanism for it if `\ExplSyntaxOn` is not active. We therefore offer a few of these commands also with CamelCase names.

\UseName A document wrapper for changing arguments to cs names for use with `\NewDocumentCommand`
\ExpandArgs and similar functions.

```

177 <*2ekernel | latexrelease>
178 <latexrelease>\IncludeInRelease{2022/06/01}%
179 <latexrelease>          {\ExpandArgs}{Some pre-expansion commands}%
180 \ExplSyntaxOn
181 \cs_new_eq:NN \UseName \use:c
182 \cs_new:Npn \ExpandArgs #1
183 {
184   \cs_if_exist_use:cF { exp_args:N #1 }
185   { \msg_expandable_error:nnn { kernel } { unknown-arg-expansion } {#1} }
186 }
187 \msg_new:nnn { kernel } { unknown-arg-expansion }
188 { Unknown-arg-expansion-"#1" }
189 \ExplSyntaxOff

```

(End of definition for `\UseName` and `\ExpandArgs`.)

```

190 </2ekernel | latexrelease>
191 <latexrelease>\EndIncludeInRelease
192 <latexrelease>\IncludeInRelease{0000/00/00}%
193 <latexrelease>          {\ExpandArgs}{Some pre-expansion commands}%
194 <latexrelease>
195 <latexrelease>\let\UseName\@undefined
196 <latexrelease>\let\ExpandArgs\@undefined
197 <latexrelease>\EndIncludeInRelease

```

\IfExplAtLeastTF A pretty simple wrapper.

`\IfExplAtLeastTF`

```
198 <*2ekernel | latexrelease>
199 <latexrelease>\IncludeInRelease{2023/11/01}%
200 <latexrelease>          {\IfExplAtLeastTF}{Test for expl3 date}%
201 \def\IfExplAtLeastTF{\@ifl@t@r\ExplLoaderFileDate}

(End of definition for \IfExplAtLeastTF.)
    We make sure the command is always available.

202 </2ekernel | latexrelease>
203 <latexrelease>\EndIncludeInRelease
204 <latexrelease>\IncludeInRelease{0000/00/00}%
205 <latexrelease>          {\IfExplAtLeastTF}{Test for expl3 date}%
206 <latexrelease>
207 <latexrelease>\def\IfExplAtLeastTF{\@ifl@t@r\ExplLoaderFileDate}
208 <latexrelease>\EndIncludeInRelease
```

File 06

ltdfn.s.dtx

1 Definitions

This section contains commands used in defining other macros.

```
1 <*2ekernel>
```

1.1 Initex initializations

`\two@digits` Prefix a number less than 10 with ‘0’.

```
2 \def\two@digits#1{\ifnum#1<10 0\fi\number#1}
```

(End of definition for \two@digits.)

`\typeout` Display something on the terminal.

```
3 </2ekernel>
4 <*2ekernel | latexrelease>
5 <latexrelease> \IncludeInRelease{2020/10/01}%
6 <latexrelease>           {\typeout}{Allow "par" in \typeout}%
7 \protected\long\def\typeout#1{\begingroup
8   \set@display@protect
9   \def\par{^^J^^J}%
10  \immediate\write\@unused{#1}\endgroup}
11 </2ekernel | latexrelease>
12 <latexrelease>\EndIncludeInRelease
13 <latexrelease>\IncludeInRelease{0000/00/00}%
14 <latexrelease>           {\typeout}{Allow "par" in \typeout}%
15 <latexrelease>
16 <latexrelease>\def\typeout#1{\begingroup\set@display@protect
17 <latexrelease>   \immediate\write\@unused{#1}\endgroup}
18 <latexrelease>\EndIncludeInRelease
19 <*2ekernel>
```

(End of definition for \typeout.)

`\newlinechar` A char to be used as new-line in output to files.

```
20 \newlinechar‘^^J
```

(End of definition for \newlinechar.)

1.2 Saved versions of TeX primitives

The TeX primitive `\foo` is saved as `\@@foo`. The following primitives are handled in this way:

`\@@par`

```
21 \let\@@par=\par
22 %\let\@@input=\input      %%% moved earlier
23 %\let\@@end=\end         %%%
```

(End of definition for \@@par.)

`\@@hyph` Save original primitive definition.

```
24 \let\@@hyph=\-
```

(End of definition for `\@@hyph`.)

`\@@italiccorr` Save the original italic correction.

```
25 \let\@@italiccorr=\/
```

(End of definition for `\@@italiccorr`.)

`\@height` The following definitions save token space. E.g., using `\@height` instead of `height` saves
`\@depth` 5 tokens at the cost in time of one macro expansion.

```
\@width 26 \def\@height{height} \def\@depth{depth} \def\@width{width}
```

```
\@minus 27 \def\@minus{minus}
```

```
\@plus 28 \def\@plus{plus}
```

The next one is another 100 tokens worth.

```
29 \def\hb@xt@{\hbox to}
```

(End of definition for `\@height` and others.)

```
30 \message{hacks,}
```

`\hb@xt@`

1.3 Command definitions

This section defines the following commands:

```
\@namedef {\NAME}
```

Expands to `\def\NAME`, except name can contain any characters.

```
\@nameuse {\NAME}
```

Expands to `\NAME`.

```
\@ifnextchar X{\YES}{\NO}
```

Expands to `\YES` if next character is an ‘X’, and to `\NO` otherwise. (Uses `\reserved@a-`
`\reserved@c`.) NOTE: GOBBLES ANY SPACE FOLLOWING IT.

```
\@ifstar {\YES}{\NO}
```

Gobbles following spaces and then tests if next the character is a ‘*’. If it is, then it gobbles the ‘*’ and expands to `\YES`, otherwise it expands to `\NO`.

```
\@dblarg {\CMD}{\ARG}
```

Expands to `\{\CMD\}[\ARG]{\ARG}`. Use `\@dblarg\CS` when `\CS` takes arguments `[ARG1]{ARG2}`, where default is `ARG1 = ARG2`.

```
\@ifundefined {\NAME}{\YES}{\NO}
```

: If `\NAME` is undefined then it executes `\YES`, otherwise it executes `\NO`. More precisely, true if `\NAME` either undefined or = `\relax`.

```
\@ifdefinable \NAME{\YES}
```

Executes `\YES` if the user is allowed to define `\NAME`, otherwise it gives an error. The user can define `\NAME` if `\@ifundefined\NAME` is true, ‘`NAME`’ ≠ ‘`relax`’ and the first three letters of ‘`NAME`’ are not ‘`end`’, and if `\endNAME` is not defined.

```
\newcommand *{\FOO}[i]{TEXT}
```

User command to define `\FOO` to be a macro with `i` arguments (`i = 0` if missing) having the definition `\FOO`. Produces an error if `\FOO` already defined.

Normally the command is defined to be `\long` (ie it may take multiple paragraphs in its argument). In the star-form, the command is not defined as `\long` and a blank line in any argument to the command would generate an error.

```
\renewcommand *{\FOO}[i]{TEXT}
```

Same as `\newcommand`, except it checks if `\FOO` already defined.

`\newenvironment` `*{<FOO>}[<i>]{<DEF1>}{<DEF2>}`
equivalent to:
`\newcommand{\FOO}[i]{DEF1} \def{\endFOO}{DEF2}`
(or the appropriate star forms).

`\renewenvironment` Obvious companion to `\newenvironment`.

`\@cons` : See description of `\output` routine.

`\@car` `\@car T1 T2 ... Tn\@nil == T1` (unexpanded)

`\@cdr` `\@cdr T1 T2 ... Tn\@nil == T2 ... Tn` (unexpanded)

`\typeout` `{<message>}`
Produces a warning message on the terminal.

`\typein` `{<message>}`
Types message, asks the user to type in a command, then executes it

`\typein` `[<\CS>]{<MSG>}`
Same as above, except defines `\CS` to be the input instead of executing it.

`\typein`

```

31 \def\typein{%
32   \let\@typein\relax
33   \@testopt\@xtypein\@typein}

34 \ifx\directlua\@undefined

35 \def\@xtypein[#1]#2{%
36   \typeout{#2}%
37   \advance\endlinechar\@M
38   \read\@inputcheck to#1%
39   \advance\endlinechar-\@M
40   \@typein}%

41 \else

42 \def\@xtypein[#1]#2{%
43   \typeout{#2}%
44   \begingroup \endlinechar\m@ne
45   \read\@inputcheck to#1%
46   \expandafter\endgroup
47   \expandafter\def\expandafter#1\expandafter{#1}%
48   \@typein}%

49 \fi

```

(End of definition for \typein.)

`\@namedef`

```

50 \def\@namedef#1{\expandafter\def\csname #1\endcsname}

```

(End of definition for \@namedef.)

`\@nameuse`

```

51 \def\@nameuse#1{\csname #1\endcsname}

```

(End of definition for \@nameuse.)

```

\@cons
52 \def\@cons#1#2{\begingroup\let\@elt\relax\xdef#1{#1\@elt #2}\endgroup}
(End of definition for \@cons.)

\@car
\@cdr
53 \def\@car#1#2\@nil{#1}
54 \def\@cdr#1#2\@nil{#2}
(End of definition for \@car and \@cdr.)

\@carcube \@carcube T1 ... Tn\@nil = T1 T2 T3 , n > 3
55 </2ekernel>
56 <latexrelease>\IncludeInRelease{2020/10/01}{\@carcube}{Make \@carcube long}%
57 <*2ekernel | latexrelease>
58 \long\def\@carcube#1#2#3#4\@nil{#1#2#3}
59 </2ekernel | latexrelease>
60 <latexrelease>\EndIncludeInRelease
61 %
62 <latexrelease>\IncludeInRelease{0000/00/00}{\@carcube}{Undo: Make \@carcube long}%
63 <latexrelease>\def\@carcube#1#2#3#4\@nil{#1#2#3}
64 <latexrelease>\EndIncludeInRelease
65 <*2ekernel>
(End of definition for \@carcube.)

\@onlypreamble This macro adds its argument to the list of commands stored in \@preamblecmds
\@preamblecmds to be disabled after \begin{document}. These commands are redefined to generate
\@notprerr at this point.
66 \def\@preamblecmds{}
67 \def\@onlypreamble#1{%
68   \expandafter\gdef\expandafter\@preamblecmds\expandafter{%
69     \@preamblecmds\do#1}}
70 \@onlypreamble\@onlypreamble
71 \@onlypreamble\@preamblecmds
(End of definition for \@onlypreamble and \@preamblecmds.)

\@star@or@long Look ahead for a *. If present reset \l@ngrel@x so that the next definition, #1, will be
non-long.
72 \def\@star@or@long#1{%
73   \ifstar
74     {\let\l@ngrel@x\relax#1}%
75     {\let\l@ngrel@x\long#1}}
(End of definition for \@star@or@long.)

\l@ngrel@x This is either \relax or \long depending on whether the *-form of a definition command
is being executed.
76 \let\l@ngrel@x\relax
(End of definition for \l@ngrel@x.)

\newcommand User level \newcommand.
77 \def\newcommand{\@star@or@long\new@command}

```

```

\newcommand 78 \def\newcommand#1{%
79 \testopt{\@newcommand#1}0}

(End of definition for \newcommand and \new@command.)

```

```

\@newcommand Handling arguments for \newcommand.
\@argdef 80 \def\@newcommand#1[#2]{%
\@xargdef 81 \kernel@ifnextchar [{\@xargdef#1[#2]}%
82 {\@argdef#1[#2]}}

```

Define #1 if it is definable.

Both here and in \@xargdef the replacement text is absorbed as an argument because if we are not allowed to make the definition we have to get rid of it completely.

```

83 \long\def\@argdef#1[#2]#3{%
84 \ifdefinable #1{\@yargdef#1\@ne-#2}{#3}}

```

Handle the second optional argument.

```

85 \long\def\@xargdef#1[#2] [#3]#4{%
86 \@ifdefinable#1{%

```

Define the actual command to be:

```

\def\foo{\@protected@testopt\foo\foo{default}}

```

where \foo is a cname generated from applying \csname and \string to \foo, ie the actual name contains a backslash and therefore can't clash easily with existing command names. "Default" is the contents of the second optional argument of (re)newcommand.

```

87 \expandafter\def\expandafter#1\expandafter{%
88 \expandafter
89 \@protected@testopt
90 \expandafter
91 #1%
92 \csname\string#1\endcsname
93 {#3}}%

```

Now we define the internal macro ie \foo which is supposed to pick up all arguments (optional and mandatory).

```

94 \expandafter\@yargdef
95 \csname\string#1\endcsname
96 \tw@
97 {#2}%
98 {#4}}

```

(End of definition for \@newcommand, \@argdef, and \@xargdef.)

\@testopt This macro encapsulates the most common call to \@ifnextchar, saving several tokens each time it is used in the definition of a command with an optional argument. #1 The code to execute in the case that there is a [need not be a single token but can be any sequence of commands that 'expects' to be followed by [. If this command were only used in \newcommand definitions then #1 would be a single token and the braces could be omitted from {#1} in the definition below, saving a bit of memory.

```

99 \long\def\@testopt#1#2{%
100 \kernel@ifnextchar[#{#1}-#{1}[#{2]}}

```

(End of definition for \@testopt.)

`\@protected@testopt` Robust version of `\@testopt`. The extra argument (`#1`) must be a single token. If protection is needed the call expands to `\protect` applied to this token, and the 2nd and 3rd arguments are discarded (by `\@x@protect`). Otherwise `\@testopt` is called on the 2nd and 3rd arguments.

This method of making commands robust avoids the need for using up two csnames per command, the price is the extra expansion time for the `\ifx` test.

```

101 \def\@protected@testopt#1{%
102   \ifx\protect\@typeset@protect
103     \expandafter\@testopt
104   \else
105     \@x@protect#1%
106   \fi}

```

(End of definition for \@protected@testopt.)

`\@yargdef` These generate a primitive argument specification, from a L^AT_EX [`<digit>`] form; in fact `<digit>` can be anything such that `\number <digit>` is single digit.

`\@yargd@f` Reorganised slightly so that `\renewcommand{\reserved@a}[1]{foo}` works. I am not sure this is worth it, as a following `\newcommand` would over-write the definition of `\reserved@a`.

Recall that L^AT_EX2.09 goes into an infinite loop with `\renewcommand[1]{\@tempa}{foo}` (DPC 6 October 93).

Reorganised again (DPC 1999). Rather than make a loop to construct the argument spec by counting, just extract the required argument spec by using a delimited argument (delimited by the digit). This is faster and uses less tokens. The coding is slightly odd to preserve the old interface (using `#2 = \tw@` as the flag to surround the first argument with []). But the new method did not allow for the number of arguments `#3` not being given as an explicit digit; hence (further expansion of this argument and use of) `\number` was added later in 1999.

It is not clear why these are still `\long`.

```

107 \long \def \@yargdef #1#2#3{%
108   \ifx#2\tw@
109     \def\reserved@b##11{####1}%
110   \else
111     \let\reserved@b@gobble
112   \fi
113   \expandafter
114     \@yargd@f \expandafter{\number #3}#1%
115 }

116 \long \def \@yargd@f#1#2{%
117   \def \reserved@a ##1#1##2##{%
118     \expandafter\def\expandafter#2\reserved@b ##1#1%
119   }%
120   \l@ngrel@x \reserved@a 0##1##2##3##4##5##6##7##8##9###1%
121 }

```

(End of definition for \@yargdef and \@yargd@f.)

`\@reargdef`

```

122 \long\def\@reargdef#1[#2]{%
123   \@yargdef#1\@ne{#2}}

```

(End of definition for \@reargdef.)

\renewcommand Check the command name is already used. If not give an error message. Then temporarily disable \@ifdefinable then call \newcommand. (Previous version \let#1=\relax but this does not work too well if #1 is \@tempa-e.)

```
124 \def\renewcommand{\@star@or@long\renew@command}
```

```
\renew@command 125 \def\renew@command#1{%
126   \begingroup \escapechar\m@ne\xdef\@gtempa{\string#1}\endgroup
127   \expandafter\ifundefined\@gtempa
128     {\@latex@error{Command \string#1 undefined}\@ehc}%
129     \relax
130   \let\@ifdefinable\@rc@ifdefinable
131   \new@command#1}
```

(End of definition for \renewcommand and \renew@command.)

\@ifdefinable Test if user is allowed to define a command.

```
\@ifdefinable 132 \long\def\@ifdefinable #1#2{%
\@ifdefinable 133   \edef\reserved@a{\expandafter\@gobble\string #1}%
\@rc@ifdefinable 134   \@ifundefined\reserved@a
135     {\edef\reserved@b{\expandafter\@carcube \reserved@a xxx\@nil}%
136     \ifx \reserved@b\@qend \@notdefinable\else
137     \ifx \reserved@a\@qrelax \@notdefinable\else
138     #2%
139     \fi
140     \fi}%
141   \@notdefinable}
```

Saved definition of \@ifdefinable.

```
142 \let\@ifdefinable\@ifdefinable
```

Version of \@ifdefinable for use with \renewcommand. Does not do the check this time, but restores the normal definition.

```
143 \long\def\@rc@ifdefinable#1#2{%
144   \let\@ifdefinable\@ifdefinable
145   #2}
```

(End of definition for \@ifdefinable, \@ifdefinable, and \@rc@ifdefinable.)

\newenvironment Define a new user environment. #1 is the environment name. #2# Grabs all the tokens up to the first {. These will be any optional arguments. They are not parsed at this point, but are just passed to \@newenv which will eventually call \newcommand. Any optional arguments will then be parsed by \newcommand as it defines the command that executes the 'begin code' of the environment.

This #2# trick removed with version 1.2i as it fails if a { occurs in the optional argument. Now use \@ifnextchar directly.

```
146 \def\newenvironment{\@star@or@long\new@environment}
```

```
\new@environment 147 \def\new@environment#1{%
148   \@testopt{\@newenva#1}0}
```



```

149 \def\@newenva#1[#2]{%
\@newenva 150   \kernel@ifnextchar [{\@newenvb#1[#2]}{\@newenv{#1}{[#2]}}}]

151 \def\@newenvb#1[#2][#3]{\@newenv{#1}{[#2][[#3]]}}
\@newenvb (End of definition for \newenvironment and others.)

\renewenvironment Redefine an environment. For \renewenvironment disable \@ifdefinable and then call
\newenvironment. It is OK to \let the argument to \relax here as there should not
be a @temp... environment.
152 \def\renewenvironment{\@star@or@long\renew@environment}

\renew@environment 153 \def\renew@environment#1{%
154   \@ifundefined{#1}%
155     {\@latex@error{Environment #1 undefined}\@ehc
156     }\relax
157   \expandafter\let\csname#1\endcsname\relax
158   \expandafter\let\csname end#1\endcsname\relax
159   \new@environment{#1}}
(End of definition for \renewenvironment and \renew@environment.)

\@newenv The internal version of \newenvironment.
Call \newcommand to define the begin-code for the environment. \def is used for
the end-code as it does not take arguments. (but may contain \pars)
Make sure that an attempt to define a ‘graf’ or ‘group’ environment fails by temporarily
letting the undefined \dots (begin code) to the definition of \end\dots and as a
result we get an error if that has a definition.
160 \long\def\@newenv#1#2#3#4{%
161   \@ifundefined{#1}%
162     {\expandafter\let\csname#1\expandafter\endcsname
163       \csname end#1\endcsname}%
164     \relax
165   \expandafter\new@command
166     \csname #1\endcsname#2{#3}%
167     \l@ngrel@x\expandafter\def\csname end#1\endcsname{#4}}
(End of definition for \@newenv.)

\newif And here’s a different sort of allocation: For example, \newif\iffoo creates \footrue,
\foofalse to go with \iffoo.
168 \def\newif#1{%
169   \count@\escapechar \escapechar@m@ne
170   \let#1\iffalse
171   \@if#1\iftrue
172   \@if#1\iffalse
173   \escapechar\count@}

```

```

\@if 174 \def\@if#1#2{%
175   \expandafter\def\csname\expandafter@gobbletwo\string#1%
176           \expandafter@gobbletwo\string#2\endcsname
177           {\let#1#2}}

```

(End of definition for \newif and \@if.)

`\providecommand` `\providecommand` takes the same arguments as `\newcommand`, but discards them if #1 is already defined, Otherwise it just acts like `\newcommand`. This implementation currently leaves any discarded definition in `\reserved@a` (and possibly `\reserved@a`) this wastes a bit of space, but it will be reclaimed as soon as these scratch macros are redefined.

```
178 \def\providecommand{\@star@or@long\provide@command}
```

```

\provide@command 179 \def\provide@command#1{%
180   \begingroup
181   \escapechar\m@ne\xdef\@gtempa{\string#1}%
182   \endgroup
183   \expandafter\ifundefined\@gtempa
184     {\def\reserved@a{\new@command#1}}%
185     {\def\reserved@a{\renew@command\reserved@a}}%
186   \reserved@a}%

```

(End of definition for \providecommand and \provide@command.)

`\CheckCommand` `\CheckCommand` takes the same arguments as `\newcommand`. If the command already exists, with the same definition, then nothing happens, otherwise a warning is issued. Useful for checking the current state before a macro package starts redefining things. Currently two macros are considered to have the same definition if they are the same except for different default arguments. That is, if the old definition was: `\newcommand\xxx[2][a]{(#1)(#2)}` then `\CheckCommand\xxx[2][b]{(#1)(#2)}` would *not* generate a warning, but, for instance `\CheckCommand\xxx[2]{(#1)(#2)}` would.

```
187 \def\CheckCommand{\@star@or@long\check@command}
```

`\CheckCommand` is only available in the preamble part of the document.

```
188 \@onlypreamble\CheckCommand
```

```

\check@command 189 \def\check@command#1#2#\@check@c#1{#2}
190 \@onlypreamble\check@command

```

(End of definition for \CheckCommand and \check@command.)

`\@check@c` `\CheckCommand` itself just grabs all the arguments we need, without actually looking for [optional argument forms. Now define `\reserved@a`. If `\reserved@a` is then defined, compare it with the “\#1’ otherwise compare `\reserved@a` with #1.

```

191 \long\def\@check@c#1#2#3{%
192   \expandafter\let\csname\string\reserved@a\endcsname\relax
193   \renew@command\reserved@a#2{#3}%
194   \@ifundefined{\string\reserved@a}%
195   {\@check@eq#1\reserved@a}%

```

```

196   {\expandafter\@check@eq
197     \csname\string#1\expandafter\endcsname
198     \csname\string\reserved@a\endcsname}}
199 \@onlypreamble\@check@c

```

(End of definition for \@check@c.)

\@check@eq Complain if #1 and #2 are not \ifx equal.

```

200 \def\@check@eq#1#2{%
201   \ifx#1#2\else
202     \latex@warning@no@line
203       {Command \noexpand#1 has
204         changed.\MessageBreak
205         Check if current package is valid}%
206   \fi}
207 \@onlypreamble\@check@eq

```

(End of definition for \@check@eq.)

\@gobble The \@gobble macro is used to get rid of its argument.

```

\@gobbletwo 208 \long\def \@gobble #1{}
\@gobblethree 209 \long\def \@gobbletwo #1#2{}
\@gobblefour 210 \long\def \@gobblethree #1#2#3{}
211 \long\def \@gobblefour #1#2#3#4{}

```

(End of definition for \@gobble and others.)

There are also \@gobble@om, \@gobble@som, \@gobble@with@sphack@om, and \@gobble@with@sphack@som. They accept an optional and a mandatory argument, possibly preceded by a star. In all cases the expansion is empty or just manipulates the spaces around the command. Used to disable commands such as \index or \label in certain situations. Since they are defined with \DeclareDocumentCommand, which is not yet available at this point, the actual definition happens in ltsect.dtx.

\@firstofone Some argument-grabbers.

```

\@firstoftwo 212 \long\def \@firstofone#1{#1}
\@secondoftwo 213 \long\def \@firstoftwo#1#2{#1}
214 \long\def \@secondoftwo#1#2{#2}

```

\@iden is another name for \@firstofone for compatibility reasons.

```

215 \let \@iden \@firstofone

```

\@iden (End of definition for \@firstofone and others.)

\@thirdofthree Another grabber now used in the encoding specific section.

```

216 \long\def \@thirdofthree#1#2#3{#3}

```

(End of definition for \@thirdofthree.)

\@expandtwoargs A macro to totally expand two arguments to another macro

```

217 \if2kernel)
218 \iflatexrelease\IncludeInRelease{2022/11/01}%
219 \iflatexrelease \def \@expandtwoargs#1#2#3{\protected@edef}%
220 \if*2kernel | \iflatexrelease
221 \def \@expandtwoargs#1#2#3{%
222 \protected@edef \reserved@a{\noexpand#1{#2}{#3}}\reserved@a}

```

```

223 </2ekernel | latexrelease>
224 <latexrelease>\EndIncludeInRelease
225 <latexrelease>\IncludeInRelease{00/00/00}%
226 <latexrelease>          {\@expandtwoargs}{protected edef}%
227 <latexrelease>\def\@expandtwoargs#1#2#3{%
228 <latexrelease>\edef\reserved@a{\noexpand#1{#2}{#3}}\reserved@a}
229 <latexrelease>\EndIncludeInRelease
230 <*2ekernel>

```

(End of definition for \@expandtwoargs.)

`\@backslashchar` A category code 12 backslash.

```

231 \edef\@backslashchar{\expandafter\@gobble\string\}

```

(End of definition for \@backslashchar.)

1.4 Robust commands and protect

Fragile and robust commands are one of the thornier issues in L^AT_EX's commands. Whilst typesetting documents, L^AT_EX makes use of many of T_EX's features, such as arithmetic, defining macros, and setting variables. However, there are (at least) three different occasions when these commands are not safe. These are called 'moving arguments' by L^AT_EX, and consist of:

- writing information to a file, such as indexes or tables of contents.
- writing information to the screen.
- inside an `\edef`, `\message`, `\mark`, or other command which evaluates its argument fully.

The method L^AT_EX uses for making fragile commands robust is to precede them with `\protect`. This can have one of four possible values:

- `\relax`, for normal typesetting. So `\protect\foo` will execute `\foo`.
- `\string`, for writing to the screen. So `\protect\foo` will write `\foo`.
- `\noexpand`, for writing to a file. So `\protect\foo` will write `\foo` followed by a space.
- `\@unexpandable@protect`, for writing a moving argument to a file. So `\protect\foo` will write `\protect\foo` followed by a space. This value is also used inside `\edefs`, `\marks` and other commands which evaluate their arguments fully. More precisely, whenever the content of an `\edef` or `\xdef` etc. can contain arbitrary user input not under the direct control of the programmer, one should use `\proctected@edef` instead of `\edef`, etc., so that `\protect` has a suitable definition and the user input will not break if it contains fragile commands.

`\@unexpandable@protect`

```

232 \def\@unexpandable@protect{\noexpand\protect\noexpand}

```

(End of definition for \@unexpandable@protect.)

`\DeclareRobustCommand` This is a package-writers command, which has the same syntax as `\newcommand`, but
`\declare@robustcommand` which declares a protected command. It does this by having

```
\DeclareRobustCommand\foo
define \foo to be \protect\foo<space>,
and then use \newcommand\foo<space>.
Since the internal command is \foo<space>, when it is written to an auxiliary file, it
will appear as \foo.
```

We have to be a bit cleverer if we're defining a short command, such as `_`, in order to make sure that the auxiliary file does not include a space after the command, since `_ a` and `_a` aren't the same. In this case we define `_` to be:

```
\x@protect\_ \protect\_<space>
```

which expands to:

```
\ifx\protect\@typeset@protect\else
  \x@protect@\_
\fi
\protect\_<space>
```

Then if `\protect` is `\@typeset@protect` (normally `\relax`) then we just perform `_<space>`, and otherwise `\x@protect@` gobbles everything up and expands to `\protect_.`

Note: setting `\protect` to any value other than `\relax` whilst in 'typesetting' mode will cause commands to go into an infinite loop! In particular, setting `\protect` to `\@empty` will cause `_` to loop forever. It will also break lots of other things, such as protected `\ifmmodes` inside `\haligns`. If you really have to do such a thing, then please set `\@typeset@protect` to be `\@empty` as well. (This is what the code for `\patterns` does, for example.)

More fun with `\expandafter` and `\csname`.

```
233 \def\DeclareRobustCommand{\@star@or@long\declare@robustcommand}
234 \def\declare@robustcommand#1{%
235   \ifx#1\@undefined\else\ifx#1\relax\else
236     \@latex@info{Redefining \string#1}%
237   \fi\fi
238   \edef\reserved@a{\string#1}%
239   \def\reserved@b{#1}%
240   \edef\reserved@b{\expandafter\strip@prefix\meaning\reserved@b}%
241   \edef#1{%
242     \ifx\reserved@a\reserved@b
243       \noexpand\x@protect
244       \noexpand#1%
245     \fi
246     \noexpand\protect
247     \expandafter\noexpand\csname
248       \expandafter\@gobble\string#1 \endcsname
249   }%
250   \let\@ifdefinable\rc@ifdefinable
251   \expandafter\new@command\csname
252     \expandafter\@gobble\string#1 \endcsname
253 }
```

(End of definition for `\DeclareRobustCommand` and `\declare@robustcommand`.)

```

\@x@protect
\@x@protect 254 \def\@x@protect#1{%
255   \ifx\protect\@typeset@protect\else
256     \@x@protect#1%
257   \fi
258 }
259 \def\@x@protect#1\fi#2#3{%
260   \fi\protect#1%
261 }

```

(End of definition for \@x@protect and \x@protect.)

`\@typeset@protect` We set `\@typeset@protect` to `\relax` rather than `\@empty` to make sure that the protection mechanism stops the look-ahead and expansion performed at the start of `\halign` cells.

```
262 \let\@typeset@protect\relax
```

(End of definition for \@typeset@protect.)

`\set@display@protect` These macros set `\protect` appropriately for typesetting or displaying.

```

\set@typeset@protect 263 \def\set@display@protect{\let\protect\string}
264 \def\set@typeset@protect{\let\protect\@typeset@protect}

```

(End of definition for \set@display@protect and \set@typeset@protect.)

`\protected@edef` The commands `\protected@edef` and `\protected@xdef` perform ‘safe’ `\edefs` and `\xdefs`, saving and restoring `\protect` appropriately. For cases where restoring `\protect` doesn’t matter, there’s an ‘unsafe’ `\unrestored@protected@xdef`, useful if you know what you’re doing!

`\protected@xdef`
`\unrestored@protected@xdef`
`\restore@protect`

```

265 \def\protected@edef{%
266   \let\@@protect\protect
267   \let\protect\@unexpandable@protect
268   \afterassignment\restore@protect
269   \edef
270 }
271 \def\protected@xdef{%
272   \let\@@protect\protect
273   \let\protect\@unexpandable@protect
274   \afterassignment\restore@protect
275   \xdef
276 }
277 \def\unrestored@protected@xdef{%
278   \let\protect\@unexpandable@protect
279   \xdef
280 }
281 \def\restore@protect{\let\protect\@@protect}

```

(End of definition for \protected@edef and others.)

`\protect` The normal meaning of `\protect`

```
282 \set@typeset@protect
```

(End of definition for \protect.)

`\MakeRobust` This macro makes an existing fragile macro robust, but only if it hasn't been robust in the past, i.e., it checks for the existence of the macro `\<name>` and if that exists it assumes that `\<name>` is already robust. In that case either undefine the inner macro first or use `\DeclareRobustCommand` to define it in a robust way directly. We could probably test the top-level definition to have the right kind of structure, but this is somewhat problematical as we then have to distinguish between `\long` macros and others and also take into account that sometimes the top-level is deliberately done manually (like with `\begin`).

The macro firstly checks if the control sequence in question exists at all.

```

283 </2kernel>
284 <latexrelease>\IncludeInRelease{2020/10/01}{\MakeRobust}{\MakeRobust}%
285 <*2kernel | latexrelease>
286 \def\MakeRobust#1{%
287   \count@=\escapechar
288   \escapechar='\'
289   \@ifundefined{\expandafter@gobble\string#1}{%
290     \@latex@error{Command '\string#1' undefined.%
291       \MessageBreak There is nothing here to make robust}%
292     \@eha
293   }%

```

Then we check if the macro is already robust. We do this by testing if the internal name for a robust macro is defined, namely `\foo`. If it is already defined do nothing, otherwise set `\foo` equal to `\foo` and redefine `\foo` so that it acts like a macro defined with `\DeclareRobustCommand`. We use `\@kernel@rename@newcommand` to copy `\foo` over to `\foo`, including a possible default optional argument.

```

294   {%
295     \@ifundefined{\expandafter@gobble\string#1\space}%
296     {%
297       \expandafter\@kernel@rename@newcommand
298       \csname\expandafter@gobble\string#1\space\endcsname
299       #1%
300       \edef\reserved@a{\string#1}%
301       \def\reserved@b{#1}%
302       \edef\reserved@b{\expandafter\strip@prefix\meaning\reserved@b}%
303       \xdef#1{%
304         \ifx\reserved@a\reserved@b
305           \noexpand\x@protect\noexpand#1%
306         \fi
307         \noexpand\protect\expandafter\noexpand
308         \csname\expandafter@gobble\string#1\space\endcsname}%
309       }%
310     {\@latex@info{Command '\string#1' is already robust}}%
311   }%
312   \escapechar=\count@
313 }%

```

`\@kernel@rename@newcommand` This macro renames a command, possibly with an optional argument (defined with `\newcommand`) from #2 to #1, by renaming the internal macro `\#2` to `\#1` and defining `\#1` appropriately, then undefining `\#2` and `\#2`. The `\afterassignment` trick is to make both definitions in `\@copy@newcommand` global (which are local by default).

In case the macro was defined with `\newcommand` and an optional argument, to replicate exactly the behaviour of `\DeclareRobustCommand` we have to move also

the internal `\\foo` to `\\foo␣`. In that case, #1 will be a parameterless macro (`\robust@command@chk@safe` checks that), and `\@if@newcommand` will return true (both defined below in this file). If so, we can use `\@copy@newcommand` rather than plain `\let` to copy the command over. `\@kernel@rename@newcommand` does this test and carries out the renaming.

```

314 \def\@kernel@rename@newcommand#1#2{%
315   \robust@command@chk@safe#2%
316   {\@if@newcommand#2%
317     {\afterassignment\global
318       \global\@copy@newcommand#1#2%
319       \global\let#2\@undefined
320       \global\expandafter\let\csname\string#2\endcsname\@undefined}%
321     {\global\let#1=#2}}%
322   {\global\let#1=#2}}

323 </2kernel | latexrelease>
324 <latexrelease>\EndIncludeInRelease
325 %
326 <latexrelease>\IncludeInRelease{2019/10/01}{\MakeRobust}{\MakeRobust}%
327 <latexrelease>\def\MakeRobust#1{%
328 <latexrelease>  \ifundefined{\expandafter\@gobble\string#1}{%
329 <latexrelease>    \latex@error{The control sequence ‘\string#1’ is undefined!%
330 <latexrelease>      \MessageBreak There is nothing here to make robust}%
331 <latexrelease>    \@eha
332 <latexrelease>  }%
333 <latexrelease>  {%
334 <latexrelease>    \ifundefined{\expandafter\@gobble\string#1\space}%
335 <latexrelease>    {%
336 <latexrelease>      \global\expandafter\let\csname
337 <latexrelease>        \expandafter\@gobble\string#1\space\endcsname=#1%
338 <latexrelease>      \edef\reserved@a{\string#1}%
339 <latexrelease>      \def\reserved@b{#1}%
340 <latexrelease>      \edef\reserved@b{\expandafter\strip@prefix\meaning\reserved@b}%
341 <latexrelease>      \xdef#1{%
342 <latexrelease>        \ifx\reserved@a\reserved@b
343 <latexrelease>          \noexpand\x@protect\noexpand#1%
344 <latexrelease>        \fi
345 <latexrelease>        \noexpand\protect\expandafter\noexpand
346 <latexrelease>        \csname\expandafter\@gobble\string#1\space\endcsname}%
347 <latexrelease>      }%
348 <latexrelease>      {\@latex@info{The control sequence ‘\string#1’ is already robust}}%
349 <latexrelease>    }%
350 <latexrelease>  }%
351 <latexrelease>\let\@kernel@rename@newcommand\@undefined
352 <latexrelease>\EndIncludeInRelease
353 %
354 <latexrelease>\IncludeInRelease{2015/01/01}{\MakeRobust}{\MakeRobust}%
355 <latexrelease>\def\MakeRobust#1{%
356 <latexrelease>  \ifundefined{\expandafter\@gobble\string#1}{%
357 <latexrelease>    \latex@error{The control sequence ‘\string#1’ is undefined!%
358 <latexrelease>      \MessageBreak There is nothing here to make robust}%
359 <latexrelease>    \@eha
360 <latexrelease>  }%
361 <latexrelease>  {%

```



```

362 <latexrelease> \@ifundefined{\expandafter@gobble\string#1\space}%
363 <latexrelease> {%
364 <latexrelease> \expandafter\let\csname
365 <latexrelease> \expandafter@gobble\string#1\space\endcsname=#1%
366 <latexrelease> \edef\reserved@a{\string#1}%
367 <latexrelease> \def\reserved@b{#1}%
368 <latexrelease> \edef\reserved@b{\expandafter\strip@prefix\meaning\reserved@b}%
369 <latexrelease> \edef#1{%
370 <latexrelease> \ifx\reserved@a\reserved@b
371 <latexrelease> \noexpand\x@protect\noexpand#1%
372 <latexrelease> \fi
373 <latexrelease> \noexpand\protect\expandafter\noexpand
374 <latexrelease> \csname\expandafter@gobble\string#1\space\endcsname}%
375 <latexrelease> }%
376 <latexrelease> {\@latex@info{The control sequence ‘\string#1’ is already robust}}%
377 <latexrelease> }%
378 <latexrelease> }%
379 <latexrelease> \let\@kernel@rename@newcommand\@undefined
380 <latexrelease> \EndIncludeInRelease
381 %
382 <latexrelease> \IncludeInRelease{0000/00/00}{\MakeRobust}{\MakeRobust}%
383 <latexrelease> \let\MakeRobust\@undefined
384 <latexrelease> \let\@kernel@rename@newcommand\@undefined
385 <latexrelease> \EndIncludeInRelease
386 <*2ekernel)

```

(End of definition for \MakeRobust and \@kernel@rename@newcommand.)

`\kernel@make@fragile` The opposite of `\MakeRobust` except that it doesn't do many checks as it is internal to the kernel. Why does one want such a thing? Only for compatibility reasons if `latexrelease` requests a rollback of the kernel. For this reason we pretend that this command existed in all earlier versions of L^AT_EX i.e., we are not rolling it back since we need it precisely then. But we have to get it into the `latexrelease` file so that a roll forward is possible too.

```

387 </2ekernel)
388 <*2ekernel | latexrelease)
389 <latexrelease> \IncludeInRelease{2020/10/01}%
390 <latexrelease> \kernel@make@fragile{Undo robustness}%
391 \def\kernel@make@fragile#1{%
392 \ifundefined{\expandafter@gobble\string#1\space}%

```

If not robust do nothing.

```

393 }%

```

Otherwise copy `\foo_` back to `\foo`. Then use `\@kernel@rename@newcommand` to check and copy `\\foo_` back to `\\foo` in case the command has an optional argument. If so, also undefine `\\foo_`, and at the end undefine `\foo_`.

```

394 {%
395 \global\expandafter\let\expandafter #1\csname
396 \expandafter@gobble\string#1\space\endcsname
397 \expandafter\@kernel@rename@newcommand
398 \csname\expandafter@gobble\string#1\space\endcsname
399 \csname\expandafter@gobble\string#1\space\endcsname
400 \global\expandafter\let\csname

```

```

401     \expandafter\@gobble\string#1\space\endcsname\@undefined
402   }%
403 }
404 \langle latexrelease \rangle \EndIncludeInRelease
405 %
406 \langle latexrelease \rangle \IncludeInRelease{0000/00/00}%
407 \langle latexrelease \rangle      {\kernel@make@fragile}{Undo robustness}%
408 \langle latexrelease \rangle \def\kernel@make@fragile#1{%
409 \langle latexrelease \rangle  \@ifundefined{\expandafter\@gobble\string#1\space}%
410 \langle latexrelease \rangle    {}%
411 \langle latexrelease \rangle    {%
412 \langle latexrelease \rangle      \global\expandafter\let\expandafter #1\csname
413 \langle latexrelease \rangle      \expandafter\@gobble\string#1\space\endcsname
414 \langle latexrelease \rangle      \global\expandafter\let\csname
415 \langle latexrelease \rangle      \expandafter\@gobble\string#1\space\endcsname\@undefined
416 \langle latexrelease \rangle    }%
417 \langle latexrelease \rangle }
418 \langle latexrelease \rangle \EndIncludeInRelease
419 \langle /2ekernel | latexrelease \rangle
420 \langle *2ekernel \rangle

```

(End of definition for \kernel@make@fragile.)

1.5 Acting on robust commands

```

421 \langle /2ekernel \rangle
422 \langle latexrelease \rangle \IncludeInRelease{2020-10-01}{\robust@command@act}
423 \langle latexrelease \rangle {Add \robust@command@act}%
424 \langle *2ekernel | latexrelease \rangle

```

With most document level commands being robust now there is more of a requirement to have a standard way of aliasing (or copying) a command to a new name, for example to save an original definition before changing a command. `\DeclareCommandCopy` is analogous to \TeX 's `\let`, except that it copes with the different types of robust commands defined by \LaTeX 's mechanisms.

A couple of “types of robustness” are defined by the $\text{\LaTeX} 2_\epsilon$ kernel, namely robust commands defined with `\DeclareRobustCommand` and commands with optional arguments defined with `\newcommand`. However there are other types of robust commands that are frequently used, which are not defined in the $\text{\LaTeX} 2_\epsilon$ kernel, like commands defined with `xparse`'s `\NewDocumentCommand` and `etoolbox`'s `\newrobustcmd`.

In this section we will define a generic extensible machinery to act on robust commands. This code will then be used to test if a command is robust, considered the different types of robustness, and then either copy that definition, if `\DeclareCommandCopy` (or similar) is used, or show the definition of the command, if `\ShowCommand` is used.

`\robust@command@act` The looping machinery is generic and knows nothing about what is to be done for each case. The syntax of the main macro `\robust@command@act` is:

```

\robust@command@act\langle action-list \rangle\langle robust-cmd \rangle
\langle fallback-action \rangle\langle act-arg \rangle

```

`\langle action-list \rangle` is a token list of the form:

```

{<if-type-1> <act-type-1>}
{<if-type-2> <act-type-2>}
...

```

`\robust@command@act` will iterate over the `<action-list>`, evaluating each `<if-type-n>` `<robust-cmd>` `{<true>}{<false>}`. If the `<if-type-n>` conditional returns `<true>`, then `<act-type-n>``<act-arg>` is executed, and the loop ends. If the conditional returns `<false>`, then `<if-type-n + 1>` is executed in the same way, until either one of the conditionals return `<true>`, or the end of the `<action-list>` is reached. If the end is reached, then `<fallback-action>``<act-arg>` is executed before `\robust@command@act` exits.

`\robust@command@act` will start by using `\robust@command@act@chk@args` to check if the `<robust-cmd>` (`#2`) is a parameterless (possibly `\protected`) macro. If it is not, the command is not a robust command: these always start with a parameterless user-level macro; in that case, `\robust@command@act@end` is used to short-circuit the process and do the `<fallback-action>` (`#3`). This first test is necessary because later on we need to be able to expand the `<robust-cmd>` without the risk of it Breaking Badly, and as a bonus, this speeds up the process in case we used `\NewCommandCopy` in a “normal” macro.

```

425 \long\def\robust@command@act#1#2#3#4{%
426   \robust@command@chk@safe#2%
427   {\expandafter\robust@command@act@loop
428     \expandafter#2%
429     #1{\@nnil\@nnil}%
430   \robust@command@act@end}%
431   {\robust@command@act@end}%
432   {#3}{#4}}%

```

If `\robust@command@act@chk@args` branched to false, then `\robust@command@act@loop` will loop over the list of items in the `<action-list>` (`#1`), and process each item as described earlier. If the `<if-type-n>` command expands to `<true>` then `\robust@command@act@do` is used to execute `<act-type-n>` on the `<act-arg>`, otherwise the loop resumes with the next item.

```

433 \long\def\robust@command@act@loop#1#2{\robust@command@act@loop@aux#1#2}
434 \long\def\robust@command@act@loop@aux#1#2#3{%
435   \ifx\@nnil#2%
436   \else
437     #2{#1}%
438     {\robust@command@act@do{#3}}%
439     {\expandafter\robust@command@act@loop\expandafter#1}%
440   \fi}
441 \long\def\robust@command@act@do#1%
442   \fi#2%
443   \robust@command@act@end#3#4{%
444   \fi
445   #1#4}

```

If the end is reached and no action was taken, then do `<fallback-action>``<act-arg>`.

```

\robust@command@act@end 446 \long\def\robust@command@act@end#1#2{#1#2}

```

```

\robust@command@chk@safe 447 \long\def\robust@command@chk@safe#1{%
\robust@command@act@chk@args 448   \begingroup

```

```

449   \escapechar='\\
450   \expandafter\endgroup\expandafter
451   \robust@command@act@chk@args\meaning#1:->\@nil}
452 \def\robust@command@act@chk@args#1:->#2\@nil{%
453   \expl@str@if@eq@nnTF{#1}{macro}%
454   {\@firstoftwo}%
455   {\expl@str@if@eq@nnTF{#1}{\protected macro}%
456   {\@firstoftwo}%
457   {\@secondoftwo}}

458 </2ekernel | latexrelease>
459 <latexrelease>\EndIncludeInRelease
460 <latexrelease>\IncludeInRelease{0000-00-00}{\robust@command@act}
461 <latexrelease> {Add \robust@command@act}%
462 <latexrelease>\let\robust@command@act\undefined
463 <latexrelease>\let\robust@command@act@loop\undefined
464 <latexrelease>\let\robust@command@act@loop@aux\undefined
465 <latexrelease>\let\robust@command@act@do\undefined
466 <latexrelease>\let\robust@command@act@end\undefined
467 <latexrelease>\let\robust@command@chk@safe\undefined
468 <latexrelease>\let\robust@command@act@chk@args\undefined
469 <latexrelease>\EndIncludeInRelease
470 <*2ekernel>

```

(End of definition for `\robust@command@act` and others.)

1.5.1 Copying robust commands

```

471 </2ekernel>
472 <latexrelease>\IncludeInRelease{2020-10-01}{\DeclareCommandCopy}
473 <latexrelease> {Add \NewCommandCopy, \RenewCommandCopy, and \DeclareCommandCopy}%
474 <*2ekernel | latexrelease>

```

`\NewCommandCopy` `\NewCommandCopy` starts by checking if `#1` is already defined, and raises an error if so, otherwise the definition is carried out. `\RenewCommandCopy` does (almost) the opposite. `\DeclareCommandCopy` If the command is *not* defined, then an error is raised. But the definition is carried out anyhow, so the behaviour is consistent with `\renewcommand`.

A `\ProvideCommandCopy` isn't defined because it's not reasonably useful. `\provide...` commands mean “define this if there's no other definition”, but copying a command (usually) implies that the command being copied is defined, so `\ProvideCommandCopy` doesn't make a lot of sense. But more importantly, the most common use case of copying a command is to redefine it later, while preserving the old definition, as in:

```

\ProvideCommandCopy \A \B
\renewcommand \B { ... \A ... }

```

then, if `\A` is already defined the first line is skipped, and in this case `\B` won't work as expected.

The three versions call the internal `\declare@commandcopy` with the proper action. `\@firstofone` will carry out the copy. The only case when the copy is not made is the `<false>` case for `\NewCommandCopy`, in which the command already exists and the definition is aborted.

```

475 \def\NewCommandCopy{%
476   \declare@commandcopy

```

```

477     {\@firstofone}%
478     {\@firstoftwo\@notdefinable}}
479 \def\RenewCommandCopy{%
480   \declare@commandcopy
481     {\@latex@error{Command \@backslashchar\reserved@a\space undefined}\@ehc
482     \@firstofone}%
483     {\@firstofone}}
484 \def\DeclareCommandCopy{%
485   \declare@commandcopy
486     {\@firstofone}%
487     {\@firstofone}}

```

Start by checking if the command is already defined. The proper action is taken by each specific command above. If all's good, then `\robust@command@act` is called with the proper arguments as described earlier, with `\@declarecommandcopylisthook` as the *⟨action-list⟩* and `\declarecommandcopy@let` as the *⟨fallback-action⟩*.

```

\declare@commandcopy
\declare@commandcopy@do
488 \long\def\declare@commandcopy#1#2#3#4{%
489   \edef\reserved@a{\@expl@cs@to@str@N#3}%
490   \@ifundefined\reserved@a{#1}{#2}%
491     {\declare@commandcopy@do{#3}{#4}}
492 \long\def\declare@commandcopy@do#1#2{%
493   \robust@command@act
494     \@declarecommandcopylisthook#2%
495     \declare@commandcopy@let{#1#2}}

```

The initial definition of `\@declarecommandcopylisthook` contains the tests for the two types of robust command in the kernel.

```

\@declarecommandcopylisthook
496 \def\@declarecommandcopylisthook{%
497   {\@if@DeclareRobustCommand \@copy@DeclareRobustCommand}%
498   {\@if@newcommand \@copy@newcommand}}

```

The initial definition of `\@declarecommandcopylisthook` contains the tests for the two types of robust command in the kernel.

```

\declare@commandcopy@let
499 \long\def\declare@commandcopy@let#1#2{\let#1=#2\relax}

```

Now the rollback code.

```

500 </2ekernel | latexrelease>
501 <latexrelease>\EndIncludeInRelease
502 <latexrelease>\IncludeInRelease{0000-00-00}{\DeclareCommandCopy}
503 <latexrelease> {Undefine \NewCommandCopy, \RenewCommandCopy, and \DeclareCommandCopy}%
504 <latexrelease>\let\NewCommandCopy\@undefined
505 <latexrelease>\let\RenewCommandCopy\@undefined
506 <latexrelease>\let\DeclareCommandCopy\@undefined
507 <latexrelease>\let\declare@commandcopy\@undefined
508 <latexrelease>\let\@declarecommandcopylisthook\@undefined
509 <latexrelease>\let\declare@commandcopy@let\@undefined
510 <latexrelease>\EndIncludeInRelease
511 <*2ekernel>

```

(End of definition for \NewCommandCopy and others.)

```

512 </2ekernel>
513 <latexrelease>\IncludeInRelease{2023-06-01}{\DeclareEnvironmentCopy}
514 <latexrelease> {Add \NewEnvironmentCopy, \RenewEnvironmentCopy, and \DeclareEnvironmentCopy}%
515 <*2ekernel | latexrelease>

```

`\NewEnvironmentCopy` If `\#1` or `\end#1` already exist one gets an error message talking about the problematical
`\RenewEnvironmentCopy` command (not the environment). The remainder of the L^AT_EX run is probably badly
`\DeclareEnvironmentCopy` broken and it is unlikely that continuing it gives reasonable results.

```

516 \def\NewEnvironmentCopy{%
517   \declare@environmentcopy
518     {\@firstofone}%
519     {\@firstoftwo\@notdefinable}}
520 \def\RenewEnvironmentCopy{%
521   \declare@environmentcopy
522     {\@latex@error{Environment \reserved@a\space undefined}\@ehc
523     \@firstofone}%
524     {\@firstofone}}
525 \def\DeclareEnvironmentCopy{%
526   \declare@environmentcopy
527     {\@firstofone}%
528     {\@firstofone}}
529 \long\def\declare@environmentcopy#1#2#3#4{%
530   \edef\reserved@a{\@ifundefined{#3}{end#3}{#3}}%
531   \@ifundefined\reserved@a
532     {\def\reserved@a{#3}#1}%
533     {\def\reserved@a{#3}#2}%
534     {\ExpandArgs{cc}\declare@commandcopy@do{#3}{#4}%
535     \ExpandArgs{cc}\declare@commandcopy@do{end#3}{end#4}}}
```

Now the rollback code.

```

536 </2kernel | latexrelease>
537 <latexrelease>\EndIncludeInRelease
538 <latexrelease>\IncludeInRelease{0000-00-00}{\DeclareEnvironmentCopy}
539 <latexrelease> {Undefine \NewEnvironmentCopy, \RenewEnvironmentCopy, and \DeclareEnvironmentCopy}
540 <latexrelease>\let\NewEnvironmentCopy\@undefined
541 <latexrelease>\let\RenewEnvironmentCopy\@undefined
542 <latexrelease>\let\DeclareEnvironmentCopy\@undefined
543 <latexrelease>\EndIncludeInRelease
544 <*2kernel>
```

(End of definition for `\NewEnvironmentCopy`, `\RenewEnvironmentCopy`, and `\DeclareEnvironmentCopy`.)

1.5.2 Showing robust commands

`\ShowCommand` Most of the machinery defined for `\NewCommandCopy` can be used to show the definition of a robust command, in a similar fashion to `\texdef`. The difference is that after the command is detected to have a given type of robustness, rather than making a copy, we use a separate routine to show its definition.

With all the machinery in place, `\ShowCommand` itself is quite simple: we use `\robust@command@act` to iterate through the `\@showcommandlisthook` list, and if nothing is found, fallback to `\show`.

```

545 </2kernel>
546 <latexrelease>\IncludeInRelease{2020-10-01}{\ShowCommand}%
547 <latexrelease> {Add \ShowCommand}%
548 <*2kernel | latexrelease>
549 \long\def\ShowCommand#1{%
550   \robust@command@act
551     \@showcommandlisthook#1%
552     \show#1}
```

`\@showcommandlisthook` The initial definition of `\@showcommandlisthook` contains the same tests as used for copying, but `\@show@...` commands instead of `\@copy@...`. Same as before, it is initialized to cope with `\DeclareRobustCommand` and `\newcommand` with optional arguments.

```
553 \def\@showcommandlisthook{%
554   {\@if@DeclareRobustCommand \@show@DeclareRobustCommand}%
555   {\@if@newcommand \@show@newcommand}}
```

Now the rollback code.

```
556 </2ekernel | latexrelease>
557 <latexrelease>\EndIncludeInRelease
558 <latexrelease>\IncludeInRelease{0000-00-00}{\ShowCommand}
559 <latexrelease> {Undefine \ShowCommand}%
560 <latexrelease>\let\ShowCommand\@undefined
561 <latexrelease>\let\@showcommandlisthook\@undefined
562 <latexrelease>\EndIncludeInRelease
563 <*2ekernel>
```

(End of definition for \ShowCommand and \@showcommandlisthook.)

```
564 </2ekernel>
565 <latexrelease>\IncludeInRelease{2020-10-01}{\@if@DeclareRobustCommand}
566 <latexrelease> {Add \@if@DeclareRobustCommand, \@if@newcommand,
567 <latexrelease>         \@copy@DeclareRobustCommand, \@copy@newcommand,
568 <latexrelease>         \@show@DeclareRobustCommand, \@show@newcommand}%
569 <*2ekernel | latexrelease>
```

1.5.3 Commands defined with `\DeclareRobustCommand`

`\@if@DeclareRobustCommand` Now that we provided a generic way to copy one macro to another, we need to define a way to check if a command is one of L^AT_EX 2_ε's robust types. These tests are heavily based on Heiko's `\LetLtxMacro`, but chopped into separate macros.

The command `\@if@DeclareRobustCommand` checks if a command `\cmd` was defined by `\DeclareRobustCommand`. The test returns true if the expansion of `\cmd` is exactly `\protect\cmdL`.

```
570 \long\def\@if@DeclareRobustCommand#1{%
571   \begingroup
572     \escapechar='\\
573     \edef\reserved@a{\string#1}%
574     \edef\reserved@b{\detokenize{#1}}%
575     \xdef\@gtempa{%
576       \ifx\reserved@a\reserved@b
577         \noexpand\x@protect
578         \noexpand#1%
579       \fi
580       \noexpand\protect
581       \expandafter\noexpand\csname\@expl@cs@to@str@N#1 \endcsname}%
582   \endgroup
583   \ifx\@gtempa#1\relax
584     \expandafter\@firstoftwo
585   \else
586     \expandafter\@secondoftwo
587   \fi}
```

```
\@copy@DeclareRobustCommand
\copy@kernel@robust@command
```

If a command was defined by `\DeclareRobustCommand` (that is, `\@if@DeclareRobustCommand` returns true), then to make a copy of `\cmd` into `\foo` we define the latter such that it expands to `\protect\foo`, then make `\foo` equal to `\cmd`.

There is one detail we need to take care of: if a command was defined with `\DeclareRobustCommand` it may still have an optional argument, in which case there is one more macro layer before the actual definition of the command. We use `\@if@newcommand` to check that and `\@copy@newcommand` to do the copying.

```
588 \long\def\@copy@DeclareRobustCommand#1#2{%
589   \begingroup
590     \escapechar='\'
591     \edef\reserved@a{\string#1}%
592     \edef\reserved@b{\detokenize{#1}}%
593     \edef\reserved@a{%
594   \endgroup
595   \def\noexpand#1{%
596     \ifx\reserved@a\reserved@b
597       \noexpand\x@protect
598       \noexpand#1%
599     \fi
600     \noexpand\protect
601     \expandafter\noexpand\csname\@expl@cs@to@str@N#1 \endcsname}%
602   \noexpand\copy@kernel@robust@command
603     \expandafter\noexpand\csname\@expl@cs@to@str@N#1 \endcsname
604     \expandafter\noexpand\csname\@expl@cs@to@str@N#2 \endcsname}%
605   \reserved@a}
606 \long\def\copy@kernel@robust@command#1#2{%
607   \robust@command@chk@safe#2%
608   {\@if@newcommand#2%
609     {\@copy@newcommand}%
610     {\declare@commandcopy@let}}
611   {\declare@commandcopy@let}%
612   #1#2}
```

```
\@show@DeclareRobustCommand
\show@kernel@robust@command
\@show@macro
```

Showing the command is pretty simple. This command prints the top-level expansion as TeX's `\show` would, but with `robust macro`: rather than just `macro`;, then a blank line and then `\show` the inner command. For a macro defined with, say, `\DeclareRobustCommand\foo[1]{bar}`, it will print:

```
> \foo=robust macro:
->\protect \foo

> \foo =\long macro:
#1->bar.
```

If the inner command is defined with an optional argument, then `\@show@newcommand` is also used.

The value of `\escapechar` is deliberately not enforced, so `\ShowCommand` behaves more like `\show`.

```
613 \long\def\@show@DeclareRobustCommand#1{%
614   \typeout{> \string#1=robust macro:}%
615   \typeout{->\@expl@cs@replacement@spec@N#1.^^J}%
616   \expandafter\show@kernel@robust@command
617     \csname\@expl@cs@to@str@N#1 \endcsname}
```



```

618 \long\def\show@kernel@robust@command#1{%
619   \robust@command@chk@safe#1%
620   {\@if@newcommand#1%
621     {\@show@newcommand}%
622     {\@show@macro}}}%
623   {\@show@macro}%
624   #1}
625 \let\@show@macro\show

```

(End of definition for \@if@DeclareRobustCommand and others.)

1.5.4 Commands defined with \newcommand (with optional argument)

`\@if@newcommand` A command `\cmd` (or `\cmdl`, if it was defined with `\DeclareRobustCommand`) with an optional argument will expand to `\@protected@testopt\cmd\cmd{<opt>}`. To check that we look at the first three tokens in the expansion of `\cmd`, and return true or false accordingly.

This test *requires* that the command be a parameterless macro, otherwise it will not work (and probably break). This is ensured with `\robust@command@chk@safe` before calling `\@if@newcommand`.

```

626 \long\def\@if@newcommand#1{%
627   \edef\reserved@a{%
628     \noexpand\@protected@testopt
629     \noexpand#1%
630     \expandafter\noexpand\csname\@backslashchar\@expl@cs@to@str@@N#1\endcsname}%
631   \edef\reserved@b{%
632     \unexpanded\expandafter\expandafter\expandafter
633     {\expandafter\@carcube#1-{}-{}-{}@nil}}}%
634   \ifx\reserved@a\reserved@b
635     \expandafter\@firstoftwo
636   \else
637     \expandafter\@secondoftwo
638   \fi}

```

Then, if a command `\cmd` takes an optional argument, we copy it to `\foo` by defining `\@copy@newcommand` the latter to expand to `\@protected@testopt\foo\foo{<opt>}`.

```

639 \long\def\@copy@newcommand#1#2{%
640   \edef#1{\noexpand\@protected@testopt
641     \noexpand#1%
642     \expandafter\noexpand\csname\@backslashchar\@expl@cs@to@str@@N#1\endcsname
643     \unexpanded\expandafter\expandafter\expandafter
644     {\expandafter\@gobblethree#2}}}%
645   \expandafter
646   \let\csname\@backslashchar\@expl@cs@to@str@@N#1\endcsname
647     \csname\@backslashchar\@expl@cs@to@str@@N#2\endcsname}

```

`\@show@newcommand`
`\@show@newcommand@aux`

A command being `\shown` here is guaranteed to have an optional argument. Start by showing the top-level expansion of the command (using `\typeout` to avoid TeX asking for interaction and extra context lines), then call `\@show@newcommand@aux` with the internal command, which contains the actual definition, and with the expansion of the command to extract the default value of the optional argument.

```

648 \long\def\@show@newcommand#1{%
649   \typeout{> \string#1=robust macro:}%

```

```

650 \typeout{->\@expl@cs@replacement@spec@N#1.^^J}%
651 \expandafter\@show@newcommand@aux
652 \csname\@backslashchar\@expl@cs@to@str@N#1\expandafter\endcsname
653 \expandafter{#1}\@show@tokens}

```

For a macro defined with, say, `\newcommand\foo[1][opt]{bar}`, it will print:

```

> \foo=robust macro:
->\@protected@testopt \foo \foo {opt}.

> \foo=\long macro:
> default #1=opt.
[#1]->bar.

```

If the command was defined with `\DeclareRobustCommand`, then another pair of lines show the top-level expansion `\protect\foo`.

```

654 \long\def\@show@newcommand@aux#1#2#3{%
655 \typeout{> \string#1=\@expl@cs@prefix@spec@N#1macro:}%
656 #3{default \string##1=\expandafter\detokenize\@gobblethree#2.^^J%
657 \@expl@cs@parameter@spec@N#1->\@expl@cs@replacement@spec@N#1}}

```

This macro prints the contents of the token list (macro) #1 using `\showtokens`. The `\@show@tokens` `\expandafter` gymnastics ensures that `\showtokens` itself, and the internals of this macro aren't showed in the context lines.

```

658 \long\def\@show@tokens#1{%
659 \edef\reserved@a{#1}%
660 \showtokens\expandafter
661 \expandafter\expandafter{\expandafter\reserved@a}}

```

Now the rollback code.

```

662 </2kernel | latexrelease>
663 <latexrelease>\EndIncludeInRelease
664 <latexrelease>\IncludeInRelease{0000-00-00}{\@if@DeclareRobustCommand}
665 <latexrelease> {Undefine \@if@DeclareRobustCommand, \@if@newcommand,
666 <latexrelease> \copy@DeclareRobustCommand, \copy@newcommand,
667 <latexrelease> \show@DeclareRobustCommand, \show@newcommand}%
668 <latexrelease>\let\@if@DeclareRobustCommand\undefined
669 <latexrelease>\let\@copy@DeclareRobustCommand\undefined
670 <latexrelease>\let\@show@DeclareRobustCommand\undefined
671 <latexrelease>\let\@if@newcommand\undefined
672 <latexrelease>\let\@copy@newcommand\undefined
673 <latexrelease>\let\@show@newcommand\undefined
674 %
675 <latexrelease>\let\copy@kernel@robust@command\undefined
676 <latexrelease>\let\show@kernel@robust@command\undefined
677 <latexrelease>\let\@show@newcommand@aux\undefined
678 <latexrelease>\EndIncludeInRelease
679 <*2kernel>

```

(End of definition for \@if@newcommand and others.)

1.5.5 Showing environments

`\ShowEnvironment`

```
680 </2ekernel>
681 <latexrelease>\IncludeInRelease{2023-06-01}{\ShowEnvironment}
682 <latexrelease> {Add \ShowEnvironment}%
683 <*2ekernel | latexrelease>
```

`\ShowEnvironment` is quite similar to `\ShowCommand`. We will pass the environment `<env>` around as the macro `\env`, because `\robust@command@act` expects a single token.

```
684 \def\ShowEnvironment#1{%
685   \expandafter\@show@environment\csname #1\endcsname}
686 \long\def\@show@environment#1{%
687   \robust@command@act
688   \@showenvironmentlisthook#1%
689   \@show@normalenv#1}
```

This is similar to `\@showcommandlisthook`, but uses the dedicated versions for environments.

`\@showenvironmentlisthook`

```
690 \def\@showenvironmentlisthook{%
691   {\@if@DeclareRobustCommand \@show@DeclareRobustCommand@env}%
692   {\@if@newcommand \@show@newcommand@env}}
```

These are similar to the command versions below, except they say “environment” and call `\@show@environment@end` to print the `\end` part.

`\@show@newcommand@env`

`\@show@DeclareRobustCommand@env`

```
693 \long\def\@show@newcommand@env#1{%
694   \@show@environment@begin#1%
695   \expandafter\@show@newcommand@aux
696   \csname\@backslashchar\@expl@cs@to@str@@N#1\expandafter\endcsname
697   \expandafter{#1}\@show@typeout
698   \@show@environment@end#1}
699 \long\def\@show@DeclareRobustCommand@env#1{%
700   \@show@environment@begin#1%
701   \begingroup
702   \let\@show@tokens\@show@typeout
703   \let\@show@macro\@show@nonstop
704   \expandafter\show@kernel@robust@command
705   \csname\@expl@cs@to@str@@N#1 \endcsname
706   \endgroup
707   \@show@environment@end#1}
708 \long\def\@show@environment@begin#1{%
709   \typeout{> \string\begin{\@expl@cs@to@str@@N#1}=environment:}%
710   \typeout{\@expl@cs@parameter@spec@@N#1->%
711     \@expl@cs@replacement@spec@@N#1.^^J}}
```

A “normal” environment is straightforward. `\@show@environment@end` needs to check if the `\end` part is defined and show it accordingly, otherwise the output would show gibberish.

`\@show@normalenv`

`\@show@environment@end`

```
712 \long\def\@show@normalenv#1{%
713   \@show@environment@begin#1%
714   \@show@environment@end#1}
715 \long\def\@show@environment@end#1{%
716   \expandafter\@show@environment@end@aux
717   \csname end\@expl@cs@to@str@@N#1\endcsname#1}
```

```

718 \long\def\@show@environment@end@aux#1#2{%
719   \@show@tokens{\string\end{\@expl@cs@to@str@N#2}}%
720   \ifx\relax#1=undefined%
721     \else:^^J\@expl@cs@parameter@spec@N#1->%
722       \@expl@cs@replacement@spec@N#1%
723   \fi}}

```

And here some auxiliaries:

`\@show@nonstop`
`\@show@typeout`

`\@show@nonstop` same output as `\show`, but doesn't stop for interaction;

`\@show@typeout` same output as `\showtokens`, but doesn't stop for interaction.

```

724 \def\@show@nonstop#1{%
725   \typeout{> \string#1=\@expl@cs@prefix@spec@N#1macro:^^J%
726   \@expl@cs@parameter@spec@N#1->\@expl@cs@replacement@spec@N#1.}}
727 \def\@show@typeout#1{\typeout{> #1.^^J}}

```

Now the rollback code.

```

728 </2ekernel | latexrelease>
729 <latexrelease>\EndIncludeInRelease
730 <latexrelease>\IncludeInRelease{0000-00-00}{\ShowEnvironment}
731 <latexrelease> {Undefine \ShowEnvironment}%
732 <latexrelease>\let\ShowEnvironment\@undefined
733 <latexrelease>\EndIncludeInRelease
734 <*2ekernel)

```

(End of definition for \ShowEnvironment and others.)

1.6 Internal defining commands

These commands are used internally to define other L^AT_EX commands.

`\@ifundefined` Check if first arg is undefined or `\relax` and execute second or third arg depending,

```

735 </2ekernel)
736 <latexrelease>\IncludeInRelease{2018-04-01}{\@ifundefined}
737 <latexrelease>{Leave commands undefined in \@ifundefined}%
738 <*2ekernel | latexrelease>

```

Version using `\ifcsname` to avoid defining undefined tokens to `\relax`. Defined here to simplify using unmatched `\fi`.

```

739 \def\@ifundefined#1{%
740   \ifcsname#1\endcsname\@ifundefin@d\i\else\@ifundefin@d\ii\fi{#1}}
741 \long\def\@ifundefin@d\i#1\fi#2{\fi
742   \expandafter\ifx\csname #2\endcsname\relax
743     \@ifundefin@d\ii
744   \fi
745   \@secondoftwo}
746 \long\def\@ifundefin@d\ii\fi#1#2#3{\fi #2}

```

Now test of engine.

```

747 \ifx\numexpr\@undefined

```

Classic version (should not be needed as etex is assumed).

```
748 \def\@ifundefined#1{%
749   \expandafter\ifx\csname#1\endcsname\relax
750   \expandafter\@firstoftwo
751   \else
752     \expandafter\@secondoftwo
753   \fi}
754 \else\ifx\directlua\@undefined
```

Use the \ifcsname defined above.

```
755 \else
```

Optimised version for LuaTeX, using \lastnamedcs

```
756 \def\@ifundefined#1{%
757   \ifcsname#1\endcsname
758   \expandafter\ifx\lastnamedcs\relax\else\@ifundefin@d@i\fi
759   \fi
760   \@firstoftwo}

761 \long\def\@ifundefin@d@i#1#2#3#4#5{#1#2#5}

762 \fi
763 \fi
764 </2ekernel | latexrelease>
765 <latexrelease>\EndIncludeInRelease
766 <latexrelease>\IncludeInRelease{0000-00-00}{\@ifundefined}
767 <latexrelease>{Leave commands undefined in \@ifundefined}%
768 <latexrelease>\def\@ifundefined#1{%
769 <latexrelease> \expandafter\ifx\csname#1\endcsname\relax
770 <latexrelease> \expandafter\@firstoftwo
771 <latexrelease> \else
772 <latexrelease> \expandafter\@secondoftwo
773 <latexrelease> \fi}
774 <latexrelease>\EndIncludeInRelease
775 <*2ekernel>
```

(End of definition for \@ifundefined.)

`\@qend` The following define `\@qend` and `\@qrelax` to be the strings ‘end’ and ‘relax’ with the
`\@qrelax` characters `\catcoded 12`.

```
776 \edef\@qend{\expandafter\@cdr\string\end\@nil}
777 \edef\@qrelax{\expandafter\@cdr\string\relax\@nil}
```

(End of definition for \@qend and \@qrelax.)

`\@ifnextchar` `\@ifnextchar` peeks at the following character and compares it with its first argument.
If both are the same it executes its second argument, otherwise its third.

```
778 \long\def\@ifnextchar#1#2#3{%
779   \let\reserved@d=#1%
780   \def\reserved@a{#2}%
781   \def\reserved@b{#3}%
782   \futurelet\@let@token\@ifnch}
```

(End of definition for \@ifnextchar.)

`\kernel@ifnextchar` This macro is the kernel version of `\@ifnextchar` which is used in a couple of places to prevent the AMS variant from being used since in some places this produced chaos. For example, if an `fd` file is loaded in a random place then the optional argument to `\ProvidesFile` could get printed there instead of being written only in the log file. This happened when there was a space or a newline between the mandatory and optional arguments! It should really be fixed in the `amsmath` package one day, but...

Note that there may be other places in the kernel where this version should be used rather than the original, but variable, version.

```
783 \let\kernel@ifnextchar\@ifnextchar
```

(End of definition for `\kernel@ifnextchar`.)

`\@ifnch` `\@ifnch` is a tricky macro to skip any space tokens that may appear before the character in question. If it encounters a space token, it calls `xifnch`.

```
784 \def\@ifnch{%
785   \ifx\@let@token\@sptoken
786     \let\reserved@c\@xifnch
787   \else
788     \ifx\@let@token\reserved@d
789       \let\reserved@c\reserved@a
790     \else
791       \let\reserved@c\reserved@b
792     \fi
793   \fi
794   \reserved@c}
```

(End of definition for `\@ifnch`.)

`\@sptoken` The following code makes `\@sptoken` a space token. It is important here that the control sequence `\:` consists of a non-letter only, so that the following whitespace is significant. Together with the fact that the equal sign in a `\let` may be followed by only one optional space the desired effect is achieved. NOTE: the following hacking must precede the definition of `\:` as math medium space.

```
795 \def\:\{\let\@sptoken= } \: % this makes \@sptoken a space token
```

(End of definition for `\@sptoken`.)

`\@xifnch` In the following definition of `\@xifnch`, `\:` is again used to get a space token as delimiter into the definition.

```
796 \def\:\{\@xifnch} \expandafter\def\:\: {\futurelet\@let@token\@ifnch}
```

(End of definition for `\@xifnch`.)

`\@ifstar` The new implementation below avoids passing the `<true code>` Through one more `\def` than the `<false code>`, which previously meant that `#` had to be written as `####` in one argument, but `##` in the other. The `*` is gobbled by `\@firstoftwo`.

```
797 \def\@ifstar#1{\@ifnextchar *{\@firstoftwo{#1}}}
```

(End of definition for `\@ifstar`.)

`\@dblarg`

```
798 \long\def\@dblarg#1{\kernel@ifnextchar[#{#1}]{\@dblarg{#1}}}
```

```
799 \long\def\@dblarg#1#2[#{#1}[#{#2}]{#2}}
```

(End of definition for `\@dblarg` and `\@xdblarg`.)

`\@sanitize` The command `\@sanitize` changes the catcode of all special characters except for braces to ‘other’. It can be used for commands like `\index` that want to write their arguments verbatim. Needless to say, this command should only be executed within a group, or chaos will ensue.

```
800 \def\@sanitize{\@makeother\ \@makeother\\\@makeother$\@makeother\&%
801 \@makeother#\@makeother^\@makeother_\@makeother%\@makeother\~}
```

(End of definition for `\@sanitize`.)

`\@onelevel@sanitize` This makes the whole “meaning” of #1 (its one-level expansion) into catcode 12 tokens: it could be used in `\DeclareRobustCommand`.

If it is to be used on default float specifiers, this should be done when they are defined.

```
802 \def \@onelevel@sanitize #1{%
803   \edef #1{\expandafter\strip@prefix
804     \meaning #1}%
805 }
```

(End of definition for `\@onelevel@sanitize`.)

`\string@makeletter` Iterates through a string, turning each alphabetic character into a catcode-11 token (partly undoes a `\detokenize`). Useful for `\ifx`-based string comparisons where `\detokenize`-ing the other string would break too much code.

The macro uses `expl3`’s `\@expl@str@map@function@NN` to iterate on the string (without losing spaces) and applies `\string@makeletter` on each character. The latter checks if character is between a–z or A–Z, and uses `\@alph` or `\@Alph` to get the corresponding catcode-11 token. Other tokens are passed through unchanged.

```
806 </2ekernel>
807 <latexrelease>\IncludeInRelease{2020/10/01}{\string@makeletter}
808 <latexrelease> {Add \string@makeletter}%
809 <*2ekernel | latexrelease>
810 \def\string@makeletter#1{%
811   \@expl@str@map@function@NN#1\string@makeletter}
812 \def\@string@makeletter#1{%
813   \@char@if@alph{#1}%
814   {\@expl@char@generate@nn{#1}{11}}%
815   {#1}}
816 \def\@char@if@alph#1{%
817   \ifnum0\ifnum'#1<'A 1\fi\ifnum'#1>'z 1\fi
818     \if\ifnum'#1>'Z @\fi\ifnum'#1<'a @\fi01\fi>0
819     \expandafter\@secondoftwo
820   \else
821     \expandafter\@firstoftwo
822   \fi}
823 </2ekernel | latexrelease>
824 <latexrelease>\EndIncludeInRelease
825 %
826 <latexrelease>\IncludeInRelease{0000/00/00}{\string@makeletter}
827 <latexrelease> {Undefine \string@makeletter}%
828 <latexrelease>\let\string@makeletter\undefined
829 <latexrelease>\let\@string@makeletter\undefined
```

```

830 <latexrelease>\let\char@if@alph\@undefined
831 <latexrelease>\EndIncludeInRelease
832 <*2ekernel>

```

(End of definition for `\string@makeletter`, `\@string@makeletter`, and `\char@if@alph`.)

`\makeatletter` Make internal control sequences accessible or inaccessible.

```

\makeatother 833 \DeclareRobustCommand\makeatletter{\catcode'\@11\relax}
834 \DeclareRobustCommand\makeatother{\catcode'\@12\relax}

```

(End of definition for `\makeatletter` and `\makeatother`.)

2 Discretionary Hyphenation

`\-`
`\@dischyph`

Moved here to be after the definition of `\DeclareRobustCommand`.

The primitive `\-` command adds a discretionary hyphen using the current font's `\hyphenchar`. Monospace fonts are usually declared with `\hyphenchar` set to `-1` to suppress hyphenation.

L^AT_EX, from L^AT_EX2.09 in 1986 defined `\-` by

```
\def\-\{\discretionary{-}{-}{-}}
```

The following comment was added when these commands were first set up, 19 April 1986:

the `\-` command is redefined to allow it to work in the `\ttfamily` type style, where automatic hyphenation is suppressed by setting `\hyphenchar` to `-1`. The original primitive T_EX definition is saved as `\@hyph` just in case anyone needs it.

L^AT_EX 2_ε, between 1993 and 2017, had a comment at this point saying that the definition “would probably change” because the definition always uses `-`. The definition used below was given in comments at this point during time.

In 2017 we finally enabled this definition by default, with the older L^AT_EX definition accessible via `latexrelease` as usual.

In LuaL^AT_EX the primitive definition of `\-` is used directly because it's use of extended hyphenation parameters means that `\-` works correctly even with `\hyphenchar` set to `-1`. This change makes `\-` under LuaL^AT_EX compatible with language specific hyphenation characters.

Temporary definition of `\@latex@info`, final definition is later.

```

835 \def\@latex@info#1{}
836 </2ekernel>
837 <latexrelease>\IncludeInRelease{2020/10/01}{\-\}{Use primitive \- in Lua\LaTeX}%
838 <*2ekernel | latexrelease>
839 \ifx\directlua\@undefined
840   \DeclareRobustCommand{\-\}{%
841     \discretionary{%
842       \char \ifnum\hyphenchar\font<\z@
843         \defaultthyphenchar
844       \else
845         \hyphenchar\font
846       \fi

```



```

847             }{}{}%
848         }
849     \else
850         \let\-\@hyph
851     \fi
852 </2ekernel | latexrelease>
853 <latexrelease>\EndIncludeInRelease
854 <latexrelease>\IncludeInRelease{2017/04/15}{\-}{Use \hyphenchar in \-}%
855 <latexrelease>\DeclareRobustCommand{-}{%
856 <latexrelease> \discretionary{%
857 <latexrelease> \char \ifnum\hyphenchar\font<\z@
858 <latexrelease> \defaultshyphenchar
859 <latexrelease> \else
860 <latexrelease> \hyphenchar\font
861 <latexrelease> \fi
862 <latexrelease> }{}{}%
863 <latexrelease>}
864 <latexrelease>\EndIncludeInRelease
865 <latexrelease>\IncludeInRelease{0000/00/00}{\-}{Use \hyphenchar in \-}%
866 <latexrelease>\def\-\{discretionary{-}{-}{-}\}
867 <latexrelease>\EndIncludeInRelease

868 <*2ekernel | latexrelease>
869 \let\@dischyph=\-
870 </2ekernel | latexrelease>
871 <*2ekernel>

(End of definition for \- and \@dischyph.)
    Delayed from ltvers.dtx

872 \newif\if@includeinrelease
873 \@includeinreleasefalse

    Delayed from ltplain.dtx

874 </2ekernel>
875 <*2ekernel | latexrelease>
876 <latexrelease>\IncludeInRelease{2019/10/01}%
877 <latexrelease> \allowbreak}{Make various commands robust}%
878 \MakeRobust\allowbreak
879 \MakeRobust\bigbreak
880 \MakeRobust\break
881 \MakeRobust\dotfill
882 \MakeRobust\frenchspacing
883 \MakeRobust\goodbreak
884 \MakeRobust\hrulefill
885 \MakeRobust\medbreak
886 \MakeRobust\nobreak
887 \MakeRobust\nonfrenchspacing
888 \MakeRobust\obeylines
889 \MakeRobust\obeyspaces
890 \MakeRobust\slash
891 \MakeRobust\smallbreak
892 \MakeRobust\strut
893 \MakeRobust\underbar
894 </2ekernel | latexrelease>
895 <latexrelease>\EndIncludeInRelease

```

```

896 <latexrelease>\IncludeInRelease{0000/00/00}%
897 <latexrelease>          {\allowbreak}{Make various commands robust}%
898 <latexrelease>
899 <latexrelease>\kernel@make@fragile\allowbreak
900 <latexrelease>\kernel@make@fragile\bigbreak
901 <latexrelease>\kernel@make@fragile\break
902 <latexrelease>\kernel@make@fragile\dotfill
903 <latexrelease>\kernel@make@fragile\frenchspacing
904 <latexrelease>\kernel@make@fragile\goodbreak
905 <latexrelease>\kernel@make@fragile\hrulefill
906 <latexrelease>\kernel@make@fragile\medbreak
907 <latexrelease>\kernel@make@fragile\nobreak
908 <latexrelease>\kernel@make@fragile\nonfrenchspacing
909 <latexrelease>\kernel@make@fragile\obeylines
910 <latexrelease>\kernel@make@fragile\obeyspaces
911 <latexrelease>\kernel@make@fragile\slash
912 <latexrelease>\kernel@make@fragile\smallbreak
913 <latexrelease>\kernel@make@fragile\strut
914 <latexrelease>\kernel@make@fragile\underbar
915 <latexrelease>
916 <latexrelease>\EndIncludeInRelease
917 <*2kernel>

```

`\g@addto@macro` Globally add to the end of a macro. This macro is used by the kernel to add to its internal hooks.

```

918 \long\def\g@addto@macro#1#2{%
919   \begingroup
920     \toks@\expandafter{#1#2}%
921     \xdef#1{the\toks@}%
922   \endgroup}

```

(End of definition for \g@addto@macro.)

```

923 </2kernel>

```

File 07

ltxcmd.dtx

1 Creating document commands

Document commands should be created using the tools provided by this module: `\NewDocumentCommand`, etc., in almost all cases. This allows clean separation of document-level syntax from code-level interfaces. Users have a need to create new document commands, and as such a significant amount of documentation for `ltxcmd` is provided as part of `usrguide3`. Here, additional material aimed at programmers is provided

```
1 <@@=cmd>
2 <*2ekernel>
3 \message{document commands,}
4 </2ekernel>
```

`ltxcmd` code contains an `^^@` character, which usually has catcode 15, so `\IncludeInRelease` will break when this code is being skipped, so we'll save the catcode of `^^@` to restore later:

```
5 <*2ekernel | latexrelease>
6 <latexrelease> \edef\@latexrelease@catcode@null{\the\catcode'\^^@ }
7 <latexrelease> \catcode'\^^@=12
8 \ExplSyntaxOn
9 <latexrelease> \NewModuleRelease{2020/10/01}{ltxcmd}
10 <latexrelease> {Document~command~parser}%
11 \cs_generate_variant:Nn \tl_head:n { e }
```

1.1 Variables and constants

`\l__cmd_arg_spec_tl` Holds the argument specification after normalization of shorthands.

```
12 \tl_new:N \l__cmd_arg_spec_tl
```

`\l__cmd_args_tl` Token list variable for grabbed arguments.

```
13 \tl_new:N \l__cmd_args_tl
```

`\l__cmd_args_i_tl` Hold the modified arguments when dealing with default values or processors.

`\l__cmd_args_ii_tl`

```
14 \tl_new:N \l__cmd_args_i_tl
```

```
15 \tl_new:N \l__cmd_args_ii_tl
```

`\l__cmd_current_arg_int` The number of the current argument being set up: this is used to make sure there are at most 9 arguments, then for creating the expandable auxiliary functions and knowing how many arguments the code function should take.

```
16 \int_new:N \l__cmd_current_arg_int
```

\l__cmd_total_args_int The total number of arguments found during normalization: this is required where special action is needed for the penultimate argument.

17 \int_new:N \l__cmd_total_args_int

\l__cmd_defaults_bool
\l__cmd_defaults_tl The boolean indicates whether there are any argument with default value other than -NoValue-; the token list holds the code to determine these default values in terms of other arguments.

18 \bool_new:N \l__cmd_defaults_bool

19 \tl_new:N \l__cmd_defaults_tl

\l__cmd_environment_bool Generating environments uses the same mechanism as generating functions. However, full processing of arguments is always needed for environments, and so the function-generating code needs to know this. This variable is also used at run time to give correct error messages.

20 \bool_new:N \l__cmd_environment_bool

\l__cmd_environment_str Name of the environment, used at definition time and at run time.

21 \str_new:N \l__cmd_environment_str

\l__cmd_expandable_bool Used to indicate if an expandable command is begin generated, as this affects both the acceptable argument types and how they are implemented.

22 \bool_new:N \l__cmd_expandable_bool

\l__cmd_expandable_aux_name_tl

Used to create pretty-printing names for the auxiliaries: although the immediate definition does not vary, the full expansion does and so it does not count as a constant.

23 \tl_new:N \l__cmd_expandable_aux_name_tl

24 \tl_set:Nn \l__cmd_expandable_aux_name_tl

25 {

26 \l__cmd_function_tl \c_space_tl

27 (arg~ \int_use:N \l__cmd_current_arg_int)

28 }

\g__cmd_grabber_int Used (in exceptional cases) to get unique names for grabbers used by expandable commands.

29 \int_new:N \g__cmd_grabber_int

\l__cmd_fn_tl For passing the pre-formed name of the auxiliary to be used as the parsing function.

30 \tl_new:N \l__cmd_fn_tl

\l__cmd_fn_code_tl For passing the pre-formed name of the auxiliary that contains the actual code.

31 \tl_new:N \l__cmd_fn_code_tl

\l__cmd_function_tl Holds the control sequence name of the function currently being defined: used to avoid passing this as an argument and to avoid repeated use of \cs_to_str:N.

32 \tl_new:N \l__cmd_function_tl

\l__cmd_grab_expandably_bool

When defining a non-expandable command, indicates whether the arguments can all safely be grabbed by expandable grabbers. This is to support abuses of `xparse` that use protected functions inside `csname` constructions.

33 \bool_new:N \l__cmd_grab_expandably_bool

\l__cmd_obey_spaces_bool For trailing optionals.

34 \bool_new:N \l__cmd_obey_spaces_bool

\l__cmd_last_delimiters_tl Holds the delimiters (first tokens) of all optional arguments since the previous mandatory argument, to warn about cases where it would be impossible to omit optional arguments completely because the following mandatory argument has the same delimiter as one of the optional arguments.

35 \tl_new:N \l__cmd_last_delimiters_tl

\l__cmd_long_bool Used to indicate that an argument is long, on a per-argument basis.

36 \bool_new:N \l__cmd_long_bool

\l__cmd_suppress_strip_bool

Used to indicate that an a pair of braces should not be stripped from an optional argument.

37 \bool_new:N \l__cmd_suppress_strip_bool

\l__cmd_m_args_int The number of `m` arguments: if this is the same as the total number of arguments, then a short-cut can be taken in the creation of the grabber code.

38 \int_new:N \l__cmd_m_args_int

\l__cmd_prefixed_bool When preparing the signature of non-expandable commands, indicates that the current argument is affected by a processor or by `+` (namely is long).

39 \bool_new:N \l__cmd_prefixed_bool

`\l__cmd_process_all_tl`
`\l__cmd_process_one_tl`
`\l__cmd_process_some_bool`

When preparing the signature, the processors that will be applied to a given argument are collected in `\l__cmd_process_one_tl`, while `\l__cmd_process_all_tl` contains processors for all arguments. The boolean indicates whether there are any processors (to bypass the whole endeavour otherwise).

```
40 \tl_new:N \l__cmd_process_all_tl
41 \tl_new:N \l__cmd_process_one_tl
42 \bool_new:N \l__cmd_process_some_bool
```

`\l__cmd_saved_args_tl` Stores `\l__cmd_args_tl` to deal with space-trimming of b-type arguments.

```
43 \tl_new:N \l__cmd_saved_args_tl
```

`\l__cmd_signature_tl` Used when constructing the signature (code for argument grabbing) to hold what will become the implementation of the main function. When arguments are grabbed (at point of use of the command/environment), it also stores the code for grabbing the remaining arguments.

```
44 \tl_new:N \l__cmd_signature_tl
```

`\l__cmd_some_obey_spaces_bool`
`\l__cmd_some_long_bool`
`\l__cmd_some_short_bool`

These flags are set while normalizing the argument specification. The `obey_spaces` one is used to detect when `!` is used on an argument that is not a trailing optional argument. The other two are used to check whether all short arguments appear before long arguments: this is needed to grab arguments expandably. As soon as the first long argument is seen (other than t-type, whose long status is ignored) the `some_long` flag is set. The `some_short` flag is used for expandable commands, to know whether to define a short auxiliary too.

```
45 \bool_new:N \l__cmd_some_obey_spaces_bool
46 \bool_new:N \l__cmd_some_long_bool
47 \bool_new:N \l__cmd_some_short_bool
```

`\l__cmd_final_verb_bool` Needed to establish whether optional arguments should be collected “verbatim safe”.

```
48 \bool_new:N \l__cmd_final_verb_bool
```

`\q__cmd_recursion_tail`

`\q__cmd_recursion_stop` Quarks and functions for internal processing.

`__cmd_use_i_delimit_by_q_recursion_stop:nw`

```
49 \quark_new:N \q__cmd_recursion_tail
50 \quark_new:N \q__cmd_recursion_stop
51 \__kernel_quark_new_test:N \__cmd_if_recursion_tail_stop_do:Nn
```

(End of definition for `__cmd_if_recursion_tail_stop_do:Nn` and
`__cmd_use_i_delimit_by_q_recursion_stop:nw`)

```

\l__cmd_tmp_prop
\l__cmd_tmpa_tl
\l__cmd_tmpb_tl
52 \prop_new:N \l__cmd_tmp_prop
53 \tl_new:N \l__cmd_tmpa_tl
54 \tl_new:N \l__cmd_tmpb_tl
55 \cs_new_eq:NN \__cmd_tmp:w ?

```

Scratch space.

(End of definition for `__cmd_tmp:w`.)

With `xparse`, information about commands being (re)defined was switched off by default, unless the `log-declarations` package option was used, so here we'll switch that off as well.

```
56 \msg_redirect_module:nnn { cmd } { info } { none }
```

Also add `cmd` to the LaTeX messages.

```
57 \prop_gput:Nnn \g_msg_module_type_prop { cmd } { LaTeX }
```

1.2 Declaring commands and environments

```

\__cmd_declare_cmd:Nnn
\__cmd_declare_expandable_cmd:Nnn
\__cmd_declare_cmd_aux:Nnn
\__cmd_declare_cmd_internal:Nnn

```

The main functions for creating commands set the appropriate flag then use the same internal code to do the definition.

```

58 \cs_new_protected:Npn \__cmd_declare_cmd:Nnn
59   {
60     \bool_set_false:N \l__cmd_expandable_bool
61     \__cmd_declare_cmd_aux:Nnn
62   }
63 \cs_new_protected:Npn \__cmd_declare_expandable_cmd:Nnn
64   {
65     \bool_set_true:N \l__cmd_expandable_bool
66     \__cmd_declare_cmd_aux:Nnn
67   }

```

The first stage is to log information, both for the user in the log and for programmatic use in a property list of all declared commands.

```

68 \cs_new_protected:Npn \__cmd_declare_cmd_aux:Nnn #1#2#3
69   {
70     \cs_if_exist:NTF #1
71     {
72       \msg_info:nxxx { cmd } { redefine }
73       { \token_to_str:N #1 } { \tl_to_str:n {#2} }
74     }
75     {
76       \bool_lazy_or:nnT
77       { \cs_if_exist_p:c { \cs_to_str:N #1 ~ code } }
78       { \cs_if_exist_p:c { \cs_to_str:N #1 ~ defaults } }
79       {
80         \msg_warning:nxx { cmd } { unsupported-let }
81         { \token_to_str:N #1 }
82       }
83       \msg_info:nxxx { cmd } { define-command }
84       { \token_to_str:N #1 } { \tl_to_str:n {#2} }
85     }
86     \bool_set_false:N \l__cmd_environment_bool
87     \__cmd_declare_cmd_internal:Nnnn #1 {#2} {#3} { }

```

```
88 }
```

At definition time, the variable `\l__cmd_fn_tl` is only used for error messages. The real business of defining a document command starts with setting up the appropriate name, then normalizing the argument specification to get rid of shorthands.

```
89 \cs_new_protected:Npn \__cmd_declare_cmd_internal:Nnnn #1#2#3#4
90 {
91   \tl_set:Nx \l__cmd_function_tl { \cs_to_str:N #1 }
92   \__cmd_normalize_arg_spec:n {#2}
93   \exp_args:No \__cmd_prepare_signature:n \l__cmd_arg_spec_tl
94   \__cmd_declare_cmd_code:Nnn #1 {#2} {#3}
95   #4
96   \__cmd_break_point:n {#2}
97 }
```

(End of definition for `__cmd_declare_cmd:Nnn` and others.)

`__cmd_break_point:n` A marker used to escape from creating a definition if necessary.

```
98 \cs_new_eq:NN \__cmd_break_point:n \use_none:n
```

(End of definition for `__cmd_break_point:n`.)

`__cmd_all_m_check:n` A quick loop to check for all (+)m-type arguments.

```
\__cmd_all_m_check_aux:n
99 \cs_new:Npn \__cmd_all_m_check:n #1
100 { \tl_map_function:nN {#1} \__cmd_all_m_check_aux:n }
101 \cs_new:Npn \__cmd_all_m_check_aux:n #1
102 {
103   \str_if_eq:nnF {#1} { m }
104   {
105     \str_if_eq:nnF {#1} { + }
106     { X }
107   }
108 }
```

(End of definition for `__cmd_all_m_check:n` and `__cmd_all_m_check_aux:n`.)

`__cmd_declare_cmd_code:Nnn` At this stage we can check for a short-cut possibility: if the argument specification is made up of just (+)m tokens, and if all arguments are either short or long, then we can produce an optimized document command. This only applies to document commands, not creation of environments (which are more complex).

```
\__cmd_declare_cmd_optimized:Nnn
\__cmd_declare_cmd_code_aux:Nnn
\__cmd_declare_cmd_code_expandable:Nnn
\__cmd_start_optimized:
```

```
109 \cs_new_protected:Npn \__cmd_declare_cmd_code:Nnn #1#2
110 {
111   \bool_lazy_any:nTF
112   {
113     { \l__cmd_environment_bool }
114     {
115       \bool_lazy_and_p:nn
116       { \l__cmd_some_short_bool }
117       { \l__cmd_some_long_bool }
118     }
119     { ! \tl_if_blank_p:e { \__cmd_all_m_check:n {#2} } }
120   }
121   {
122     \tl_set:Nx \l__cmd_fn_tl
123     { \exp_not:c { \l__cmd_function_tl \c_space_tl } }
```



```

124     \bool_if:NTF \l__cmd_grab_expandably_bool
125     { \__cmd_declare_cmd_code_expandable:Nnn }
126     { \__cmd_declare_cmd_code_aux:Nnn }
127   }
128   { \__cmd_declare_cmd_optimized:Nnn }
129   #1 {#2}
130 }

```

The optimized version of commands just has to worry about whether to make them protected or long. The commands start with an expandable marker so that other parts of the kernel know these are set up by `ltxcmd`. We need the two layers of redirection so that the code internal function has the same form as it would for any other document command. Optimization means that there is no `\group_align_safe_begin:` before grabbing the arguments, so anything involving `&` tokens will not work. However, this is really only intended for making optional argument processing safe anyway, so in practice should not be an issue.

```

131 \cs_new_protected:Npn \__cmd_declare_cmd_optimized:Nnn #1#2#3
132 {
133   \bool_if:NTF \l__cmd_expandable_bool
134   { \cs_set_nopar:Npe }
135   { \cs_set_protected_nopar:Npe }
136   #1
137   {
138     \exp_not:N \__cmd_start_optimized:
139     \exp_not:c { \l__cmd_function_tl \c_space_tl code }
140   }
141   \exp_args:Ncc \cs_generate_from_arg_count:NNnn
142   { \l__cmd_function_tl \c_space_tl code }
143   {
144     cs_set
145     \bool_if:NF \l__cmd_expandable_bool { _protected }
146     \bool_if:NF \l__cmd_some_long_bool { _nopar }
147     :Npn
148   }
149   \l__cmd_current_arg_int
150   {#3}
151 }
152 \cs_new:Npn \__cmd_start_optimized: { }

```

Standard functions call `__cmd_start:nNNnnn`, which receives the argument specification, an auxiliary used for grabbing arguments, an auxiliary containing the code, and then the signature, default arguments, and processors.

```

153 \cs_new_protected:Npn \__cmd_declare_cmd_code_aux:Nnn #1#2#3
154 {
155   \cs_generate_from_arg_count:cNnn
156   { \l__cmd_function_tl \c_space_tl code }
157   \cs_set_protected:Npn \l__cmd_current_arg_int {#3}
158   \cs_set_protected_nopar:Npx #1
159   {
160     \bool_if:NTF \l__cmd_environment_bool
161     {
162       \__cmd_start_env:nnnnn { \exp_not:n {#2} }
163       { \l__cmd_environment_str }
164     }

```

```

165     {
166       \__cmd_start:nNNnnn { \exp_not:n {#2} }
167       \exp_not:c { \l__cmd_function_tl \c_space_tl }
168       \exp_not:c { \l__cmd_function_tl \c_space_tl code }
169     }
170     { \exp_not:o \l__cmd_signature_tl }
171     {
172       \bool_if:NT \l__cmd_defaults_bool
173       { \exp_not:o \l__cmd_defaults_tl }
174     }
175     {
176       \bool_if:NT \l__cmd_process_some_bool
177       { \exp_not:o \l__cmd_process_all_tl }
178     }
179   }
180 }

```

Expandable functions and functions whose arguments can be grabbed expandably call `__cmd_start_expandable:nNNNNn`, which receives the argument specification, four auxiliaries (two for grabbing arguments, one for the code, and one for default arguments), and finally the signature. Non-expandable functions that take this branch should nevertheless be protected, as well as their code function. They will only be expanded in contexts such as constructing a csname. The two grabbers (named after the function with one or two spaces) are needed when there are both short and long arguments; otherwise the same grabber is included twice in the definition. If all arguments are long or all are short the (only) grabber is defined correspondingly to be long/short. Otherwise two grabbers are defined, one long, one short.

```

181 \cs_new_protected:Npn \__cmd_declare_cmd_code_expandable:Nnn #1#2#3
182 {
183   \exp_args:Ncc \cs_generate_from_arg_count:NNnn
184   { \l__cmd_function_tl \c_space_tl code }
185   { \cs_set \bool_if:NF \l__cmd_expandable_bool { _protected } :Npn }
186   \l__cmd_current_arg_int {#3}
187   \bool_if:NT \l__cmd_defaults_bool
188   {
189     \use:x
190     {
191       \cs_generate_from_arg_count:cNnn
192       { \l__cmd_function_tl \c_space_tl defaults }
193       \cs_set:Npn \l__cmd_current_arg_int
194       { \exp_not:o \l__cmd_defaults_tl }
195     }
196   }
197   \bool_if:NTF \l__cmd_expandable_bool
198   { \cs_set_nopar:Npx } { \cs_set_protected_nopar:Npx } #1
199   {
200     \exp_not:N \__cmd_start_expandable:nNNNNn
201     { \exp_not:n {#2} }
202     \exp_not:c { \l__cmd_function_tl \c_space_tl }
203     \exp_not:c
204     {
205       \l__cmd_function_tl \c_space_tl
206       \bool_if:NT \l__cmd_some_short_bool
207       { \bool_if:NT \l__cmd_some_long_bool { \c_space_tl } }

```

```

208     }
209     \exp_not:c { \l__cmd_function_tl \c_space_tl code }
210     \bool_if:NTF \l__cmd_defaults_bool
211     { \exp_not:c { \l__cmd_function_tl \c_space_tl defaults } }
212     { ? }
213     { \exp_not:o \l__cmd_signature_tl }
214 }
215 \bool_if:NTF \l__cmd_some_long_bool
216 {
217     \bool_if:NT \l__cmd_some_short_bool
218     {
219         \cs_set_nopar:cpx { \l__cmd_function_tl \c_space_tl \c_space_tl }
220         ##1##2 { ##1 {##2} }
221     }
222     \cs_set:cpx
223 }
224 { \cs_set_nopar:cpx }
225     { \l__cmd_function_tl \c_space_tl } ##1##2 { ##1 {##2} }
226 }

```

(End of definition for __cmd_declare_cmd_code:Nnn and others.)

```

\__cmd_declare_env:nnnn
\__cmd_declare_env:ennn
  \__cmd_declare_env_internal:nnnn
\__cmd_set_environment_end:n

```

The lead-off to creating an environment is much the same as that for creating a command: issue the appropriate message, store the argument specification then hand off to an internal function.

```

227 <latexrelease>\IncludeInRelease{2024/11/01}{\__cmd_declare_env:nnnn}%
228 <latexrelease>           {Use~space~trimmed~envname~directly}
229 \cs_new_protected:Npn \__cmd_declare_env:nnnn #1#2
230 {
231     \str_set:Nn \l__cmd_environment_str {#1}
232     \cs_if_exist:cTF { #1 }
233     { \msg_info:nnnn { cmd } { redefine-env } { #1 } { #2 } }
234     { \msg_info:nnnn { cmd } { define-env } { #1 } { #2 } }
235     \bool_set_false:N \l__cmd_expandable_bool
236     \bool_set_true:N \l__cmd_environment_bool
237     \__cmd_declare_env_internal:nnnn {#1} {#2}
238 }
239 \cs_generate_variant:Nn \__cmd_declare_env:nnnn { e }
240 <latexrelease>\EndIncludeInRelease
241 <latexrelease>\IncludeInRelease{2024/06/01}{\__cmd_declare_env:nnnn}%
242 <latexrelease>           {Use~space~trimmed~envname~directly}
243 <latexrelease>\cs_new_protected:Npn \__cmd_declare_env:nnnn #1#2
244 <latexrelease> {
245 <latexrelease>     \str_set:Nx \l__cmd_environment_str {#1}
246 <latexrelease>     \str_set:Nx \l__cmd_environment_str
247 <latexrelease>       { \tl_trim_spaces:o { \l__cmd_environment_str } }
248 <latexrelease>     \cs_if_exist:cTF { \l__cmd_environment_str }
249 <latexrelease>       {
250 <latexrelease>         \msg_info:nnxx { cmd } { redefine-env }
251 <latexrelease>           { \l__cmd_environment_str } { \tl_to_str:n {#2} }
252 <latexrelease>       }
253 <latexrelease>       {
254 <latexrelease>         \msg_info:nnxx { cmd } { define-env }
255 <latexrelease>           { \l__cmd_environment_str } { \tl_to_str:n {#2} }

```

```

256 <latexrelease>    }
257 <latexrelease>    \bool_set_false:N \l__cmd_expandable_bool
258 <latexrelease>    \bool_set_true:N \l__cmd_environment_bool
259 <latexrelease>    \exp_args:NV \__cmd_declare_env_internal:nmmm
260 <latexrelease>    \l__cmd_environment_str {#2}
261 <latexrelease> }
262 <latexrelease>
263 <latexrelease> \EndIncludeInRelease

```

Creating a document environment requires a few more steps than creating a single command. In order to pass the arguments of the command to the end of the function, it is necessary to store the grabbed arguments. To do that, the function used at the end of the environment has to be redefined to contain the appropriate information. To minimize the amount of expansion at point of use, the code here is expanded now as well as when used. The last argument of `__cmd_declare_cmd_internal:Nmmn` is only run if the definition succeeded. In package mode this ensures that the original definition of the environment is not changed if the definition fails for any reason. This also avoids an error when defining the `end_au_x_` function when the user asks for more than 9 arguments.

```

264 \cs_new_protected:Npn \__cmd_declare_env_internal:nmmm #1#2#3#4
265 {
266   \exp_args:Nc \__cmd_declare_cmd_internal:Nmmn { environment~ #1 } {#2}
267   {#3}
268   {
269     \cs_set_nopar:cpx { environment~ #1 ~end }
270     { \exp_not:c { environment~ #1 ~end~aux } }
271     \cs_generate_from_arg_count:cNmm
272     { environment~ #1 ~end~aux~ } \cs_set:Npn
273     \l__cmd_current_arg_int {#4}
274     \cs_set_eq:cc {#1} { environment~ #1 }
275     \cs_set_eq:cc { end #1 } { environment~ #1 ~end }
276   }
277 }
278 \cs_new_protected:Npn \__cmd_set_environment_end:n #1
279 {
280   \cs_set_nopar:cpx { environment~ #1 ~end~aux }
281   {
282     \exp_not:c { environment~ #1 ~end~aux~ }
283     \exp_not:o \l__cmd_args_tl
284   }
285 }

```

(End of definition for `__cmd_declare_env:nmmn`, `__cmd_declare_env_internal:nmmm`, and `__cmd_set_environment_end:n`.)

1.3 Structure of `xparse` commands

`__cmd_start_env:nmmmm`
`__cmd_start:nNNmmn`

For error messages that occur during run-time when getting arguments of environments it is necessary to keep track of the environment name. We begin non-expandable commands with a token equal to `\scan_stop:`, whose name gives a reasonable error message if the command is used inside a csname and protects against f-expansion. This is useless for environments since `\begin` is already not expandable. Both the command and environment codes start with `\group_align_safe_begin:`, then `__cmd_run_code:` (used

by both) does `\group_align_safe_end:`, so that delimited arguments may be grabbed in alignments if they contain an alignment tab token (see [latex3/latex3/issues/839](https://www.latex3.org/latex3/issues/839)).

```

286 \cs_new_protected:Npn \__cmd_start_env:nnnnn #1#2
287 {
288   \conditionally@traceoff
289   \group_align_safe_begin:
290   \str_set:Nn \l__cmd_environment_str {#2}
291   \bool_set_true:N \l__cmd_environment_bool
292   \__cmd_start_aux:ccnnnn
293   { environment~ \l__cmd_environment_str \c_space_tl }
294   { environment~ \l__cmd_environment_str \c_space_tl code }
295   {#1}
296 }
297 \cs_new_protected:Npx \__cmd_start:nNNnnn #1#2#3
298 {
299   \exp_not:c { xparse~function~is~not~expandable }
300   \exp_not:N \conditionally@traceoff
301   \exp_not:N \group_align_safe_begin:
302   \exp_not:n { \bool_set_false:N \l__cmd_environment_bool }
303   \exp_not:N \__cmd_start_aux:NNnnnn
304   #2 #3 {#1}
305 }

```

(End of definition for `__cmd_start_env:nnnnn` and `__cmd_start:nNNnnn`.)

`__cmd_start_aux:NNnnnn`
`__cmd_start_aux:ccnnnn`

This sets up a few variables to minimize the boilerplate code included in all `xparse`-defined commands. It then runs the grabbers #4. Again, the argument specification #1 is only for diagnostics.

```

306 \cs_new_protected:Npn \__cmd_start_aux:NNnnnn #1#2#3#4#5#6
307 {
308   \tl_clear:N \l__cmd_args_tl
309   \tl_set:Nn \l__cmd_fn_tl {#1}
310   \tl_set:Nn \l__cmd_fn_code_tl {#2}
311   \tl_set:Nn \l__cmd_defaults_tl {#5}
312   \tl_set:Nn \l__cmd_process_all_tl {#6}
313   #4
314   \__cmd_run_code:
315 }
316 \cs_generate_variant:Nn \__cmd_start_aux:NNnnnn { cc }

```

(End of definition for `__cmd_start_aux:NNnnnn`.)

`__cmd_run_code:`

After arguments are grabbed, this function is responsible for inserting default values, running processors, and finally doing `\group_align_safe_end:` as promised, and running the code.

```

317 \cs_new_protected:Npn \__cmd_run_code:
318 {
319   \tl_if_empty:NF \l__cmd_defaults_tl { \__cmd_defaults: }
320   \tl_if_empty:NF \l__cmd_process_all_tl { \__cmd_args_process: }
321   \bool_if:NT \l__cmd_environment_bool
322   { \exp_args:No \__cmd_set_environment_end:n \l__cmd_environment_str }
323   \group_align_safe_end:
324   \conditionally@traceon
325   \exp_after:wN \l__cmd_fn_code_tl \l__cmd_args_tl
326 }

```

(End of definition for `__cmd_run_code:`.)

`__cmd_defaults:` First construct `__cmd_tmp:w` (see below) that will receive the arguments found so far and determine default values for any missing argument. Then call it repeatedly until the set of arguments stabilizes. Since that could lead to an infinite loop we only call it up to nine times, the maximal number needed for stabilization if there is a chain of arguments that depend on each other. If that fails to stabilize raise an error.

```

\__cmd_defaults_error:w 327 \cs_new_protected:Npn \__cmd_defaults:
                        {
328   \__cmd_defaults_def:
329   \tl_set_eq:NN \l__cmd_args_i_tl \l__cmd_args_tl
330   \__cmd_defaults_aux: \__cmd_defaults_aux: \__cmd_defaults_aux:
331   \__cmd_defaults_aux: \__cmd_defaults_aux: \__cmd_defaults_aux:
332   \__cmd_defaults_aux: \__cmd_defaults_aux: \__cmd_defaults_aux:
333   \__cmd_defaults_error:w
334   \q_recursion_stop
335   \tl_set_eq:NN \l__cmd_args_tl \l__cmd_args_i_tl
336 }
337 \cs_new_protected:Npn \__cmd_defaults_aux:
                        {
338   \tl_set:Nx \l__cmd_args_ii_tl
339   { \exp_after:wN \__cmd_tmp:w \l__cmd_args_i_tl }
340   \tl_if_eq:NNT \l__cmd_args_ii_tl \l__cmd_args_i_tl
341   { \use_none_delimit_by_q_recursion_stop:w }
342   \tl_set_eq:NN \l__cmd_args_i_tl \l__cmd_args_ii_tl
343 }
344 \cs_new_protected:Npn \__cmd_defaults_error:w \q_recursion_stop
345 {
346   \msg_error:nxx { cmd } { default-loop }
347   { \__cmd_environment_or_command: }
348 }
349
350

```

To construct `__cmd_tmp:w`, first go through the arguments found and the corresponding defaults, building a token list with `{#⟨arg number⟩}` for arguments found in the input (whose default will not be used) and otherwise `{\exp_not:n{⟨default⟩}}` for arguments whose default will be used.

```

351 \cs_new_protected:Npn \__cmd_defaults_def:
352 {
353   \tl_clear:N \l__cmd_tmpa_tl
354   \int_zero:N \l__cmd_current_arg_int
355   \__cmd_tl_mapthread_function:NNN \l__cmd_args_tl \l__cmd_defaults_tl
356   \__cmd_defaults_def:nn
357   \cs_generate_from_arg_count:NNno \__cmd_tmp:w \cs_set:Npn
358   \l__cmd_current_arg_int \l__cmd_tmpa_tl
359 }
360 \cs_generate_variant:Nn \cs_generate_from_arg_count:NNnn { NNno }
361 \cs_new_protected:Npn \__cmd_defaults_def:nn
362 {
363   \int_incr:N \l__cmd_current_arg_int
364   \exp_args:NV \__cmd_defaults_def:nnn \l__cmd_current_arg_int
365 }
366 \cs_new_protected:Npn \__cmd_defaults_def:nnn #1#2#3
367 {
368   \tl_put_right:Nx \l__cmd_tmpa_tl

```

```

369     {
370     {
371         \exp_not:N \exp_not:n
372         {
373             \tl_if_novalue:nTF {#2}
374             { \exp_not:o {#3} }
375             { \exp_not:n { ## #1 } }
376         }
377     }
378 }
379 }

```

(End of definition for `__cmd_defaults:` and others.)

`__cmd_args_process:` Loop through arguments (stored in `\l__cmd_args_tl`) and the corresponding processors (in `\l__cmd_process_all_tl`) simultaneously, apply all processors for each argument and store the result back into `\l__cmd_args_tl`. To allow processors to depend on other arguments, for every processor define a temporary auxiliary that receives all arguments `\l__cmd_args_tl`.

```

380 \cs_new_protected:Npn \__cmd_args_process:
381 {
382     \tl_clear:N \l__cmd_args_ii_tl
383     \__cmd_tl_mapthread_function:NNN
384     \l__cmd_args_tl
385     \l__cmd_process_all_tl
386     \__cmd_args_process_loop:nn
387     \tl_set_eq:NN \l__cmd_args_tl \l__cmd_args_ii_tl
388 }
389 \cs_new_protected:Npn \__cmd_args_process_loop:nn #1#2
390 {
391     \tl_set:Nn \ProcessedArgument {#1}
392     \tl_if_novalue:nF {#1}
393     { \tl_map_function:nN {#2} \__cmd_args_process_aux:n }
394     \tl_put_right:No \l__cmd_args_ii_tl
395     { \exp_after:wN { \ProcessedArgument } }
396 }
397 \cs_new_protected:Npn \__cmd_args_process_aux:n #1
398 {
399     \cs_generate_from_arg_count:NNnn \__cmd_tmp:w \cs_set:Npn
400     { \tl_count:N \l__cmd_args_tl } {#1}
401     \exp_args:NNNo \exp_after:wN \__cmd_tmp:w \l__cmd_args_tl
402     { \ProcessedArgument }
403 }

```

(End of definition for `__cmd_args_process:`, `__cmd_args_process_loop:nn`, and `__cmd_args_process_aux:n`.)

`__cmd_start_expandable:nNNNNn` This is called for all expandable commands. #6 is the signature, responsible for grabbing arguments. #5 is used to determine default values (or is ? if there are none). #4 is the code to run. #2 and #3 are functions (named after the command) that grab a single argument in the input stream (#3 is short). The argument specification #1 is only used by diagnostic functions. Same as for the non-expandable version, this starts with `\group_align_safe_begin:`, which expands to nothing, so may be safely used in an expandable context.

```

404 \cs_new:Npn \__cmd_start_expandable:nNNNNn #1#2#3#4#5#6
405   {
406     \group_align_safe_begin:
407     #6 \__cmd_end_expandable:NNw #5 #4 \q__cmd #2#3
408   }

```

(End of definition for __cmd_start_expandable:nNNNNn.)

```

\__cmd_end_expandable:NNw
\__cmd_end_expandable_aux:w
  \__cmd_end_expandable_aux:nNNNN
\__cmd_end_expandable_defaults:nnnNNn
  \__cmd_end_expandable_defaults:nnw
  \__cmd_end_expandable_defaults:nw

```

Followed by a function #1 to determine default values (or ? if there are no defaults), the code #2, arguments that have been grabbed, then \q__cmd and two generic grabbers. The idea to find default values is similar to the non-expandable case but we cannot define an auxiliary function, so at every step in the loop we need to go through all arguments searching for which ones started out as -NoValue- and replacing these by the newly computed values. In fact we need to keep track of three versions of all arguments: the original version, the previous version with default values, and the currently built version (first argument of __cmd_end_expandable_defaults:nnnNNn).

```

409 \cs_new:Npn \__cmd_end_expandable:NNw #1#2
410   { \__cmd_end_expandable_aux:w #1#2 \prg_do_nothing: }
411 \cs_new:Npn \__cmd_end_expandable_aux:w #1#2#3 \q__cmd
412   { \exp_args:No \__cmd_end_expandable_aux:nNNNN {#3} #1 #2 }
413 \cs_new:Npn \__cmd_end_expandable_aux:nNNNN #1#2#3#4#5
414   {
415     \token_if_eq_charcode:NNT ? #2 { \exp_after:wN \use_iv:nnnn }
416     \__cmd_end_expandable_defaults:nnnNNn {#1} { } {#1} #2#3
417     { } { } { } { } { } { } { } { } { } { } { } { }
418     {
419       \msg_expandable_error:nmf { cmd } { default-loop }
420       { \exp_args:Nf \tl_trim_spaces:n { \token_to_str:N #4 } }
421       \use_iv:nnnn
422     }
423     \q_stop
424   }
425 \cs_new:Npn \__cmd_end_expandable_defaults:nnnNNn #1#2#3#4#5#6
426   {
427     #6
428     \str_if_eq:nnTF {#1} {#2}
429     { \use_i_delimit_by_q_stop:nw { \group_align_safe_end: #5 #1 } }
430     {
431       \exp_args:No \__cmd_tl_mapthread_function:nnN
432       { #4 #1 } {#3}
433       \__cmd_end_expandable_defaults:nnw
434       \__cmd_end_expandable_defaults:nnnNNn { } {#1} {#3} #4 #5
435     }
436   }
437 \cs_new:Npn \__cmd_end_expandable_defaults:nnw #1#2
438   {
439     \tl_if_novalue:nTF {#2}
440     { \exp_args:No \__cmd_end_expandable_defaults:nw {#1} }
441     { \__cmd_end_expandable_defaults:nw {#2} }
442   }
443 \cs_new:Npn \__cmd_end_expandable_defaults:nw
444   #1#2 \__cmd_end_expandable_defaults:nnnNNn #3
445   { #2 \__cmd_end_expandable_defaults:nnnNNn { #3 {#1} } }

```

(End of definition for __cmd_end_expandable:NNw and others.)

1.4 Normalizing the argument specifications

The goal here is to expand aliases and check that the argument specification is valid before the main parsing run. If it is not valid the entire set up is abandoned to avoid any strange internal errors. A function is provided for each argument type that will grab any extra data items and call the loop function after performing the following checks and tasks.

- Check that each argument has the correct number of data items associated with it, and that where a single character is required, one has actually been supplied.
- Check that processors and the markers +, ! and = are followed by an argument for which they make sense, and are not redundant.
- Check the absence of forbidden types for expandable commands, namely G/v always, and l/u after optional arguments (xparse may have inserted braces due to a failed search for an optional argument).
- Check that no optional argument is followed by a mandatory argument with the same delimiter, as otherwise the optional argument could never be omitted.
- Keep track in `\l__cmd_some_long_bool` and `\l__cmd_some_short_bool` of whether the command has some long/short arguments.
- Keep track in `\l__cmd_grab_expandably_bool` of whether all arguments are m/l/u type and short arguments appear before long ones, in which case they can be grabbed expandably just as safely as they could be grabbed nonexpandably. Regardless of that, arguments of expandable commands will be grabbed expandably and arguments of environments will not (because the list of arguments built by non-expandable grabbing is used to pass them to the end-environment code).

Further checks happen at the end of the loop:

- that there are at most 9 arguments;
- that an expandable command does not end with an optional argument (this case is detected by using the fact that `\l__cmd_last_delimiters_tl` is cleared by every mandatory argument and filled by every optional argument).

`__cmd_normalize_arg_spec:n` Loop through the argument specification, calling an auxiliary specific to each argument type. If any argument is unknown stop the definition.

`__cmd_normalize_arg_spec_loop:n`

```
446 \cs_new_protected:Npn \__cmd_normalize_arg_spec:n #1
447   {
448     \int_zero:N \l__cmd_current_arg_int
449     \tl_clear:N \l__cmd_last_delimiters_tl
450     \tl_clear:N \l__cmd_arg_spec_tl
451     \bool_set_true:N \l__cmd_grab_expandably_bool
452     \bool_set_false:N \l__cmd_obey_spaces_bool
453     \bool_set_false:N \l__cmd_long_bool
454     \bool_set_false:N \l__cmd_suppress_strip_bool
455     \bool_set_false:N \l__cmd_some_obey_spaces_bool
456     \bool_set_false:N \l__cmd_some_long_bool
457     \bool_set_false:N \l__cmd_some_short_bool
458     \__cmd_normalize_arg_spec_loop:n #1
459     \q_recursion_tail \q_recursion_tail \q_recursion_stop
```

```

460 \int_compare:nNnT \l__cmd_current_arg_int > 9
461 {
462   \msg_error:nxxx { cmd } { too-many-args }
463   { \__cmd_environment_or_command: } { \tl_to_str:n {#1} }
464   \__cmd_bad_def:wn
465 }
466 \bool_if:NT \l__cmd_expandable_bool
467 {
468   \tl_if_empty:NF \l__cmd_last_delimiters_tl
469   {
470     \msg_error:nxxx { cmd } { expandable-ending-optional }
471     { \iow_char:N \ \ \l__cmd_function_tl } { \tl_to_str:n {#1} }
472     \__cmd_bad_def:wn
473   }
474 }
475 \bool_if:NT \l__cmd_expandable_bool
476 { \bool_set_true:N \l__cmd_grab_expandably_bool }
477 \bool_if:NT \l__cmd_environment_bool
478 { \bool_set_false:N \l__cmd_grab_expandably_bool }
479 }
480 \cs_new_protected:Npn \__cmd_normalize_arg_spec_loop:n #1
481 {
482   \quark_if_recursion_tail_stop:n {#1}
483   \int_incr:N \l__cmd_current_arg_int
484   \cs_if_exist_use:cF { __cmd_normalize_type_ \tl_to_str:n {#1} :w }
485   {
486     \bool_lazy_any:nTF
487     {
488       { \str_if_eq_p:nn {#1} { G } }
489       { \str_if_eq_p:nn {#1} { g } }
490       { \str_if_eq_p:nn {#1} { l } }
491       { \str_if_eq_p:nn {#1} { u } }
492     }
493     {
494       \msg_error:nxxx { cmd } { xparse-arg-type }
495       { \__cmd_environment_or_command: } { \tl_to_str:n {#1} }
496     }
497     {
498       \msg_error:nxxx { cmd } { unknown-argument-type }
499       { \__cmd_environment_or_command: } { \tl_to_str:n {#1} }
500     }
501     \__cmd_bad_def:wn
502   }
503 }

```

(End of definition for __cmd_normalize_arg_spec:n and __cmd_normalize_arg_spec_loop:n.)

```

\__cmd_normalize_type_d:w These argument types are aliases of more general ones, for example with the default
\__cmd_normalize_type_e:w argument -NoValue-. To easily insert that marker expanded in the definitions we call
\__cmd_normalize_type_o:w \__cmd_tmp:w with the argument -NoValue-. For argument types that need additional
\__cmd_normalize_type_0:w data, check that the data is present (not \q_recursion_tail) before proceeding.
\__cmd_normalize_type_r:w 504 \cs_set_protected:Npn \__cmd_tmp:w #1
\__cmd_normalize_type_s:w 505 {
506   \cs_new_protected:Npn \__cmd_normalize_type_d:w ##1##2

```

```

507     {
508     \quark_if_recursion_tail_stop_do:nn {##2} { \__cmd_bad_arg_spec:wn }
509     \__cmd_normalize_type_D:w {##1} {##2} {#1}
510     }
511 \cs_new_protected:Npn \__cmd_normalize_type_e:w ##1
512 {
513     \quark_if_recursion_tail_stop_do:nn {##1} { \__cmd_bad_arg_spec:wn }
514     \__cmd_normalize_type_E:w {##1} { }
515 }
516 \cs_new_protected:Npn \__cmd_normalize_type_o:w
517 { \__cmd_normalize_type_D:w [ ] {#1} }
518 \cs_new_protected:Npn \__cmd_normalize_type_O:w
519 { \__cmd_normalize_type_D:w [ ] }
520 \cs_new_protected:Npn \__cmd_normalize_type_r:w ##1##2
521 {
522     \quark_if_recursion_tail_stop_do:nn {##2} { \__cmd_bad_arg_spec:wn }
523     \__cmd_normalize_type_R:w {##1} {##2} {#1}
524 }
525 \cs_new_protected:Npn \__cmd_normalize_type_s:w
526 { \__cmd_normalize_type_t:w * }
527 }
528 \exp_args:No \__cmd_tmp:w { \c_novalue_tl }

```

(End of definition for __cmd_normalize_type_d:w and others.)

__cmd_normalize_type_>:w Check that these prefixes have arguments, namely that the next token is not \q-recursion_tail, and remember to leave it after the looping macro. Processors are forbidden in expandable commands. If all is good, store the prefix in the cleaned up \l__cmd_arg_spec_tl, and decrement the argument number as prefixes do not correspond to arguments.

```

529 \cs_new_protected:cpn { __cmd_normalize_type_>:w } #1#2
530 {
531     \quark_if_recursion_tail_stop_do:nn {#2} { \__cmd_bad_arg_spec:wn }
532     \bool_if:NT \l__cmd_expandable_bool
533     {
534         \msg_error:nxxx { cmd } { processor-in-expandable }
535         { \iow_char:N \ \ \l__cmd_function_tl } { \tl_to_str:n {#1} }
536         \__cmd_bad_def:wn
537     }
538     \tl_put_right:Nx \l__cmd_arg_spec_tl { > { \tl_trim_spaces:n {#1} } }
539     \int_decr:N \l__cmd_current_arg_int
540     \bool_set_false:N \l__cmd_grab_expandably_bool
541     \__cmd_normalize_arg_spec_loop:n {#2}
542 }
543 \cs_new_protected:cpn { __cmd_normalize_type_+:w } #1
544 {
545     \__cmd_normalize_type_aux:NnNn + {#1}
546     \l__cmd_long_bool
547     { \bool_set_true:N \l__cmd_long_bool }
548 }
549 \cs_new_protected:cpn { __cmd_normalize_type_!:w } #1
550 {
551     \__cmd_normalize_type_aux:NnNn ! {#1}
552     \l__cmd_obey_spaces_bool

```

```

553     {
554         \bool_set_true:N \l__cmd_obey_spaces_bool
555         \bool_set_true:N \l__cmd_some_obey_spaces_bool
556     }
557 }
558 \cs_new_protected:cpn { __cmd_normalize_type_=:w } #1#2
559 {
560     \__cmd_normalize_type_aux:NnNn = {#2}
561     \l__cmd_suppress_strip_bool
562     {
563         \bool_if:NT \l__cmd_expandable_bool
564         {
565             \msg_error:nxxx { cmd } { keyval-in-expandable }
566             { \iow_char:N \ \ \l__cmd_function_tl } { \tl_to_str:n {#1} }
567             \__cmd_bad_def:wn
568         }
569         \bool_set_true:N \l__cmd_suppress_strip_bool
570         \bool_set_false:N \l__cmd_grab_expandably_bool
571         \tl_put_right:Nx \l__cmd_arg_spec_tl
572         { = { \tl_trim_spaces:n {#1} } }
573     }
574 }
575 \cs_new_protected:Npn \__cmd_normalize_type_aux:NnNn #1#2#3#4
576 {
577     \quark_if_recursion_tail_stop_do:nn {#2} { \__cmd_bad_arg_spec:wn }
578     \bool_if:NT #3
579     {
580         \msg_error:nxxx { cmd } { two-markers }
581         { \__cmd_environment_or_command: } { #1 }
582         \__cmd_bad_def:wn
583     }
584     #4
585     \int_decr:N \l__cmd_current_arg_int
586     \__cmd_normalize_arg_spec_loop:n {#2}
587 }

```

(End of definition for __cmd_normalize_type_>:w and others.)

__cmd_normalize_type_D:w
 __cmd_normalize_type_E:w
 __cmd_normalize_type_t:w
 __cmd_normalize_E_unique_check:w

Optional argument types. Check that all required data is present (and consists of single characters if applicable) and check for forbidden types for expandable commands. For E-type require that there is at least one embellishment, that each one is a single character, and that there aren't more optional arguments than embellishments; also remember that each embellishment counts as one argument for \l__cmd_current_arg_int. Then in each case store the data in \l__cmd_arg_spec_tl, and for later checks store in \l__cmd_last_delimiters_tl the tokens whose presence determines whether there is an optional argument (for braces store {}, seen later as an empty delimiter).

```

588 \cs_new_protected:Npn \__cmd_normalize_type_D:w #1#2#3
589 {
590     \quark_if_recursion_tail_stop_do:nn {#3} { \__cmd_bad_arg_spec:wn }
591     \__cmd_single_token_check:n {#1} \__cmd_allowed_token_check:N #1
592     \__cmd_single_token_check:n {#2}
593     \__cmd_add_arg_spec:n { D #1 #2 {#3} }
594     \tl_put_right:Nn \l__cmd_last_delimiters_tl {#1}
595     \bool_set_false:N \l__cmd_grab_expandably_bool

```

```

596   \__cmd_normalize_arg_spec_loop:n
597 }
598 \cs_new_protected:Npn \__cmd_normalize_type_E:w #1#2
599 {
600   \quark_if_recursion_tail_stop_do:nn {#2} { \__cmd_bad_arg_spec:wn }
601   \tl_if_blank:nT {#1} { \__cmd_bad_arg_spec:wn }
602   \tl_map_function:nN {#1} \__cmd_single_token_check:n
603   \tl_map_function:nN {#1} \__cmd_allowed_token_check:N
604   \__cmd_normalize_E_unique_check:w #1 \q_nil \q_stop
605   \int_compare:nNnT { \tl_count:n {#2} } > { \tl_count:n {#1} }
606     { \__cmd_bad_arg_spec:wn }
607   \__cmd_add_arg_spec:n { E {#1} {#2} }
608   \tl_put_right:Nn \l__cmd_last_delimiters_tl {#1}
609   \bool_set_false:N \l__cmd_grab_expandably_bool
610   \int_add:Nn \l__cmd_current_arg_int { \tl_count:n {#1} - 1 }
611   \__cmd_normalize_arg_spec_loop:n
612 }
613 \cs_new_protected:Npn \__cmd_normalize_E_unique_check:w #1#2 \q_stop
614 {
615   \quark_if_nil:NF #1
616   {
617     \tl_if_in:nnT {#2} {#1} { \__cmd_bad_arg_spec:wn }
618     \__cmd_normalize_E_unique_check:w #2 \q_stop
619   }
620 }
621 \cs_new_protected:Npn \__cmd_normalize_type_t:w #1
622 {
623   \quark_if_recursion_tail_stop_do:Nn #1 { \__cmd_bad_arg_spec:wn }
624   \__cmd_single_token_check:n {#1} \__cmd_allowed_token_check:N #1
625   \tl_put_right:Nx \l__cmd_arg_spec_tl
626     {
627       \bool_if:NT \l__cmd_obey_spaces_bool { ! }
628       t \exp_not:n {#1}
629     }
630   \tl_put_right:Nn \l__cmd_last_delimiters_tl {#1}
631   \bool_set_false:N \l__cmd_grab_expandably_bool
632   \bool_set_false:N \l__cmd_obey_spaces_bool
633   \bool_set_false:N \l__cmd_long_bool
634   \__cmd_normalize_arg_spec_loop:n
635 }

```

(End of definition for __cmd_normalize_type_D:w and others.)

__cmd_normalize_type_m:w Mandatory arguments. First check the required data is present, consists of single characters where applicable, and that the argument type is allowed for expandable commands if applicable. For the m and R argument types check that they do not follow some optional argument with that delimiter as otherwise the optional argument could not be omitted. Then save data in \l__cmd_arg_spec_tl, count the mandatory argument, and empty the list of last delimiters.

```

636 \cs_new_protected:Npn \__cmd_normalize_type_m:w
637 {
638   \__cmd_delimiter_check:nnn { } { m } { \iow_char:N \{ }
639   \__cmd_add_arg_spec_mandatory:n { m }
640   \__cmd_normalize_arg_spec_loop:n

```

```

641 }
642 \cs_new_protected:Npn \__cmd_normalize_type_R:w #1#2#3
643 {
644   \quark_if_recursion_tail_stop_do:nn {#3} { \__cmd_bad_arg_spec:wn }
645   \__cmd_single_token_check:n {#1} \__cmd_allowed_token_check:N #1
646   \__cmd_single_token_check:n {#2}
647   \__cmd_delimiter_check:nnn {#1} { R/r } { \tl_to_str:n {#1} }
648   \bool_set_false:N \l__cmd_grab_expandably_bool
649   \__cmd_add_arg_spec_mandatory:n { R #1 #2 {#3} }
650   \__cmd_normalize_arg_spec_loop:n
651 }
652 \cs_new_protected:Npn \__cmd_normalize_type_v:w
653 {
654   \__cmd_normalize_check_gv:N v
655   \__cmd_add_arg_spec_mandatory:n { v }
656   \__cmd_normalize_arg_spec_loop:n
657 }

```

(End of definition for __cmd_normalize_type_m:w, __cmd_normalize_type_R:w, and __cmd_normalize_type_v:w.)

__cmd_normalize_type_b:w These argument types are not allowed for commands. They are only allowed at the end of the argument specification, hence we check that #1 is the end.

__cmd_normalize_type_c:w

__cmd_normalize_type_b_or_c:nn

```

658 \cs_new_protected:Npn \__cmd_normalize_type_b:w #1
659 { \__cmd_normalize_type_b_or_c:nn {#1} { b } }
660 \cs_new_protected:Npn \__cmd_normalize_type_c:w #1
661 { \__cmd_normalize_type_b_or_c:nn {#1} { c } }
662 \cs_new_protected:Npn \__cmd_normalize_type_b_or_c:nn #1#2
663 {
664   \bool_if:NF \l__cmd_environment_bool
665   {
666     \msg_error:nnxx { cmd } { invalid-command-arg }
667     { \__cmd_environment_or_command: } {#2}
668     \__cmd_bad_def:wn
669   }
670   \tl_clear:N \l__cmd_last_delimiters_tl
671   \__cmd_add_arg_spec:n {#2}
672   \quark_if_recursion_tail_stop:n {#1}
673   \msg_error:nnxxx { cmd } { arg-after-body }
674   {#2}
675   { \__cmd_environment_or_command: }
676   { \tl_to_str:n {#1} }
677   \__cmd_bad_def:wn
678 }

```

(End of definition for __cmd_normalize_type_b:w, __cmd_normalize_type_c:w, and __cmd_normalize_type_b_or_c:nn.)

__cmd_single_token_check:n

Checks that the argument is a single (non-space) token (possibly surrounded by spaces), and aborts the definition otherwise.

```

679 \cs_new_protected:Npn \__cmd_single_token_check:n #1
680 {
681   \tl_trim_spaces_apply:nN {#1} \tl_if_single_token:nF
682   {
683     \msg_error:nnxx { cmd } { not-single-token }

```

```

684         { \_cmd_environment_or_command: } { \tl_to_str:n {#1} }
685     \_cmd_bad_def:wn
686 }
687 }

```

(End of definition for _cmd_single_token_check:n.)

_cmd_allowed_token_check:N Some tokens are not allowed as delimiters for some argument types, notably implicit begin/end-group tokens (\bgroup/\egroup). The major problem with these tokens is that for \peek... functions, a literal {₁ is virtually indistinguishable from a \bgroup or other token which was \let to a {₁, and the same goes for }₂. All other tokens can be easily distinguished from their implicit counterparts by grabbing them and looking at the string length (see _cmd_token_if_cs:NTF), but for begin/end group tokens that is not possible without the risk of mistakenly grabbing the entire brace group (potentially leading to a ! Runaway argument error) or trying to grab a }₂, leading to an ! Argument of \dots has an extra } error.

```

688 \cs_new_protected:Npn \_cmd_allowed_token_check:N #1
689 {
690     \token_if_eq_meaning:NNTF #1 \c_group_begin_token
691     { \use:n }
692     {
693         \token_if_eq_meaning:NNTF #1 \c_group_end_token
694         { \use:n }
695         { \use_none:n }
696     }
697     {
698         \msg_error:nxxx { cmd } { forbidden-group-token }
699         { \_cmd_environment_or_command: } { \tl_to_str:n {#1} }
700         {
701             \token_if_eq_meaning:NNTF #1 \c_group_begin_token
702             { begin } { end }
703         }
704         \_cmd_bad_def:wn
705     }
706 }

```

(End of definition for _cmd_allowed_token_check:N.)

_cmd_normalize_check_gv:N Called for arguments that are always forbidden, or forbidden after an optional argument,
_cmd_normalize_check_lu:N for expandable commands.

```

707 \cs_new_protected:Npn \_cmd_normalize_check_gv:N #1
708 {
709     \bool_if:NT \l__cmd_expandable_bool
710     {
711         \msg_error:nxxx { cmd } { invalid-expandable-arg }
712         { \iow_char:N \ \l__cmd_function_tl } { \tl_to_str:n {#1} }
713         \_cmd_bad_def:wn
714     }
715     \bool_set_false:N \l__cmd_grab_expandably_bool
716 }
717 \cs_new_protected:Npn \_cmd_normalize_check_lu:N #1
718 {
719     \bool_if:NT \l__cmd_expandable_bool
720     {

```

```

721     \tl_if_empty:NF \l__cmd_last_delimiters_tl
722     {
723       \msg_error:nxxx { cmd } { invalid-after-optional-expandably }
724       { \iow_char:N \ \l__cmd_function_tl } { \tl_to_str:n {#1} }
725       \__cmd_bad_def:wn
726     }
727   }
728 }

```

(End of definition for __cmd_normalize_check_gv:N and __cmd_normalize_check_lu:N.)

`__cmd_delimiter_check:nnn` Called for `m` and `R` arguments. Checks that the leading token does not coincide with the token denoting the presence of a previous optional argument. Instead of dealing with braces for the `m`-type we use an empty delimiter to denote that case.

```

729 \cs_new_protected:Npn \__cmd_delimiter_check:nnn #1#2#3
730 {
731   \tl_map_inline:Nn \l__cmd_last_delimiters_tl
732   {
733     \tl_if_eq:nnT {##1} {#1}
734     {
735       \msg_warning:nxxx { cmd } { optional-mandatory }
736       {#2} {#3}
737     }
738   }
739 }

```

(End of definition for __cmd_delimiter_check:nnn.)

`__cmd_bad_arg_spec:wn` If the argument specification is wrong, this provides an escape from the entire definition process.

`__cmd_bad_def:wn`

```

740 \cs_new_protected:Npn \__cmd_bad_arg_spec:wn #1 \__cmd_break_point:n #2
741 {
742   \msg_error:nxxx { cmd } { bad-arg-spec }
743   { \__cmd_environment_or_command: } { \tl_to_str:n {#2} }
744 }
745 \cs_new_protected:Npn \__cmd_bad_def:wn #1 \__cmd_break_point:n #2 { }

```

(End of definition for __cmd_bad_arg_spec:wn and __cmd_bad_def:wn.)

`__cmd_add_arg_spec:n`

`__cmd_add_arg_spec_mandatory:n`

When adding an argument to the argument specification, set the `some_long` or `some_short` booleans as appropriate and clear the booleans keeping track of `+`, `!` and `=` markers. Before that, test for a short argument following some long arguments: this is forbidden for expandable commands and prevents grabbing arguments expandably.

For mandatory arguments do some more work, in particular complain if they were preceded by `!`.

```

746 \cs_new_protected:Npn \__cmd_add_arg_spec:n #1
747 {
748   \bool_lazy_and:nnT
749   { ! \l__cmd_long_bool }
750   { \l__cmd_some_long_bool }
751   {
752     \bool_if:NT \l__cmd_expandable_bool
753     {
754       \msg_error:nnx { cmd } { long-short-mix }

```



```

755         { \iow_char:N \ \ \l__cmd_function_tl }
756         \__cmd_bad_def:wn
757     }
758     \bool_set_false:N \l__cmd_grab_expandably_bool
759 }
760 \bool_if:NTF \l__cmd_long_bool
761 { \bool_set_true:N \l__cmd_some_long_bool }
762 { \bool_set_true:N \l__cmd_some_short_bool }
763 \tl_put_right:Nx \l__cmd_arg_spec_tl
764 {
765     \bool_lazy_and:nnT
766     { \l__cmd_long_bool }
767     { ! \str_if_eq_p:nn {#1} { c } }
768     { + }
769     \bool_if:NT \l__cmd_obey_spaces_bool { ! }
770     \exp_not:n {#1}
771 }
772 \bool_set_false:N \l__cmd_long_bool
773 \bool_set_false:N \l__cmd_obey_spaces_bool
774 }
775 \cs_new_protected:Npn \__cmd_add_arg_spec_mandatory:n #1
776 {
777     \bool_if:NT \l__cmd_some_obey_spaces_bool
778     {
779         \msg_error:nxxx { cmd } { invalid-bang }
780         { \__cmd_environment_or_command: }
781         {
782             \bool_if:NTF \l__cmd_obey_spaces_bool
783             { \tl_to_str:n {'#1'} }
784             { an-optional-argument-before-mandatory~ \tl_to_str:n {'#1'} }
785         }
786         \__cmd_bad_def:wn
787     }
788     \tl_clear:N \l__cmd_last_delimiters_tl
789     \__cmd_add_arg_spec:n {#1}
790 }

```

(End of definition for __cmd_add_arg_spec:n and __cmd_add_arg_spec_mandatory:n.)

1.5 Preparing the signature: general mechanism

Actually creating the signature uses the same loop approach as normalizing the signature. There are first a number of variables which need to be set to track what is going on. Many of these variables are unused when defining expandable commands.

```

\__cmd_prepare_signature:n
  \__cmd_prepare_signature_verb_chk:n
\__cmd_prepare_signature:N
  \__cmd_prepare_signature_bypass:N
791 \cs_new_protected:Npn \__cmd_prepare_signature:n #1
792 {
793     \int_set_eq:NN \l__cmd_total_args_int \l__cmd_current_arg_int
794     \int_zero:N \l__cmd_current_arg_int
795     \bool_set_false:N \l__cmd_long_bool
796     \bool_set_false:N \l__cmd_obey_spaces_bool
797     \bool_set_false:N \l__cmd_suppress_strip_bool
798     \int_zero:N \l__cmd_m_args_int
799     \bool_set_false:N \l__cmd_defaults_bool
800     \tl_clear:N \l__cmd_defaults_tl

```

```

801 \tl_clear:N \l__cmd_process_all_tl
802 \tl_clear:N \l__cmd_process_one_tl
803 \bool_set_false:N \l__cmd_process_some_bool
804 \tl_clear:N \l__cmd_signature_tl
805 \__cmd_prepare_signature_verb_chk:n {#1}
806 \__cmd_prepare_signature:N #1 \q_recursion_tail \q_recursion_stop
807 \bool_if:NF \l__cmd_expandable_bool { \__cmd_flush_m_args: }
808 }

```

A quick check on the final arg. type.

```

809 \cs_new_protected:Npn \__cmd_prepare_signature_verb_chk:n #1
810 {
811   \str_if_eq:eeTF { \tl_head:e { \tl_reverse:n {#1} } } { c }
812   { \bool_set_true:N \l__cmd_final_verb_bool }
813   { \bool_set_false:N \l__cmd_final_verb_bool }
814 }

```

The main looping function does not take an argument, but carries out the reset on the processor boolean. This is split off from the rest of the process so that when actually setting up processors the flag-reset can be bypassed.

For each known argument type there is an appropriate function to actually do the addition to the signature. These are separate for expandable and standard functions, as the approaches are different.

```

815 \cs_new_protected:Npn \__cmd_prepare_signature:N
816 {
817   \bool_set_false:N \l__cmd_prefixed_bool
818   \__cmd_prepare_signature_bypass:N
819 }
820 \cs_new_protected:Npn \__cmd_prepare_signature_bypass:N #1
821 {
822   \quark_if_recursion_tail_stop:N #1
823   \use:c
824   {
825     \__cmd_add
826     \bool_if:NT \l__cmd_grab_expandably_bool { _expandable }
827     _type_ \token_to_str:N #1 :w
828   }
829 }

```

(End of definition for __cmd_prepare_signature:n and others.)

1.6 Setting up a standard signature

Each argument-adding function appends to the signature a grabber (and for some types, the delimiters or default value), except the one for *m* arguments. These are collected and added to the signature all at once by `__cmd_flush_m_args:`, called for every other argument type. All of the functions then call the loop function `__cmd_prepare_signature:N`. Default values of arguments are collected by `__cmd_add_default:n` rather than being stored with the argument; this function and `__cmd_add_default:` are also responsible for keeping track of `\l__cmd_current_arg_int`.

`__cmd_add_type_+ :w` Making the next argument long means setting the flag. The *m* arguments are recorded here as this has to be done for every case where there is then a long argument.

```

830 \cs_new_protected:cpn { __cmd_add_type_+ :w }

```

```

831 {
832   \__cmd_flush_m_args:
833   \bool_set_true:N \l__cmd_long_bool
834   \bool_set_true:N \l__cmd_prefixed_bool
835   \__cmd_prepare_signature_bypass:N
836 }

```

(End of definition for __cmd_add_type_+ :w.)

__cmd_add_type_! :w Much the same for controlling trailing optional arguments.

```

837 \cs_new_protected:cpn { __cmd_add_type_! :w }
838 {
839   \__cmd_flush_m_args:
840   \bool_set_true:N \l__cmd_obey_spaces_bool
841   \bool_set_true:N \l__cmd_prefixed_bool
842   \__cmd_prepare_signature_bypass:N
843 }

```

(End of definition for __cmd_add_type_! :w.)

__cmd_add_type_> :w When a processor is found, the processor code is stored. It will be used by __cmd_args_process: once arguments are all found. Here too the loop calls __cmd_prepare_signature_bypass:N rather than __cmd_prepare_signature:N so that the flag is not reset.

```

844 \cs_new_protected:cpn { __cmd_add_type_> :w } #1
845 {
846   \__cmd_flush_m_args:
847   \bool_set_true:N \l__cmd_prefixed_bool
848   \bool_set_true:N \l__cmd_process_some_bool
849   \tl_put_left:Nn \l__cmd_process_one_tl { {#1} }
850   \__cmd_prepare_signature_bypass:N
851 }

```

(End of definition for __cmd_add_type_> :w.)

__cmd_add_type_= :w A mix of the ideas from above: set a flag and add a processor.

```

852 \cs_new_protected:cpn { __cmd_add_type_= :w } #1
853 {
854   \__cmd_flush_m_args:
855   \bool_set_true:N \l__cmd_prefixed_bool
856   \bool_set_true:N \l__cmd_suppress_strip_bool
857   \bool_set_true:N \l__cmd_process_some_bool
858   \tl_put_left:Nn \l__cmd_process_one_tl
859   { { \__cmd_arg_to_keyvalue:nm {#1} } }
860   \__cmd_prepare_signature_bypass:N
861 }

```

(End of definition for __cmd_add_type_= :w.)

```

\__cmd_add_type_b:w
\__cmd_add_type_c:w
\__cmd_add_type_b_or_c:N
862 \cs_new_protected:Npn \__cmd_add_type_b:w
863 { \__cmd_add_type_b_or_c:N b }
864 \cs_new_protected:Npn \__cmd_add_type_c:w
865 { \__cmd_add_type_b_or_c:N c }
866 \cs_new_protected:Npn \__cmd_add_type_b_or_c:N #1

```

```

867 {
868   \__cmd_flush_m_args:
869   \__cmd_add_default:
870   \__cmd_add_grabber:N #1
871   \__cmd_prepare_signature:N
872 }

```

(End of definition for __cmd_add_type_b:w, __cmd_add_type_c:w, and __cmd_add_type_b_or_c:N.)

__cmd_add_type_D:w

```

873 \cs_new_protected:Npn \__cmd_add_type_D:w #1#2#3
874 {
875   \__cmd_flush_m_args:
876   \__cmd_add_default:n {#3}
877   \__cmd_add_grabber:N D
878   \tl_put_right:Nn \l__cmd_signature_tl { #1 #2 }
879   \__cmd_prepare_signature:N
880 }

```

(End of definition for __cmd_add_type_D:w.)

__cmd_add_type_E:w

The E-type argument needs a special handling of default values. Since each embellishment is a separate argument, it also needs to replicate the argument processors for each embellishment argument so that the numbers of arguments and processors remain in sync.

```

881 \cs_new_protected:Npn \__cmd_add_type_E:w #1#2
882 {
883   \__cmd_flush_m_args:
884   \__cmd_add_default_E:nn {#1} {#2}
885   \use:x
886   {
887     \__cmd_replicate_processor:nn { \tl_count:n {#1} }
888     { \exp_not:o \l__cmd_process_one_tl }
889   }
890   \__cmd_add_grabber:N E
891   \tl_put_right:Nn \l__cmd_signature_tl { {#1} }
892   \__cmd_prepare_signature:N
893 }

```

(End of definition for __cmd_add_type_E:w.)

__cmd_replicate_processor:nn

In the command's argument processor signature (the final argument of __cmd_start:nNNnnn) there is one braced item for each formal argument (up to nine), and in each of these items there is one braced item for each processor (as many as there were processors declared for a given argument). Something like this:

```

{ % argument processors
  { % argument 1
    { processor 1 } { processor 2 } ... { processor n }
  } % end argument 1
  { ... } % argument 2
  :

```

```

    { ... } % argument n
  } % end argument processors

```

The function `__cmd_add_grabber:N` adds one single grabber for an argument, and adds the braced item for that one argument. However, in an E-type argument each embellishment requires its own formal argument, so we need to break out of one layer of braces in `\l__cmd_process_one_tl`, add copies of the processor as necessary, and then return the removed brace. The function below does just that: it defines `\l__cmd_process_one_tl` starting with a `}_2` and ending with a `{_1`, so that it adds as many processors as needed when x-expanded.

```

894 \cs_new_protected:Npn \__cmd_replicate_processor:nn #1 #2
895   {
896     \int_compare:nNnF {#1} > { 1 } { \use_none:nnn }
897     \tl_set:Nx \l__cmd_process_one_tl
898       {
899         \exp_not:n { \exp_not:n {#2} \if_false: { \fi: } }
900         \prg_replicate:nn { #1 - 2 }
901         { \exp_not:n { \exp_not:n { {#2} } } }
902         \exp_not:n { { \if_false: } \fi: \exp_not:n {#2} }
903       }
904   }

```

(End of definition for __cmd_replicate_processor:nn.)

`__cmd_add_type_m:w` The `m` type is special as short arguments which are not post-processed are simply counted at this stage. Thus there is a check to see if either of these cases apply. If so, a one-argument grabber is added to the signature. On the other hand, if a standard short argument is required it is simply counted at this stage, to be added later using `__cmd_flush_m_args:.`

```

905 \cs_new_protected:Npn \__cmd_add_type_m:w
906   {
907     \__cmd_add_default:
908     \bool_if:NTF \l__cmd_prefixed_bool
909       { \__cmd_add_grabber:N m }
910       { \int_incr:N \l__cmd_m_args_int }
911     \__cmd_prepare_signature:N
912   }

```

(End of definition for __cmd_add_type_m:w.)

`__cmd_add_type_R:w` The R-type argument is very similar to the D-type.

```

913 \cs_new_protected:Npn \__cmd_add_type_R:w #1#2#3
914   {
915     \__cmd_flush_m_args:
916     \__cmd_add_default:n {#3}
917     \__cmd_add_grabber:N R
918     \tl_put_right:Nn \l__cmd_signature_tl { #1 #2 }
919     \__cmd_prepare_signature:N
920   }

```

(End of definition for __cmd_add_type_R:w.)

`__cmd_add_type_t:w` Setting up a `t` argument means collecting one token for the test, and adding it along with the grabber to the signature.

```
921 \cs_new_protected:Npn \__cmd_add_type_t:w #1
922   {
923     \__cmd_flush_m_args:
924     \__cmd_add_default:
925     \__cmd_add_grabber:N t
926     \tl_put_right:Nn \l__cmd_signature_tl {#1}
927     \__cmd_prepare_signature:N
928   }
```

(End of definition for __cmd_add_type_t:w.)

`__cmd_add_type_v:w` At this stage, the `v` argument is identical to `l` except that since the grabber may fail to read a verbatim argument we need a default value.

```
929 \cs_new_protected:Npn \__cmd_add_type_v:w
930   {
931     \__cmd_flush_m_args:
932     \exp_args:No \__cmd_add_default:n \c_novalue_tl
933     \__cmd_add_grabber:N v
934     \__cmd_prepare_signature:N
935   }
```

(End of definition for __cmd_add_type_v:w.)

`__cmd_flush_m_args:` As `m` arguments are simply counted, there is a need to add them to the token register in a block. As this function can only be called if something other than `m` turns up, the flag can be switched here.

```
936 \cs_new_protected:Npn \__cmd_flush_m_args:
937   {
938     \int_compare:nNnT \l__cmd_m_args_int > 0
939     {
940       \tl_put_right:Nx \l__cmd_signature_tl
941       { \exp_not:c { __cmd_grab_m_ \int_use:N \l__cmd_m_args_int :w } }
942       \tl_put_right:Nx \l__cmd_process_all_tl
943       { \prg_replicate:nn { \l__cmd_m_args_int } { { } } }
944     }
945     \int_zero:N \l__cmd_m_args_int
946   }
```

(End of definition for __cmd_flush_m_args:.)

`__cmd_add_grabber:N` To keep the various checks needed in one place, adding the grabber to the signature is done here. The only questions are whether the grabber should be long or not, and whether to obey spaces. The `\l__cmd_obey_spaces_bool` boolean can only be `true` for trailing optional arguments. In that case spaces will not be ignored when looking for that optional argument.

```
947 \cs_new_protected:Npn \__cmd_add_grabber:N #1
948   {
949     \tl_put_right:Nx \l__cmd_signature_tl
950     {
951       \exp_not:c
952       {
953         __cmd_grab_ #1
```

```

954         \bool_if:NT \l__cmd_long_bool { _long }
955         \bool_if:NT \l__cmd_obey_spaces_bool { _obey_spaces }
956         \bool_lazy_and:nnT
957           { \l__cmd_suppress_strip_bool }
958           { \str_if_eq_p:nn {#1} { D } }
959           { _no_strip }
960         \bool_lazy_all:nT
961           {
962             { \l__cmd_final_verb_bool }
963             { \str_if_eq_p:nn {#1} { D } }
964             {
965               \int_compare_p:nNn \l__cmd_current_arg_int
966                 = { \l__cmd_total_args_int - 1 }
967             }
968           }
969           { _verb_safe }
970         :w
971       }
972     }
973     \bool_set_false:N \l__cmd_long_bool
974     \bool_set_false:N \l__cmd_obey_spaces_bool
975     \bool_set_false:N \l__cmd_suppress_strip_bool
976     \bool_set_false:N \l__cmd_verb_safe_bool
977     \tl_put_right:Nx \l__cmd_process_all_tl
978       {
979         {
980           \if_charcode:w E #1 \use_i:nn \fi:
981           \exp_not:o \l__cmd_process_one_tl
982         }
983       }
984     \tl_clear:N \l__cmd_process_one_tl
985   }

```

(End of definition for __cmd_add_grabber:N.)

__cmd_add_default:n Store the default value of an argument, or rather code that gives that default value (it may involve other arguments). This is \c_novalue_tl for arguments with no actual default or with default -NoValue-; and (in a brace group) \prg_do_nothing: followed by a default value for others. For E-type arguments, pad the defaults #2 with some \c_novalue_tl until there are as many as embellishments #1. These functions are also used when defining expandable commands.

```

986 \cs_new_protected:Npn \__cmd_add_default:n #1
987   {
988     \tl_if_novalue:nTF {#1}
989       { \__cmd_add_default: }
990       {
991         \int_incr:N \l__cmd_current_arg_int
992         \bool_set_true:N \l__cmd_defaults_bool
993         \tl_put_right:Nn \l__cmd_defaults_tl { { \prg_do_nothing: #1 } }
994       }
995   }
996 \cs_new_protected:Npn \__cmd_add_default:
997   {
998     \int_incr:N \l__cmd_current_arg_int

```

```

999   \tl_put_right:Nn \l__cmd_defaults_tl { \c_novalue_tl }
1000 }
1001 \cs_new_protected:Npn \__cmd_add_default_E:nn #1#2
1002 {
1003   \tl_map_function:nN {#2} \__cmd_add_default:n
1004   \prg_replicate:nn
1005     { \tl_count:n {#1} - \tl_count:n {#2} }
1006     { \__cmd_add_default: }
1007 }

```

(End of definition for __cmd_add_default:n, __cmd_add_default:, and __cmd_add_default_E:nn.)

1.7 Setting up expandable types

The approach here is not dissimilar to that for standard types, but fewer types are supported. There is also a need to define the per-function auxiliaries: this is done here, while the general grabbers are dealt with later.

`__cmd_add_expandable_type_+:` We have already checked that short arguments are before long arguments, so `\l__cmd_long_bool` only changes from `false` to `true` once (and there is no need to reset it after each argument). Continue the loop.

```

1008 \cs_new_protected:cpn { __cmd_add_expandable_type_+:w }
1009 {
1010   \bool_set_true:N \l__cmd_long_bool
1011   \__cmd_prepare_signature:N
1012 }

```

(End of definition for __cmd_add_expandable_type_+:w.)

`__cmd_add_expandable_type_D:` The set up for D-type arguments involves constructing a rather complex auxiliary which is used repeatedly when grabbing. There is an auxiliary here so that the R-type can share code readily: #1 is D or R. The `_aux:NN` auxiliary is needed if the two delimiting tokens are identical: in contrast to the non-expandable route, the grabber here has to act differently for this case.

```

1013 \cs_new_protected:Npn \__cmd_add_expandable_type_D:w
1014   { \__cmd_add_expandable_type_D_aux:NNNn D }
1015 \cs_new_protected:Npn \__cmd_add_expandable_type_D_aux:NNNn #1#2#3#4
1016   {
1017     \__cmd_add_default:n {#4}
1018     \tl_if_eq:nnTF {#2} {#3}
1019       { \__cmd_add_expandable_type_D_aux:NN #1 #2 }
1020       { \__cmd_add_expandable_type_D_aux:NNN #1 #2 #3 }
1021     \__cmd_prepare_signature:N
1022   }
1023 \cs_new_protected:Npn \__cmd_add_expandable_type_D_aux:NNN #1#2#3
1024   {
1025     \bool_if:NTF \l__cmd_long_bool
1026       { \cs_set:cpx }
1027       { \cs_set_nopar:cpx }
1028     { \l__cmd_expandable_aux_name_tl } ##1 ##2 #2 ##3 \q__cmd ##4 #3
1029     { ##1 {##2} {##3} {##4} }
1030     \__cmd_add_expandable_grabber:nn {#1}
1031     {
1032       \exp_not:c { \l__cmd_expandable_aux_name_tl }

```



```

1033     \exp_not:n { #2 #3 }
1034   }
1035 }
1036 \cs_new_protected:Npn \__cmd_add_expandable_type_D_aux:NN #1#2
1037 {
1038   \bool_if:NTF \l__cmd_long_bool
1039     { \cs_set:cpx }
1040     { \cs_set_nopar:cpx }
1041     { \l__cmd_expandable_aux_name_tl } ##1 #2 ##2 #2
1042     { ##1 {##2} }
1043   \__cmd_add_expandable_grabber:nn { #1_alt }
1044   {
1045     \exp_not:c { \l__cmd_expandable_aux_name_tl }
1046     \exp_not:n {#2}
1047   }
1048 }

```

(End of definition for `__cmd_add_expandable_type_D:w` and others.)

`__cmd_add_expandable_type_E:w`
`__cmd_add_expandable_type_E_aux:n`

For each embellishment, use `__cmd_get_grabber:NN` to obtain an auxiliary delimited by that token and store a pair constituted of the auxiliary and the token in `\l__cmd_tmpb_tl`, before appending the whole set of these pairs to the signature, and an equal number of `-NoValue-` markers (regardless of the default values of arguments). Set the current argument appropriately.

```

1049 \cs_new_protected:Npn \__cmd_add_expandable_type_E:w #1#2
1050 {
1051   \__cmd_add_default_E:nn {#1} {#2}
1052   \tl_clear:N \l__cmd_tmpb_tl
1053   \tl_map_function:nN {#1} \__cmd_add_expandable_type_E_aux:n
1054   \__cmd_add_expandable_grabber:nn
1055     { E \bool_if:NT \l__cmd_long_bool { _long } }
1056   {
1057     { \exp_not:o \l__cmd_tmpb_tl }
1058     {
1059       \prg_replicate:nn { \tl_count:n {#1} }
1060       { { \c_novalue_tl } }
1061     }
1062   }
1063   \__cmd_prepare_signature:N
1064 }
1065 \cs_new_protected:Npn \__cmd_add_expandable_type_E_aux:n #1
1066 {
1067   \__cmd_get_grabber:NN #1 \l__cmd_tmpa_tl
1068   \tl_put_right:Nx \l__cmd_tmpb_tl
1069     { \exp_not:o \l__cmd_tmpa_tl \exp_not:N #1 }
1070 }

```

(End of definition for `__cmd_add_expandable_type_E:w` and `__cmd_add_expandable_type_E_aux:n`.)

`__cmd_add_expandable_type_m:w`

Unlike the standard case, when working expandably each argument is always grabbed separately.

```

1071 \cs_new_protected:Npn \__cmd_add_expandable_type_m:w
1072 {
1073   \__cmd_add_default:

```

```

1074 \__cmd_add_expandable_grabber:nn
1075 { m \bool_if:NT \l__cmd_long_bool { _long } } { }
1076 \__cmd_prepare_signature:N
1077 }

```

(End of definition for __cmd_add_expandable_type_m:w.)

`__cmd_add_expandable_type_R:w` The R-type is very similar to the D-type argument, and so the same internals are used.

```

1078 \cs_new_protected:Npn \__cmd_add_expandable_type_R:w
1079 { \__cmd_add_expandable_type_D_aux:NNNn R }

```

(End of definition for __cmd_add_expandable_type_R:w.)

`__cmd_add_expandable_type_t:w` An auxiliary delimited by #1 is built now. It will be used to test for the presence of that token.

```

1080 \cs_new_protected:Npn \__cmd_add_expandable_type_t:w #1
1081 {
1082   \__cmd_add_default:
1083   \__cmd_get_grabber:NN #1 \l__cmd_tmpa_tl
1084   \__cmd_add_expandable_grabber:nn { t }
1085   {
1086     \exp_not:o \l__cmd_tmpa_tl
1087     \exp_not:N #1
1088   }
1089   \__cmd_prepare_signature:N
1090 }

```

(End of definition for __cmd_add_expandable_type_t:w.)

`__cmd_add_expandable_grabber:nn` This is called for all arguments to place the right grabber in the signature.

```

1091 \cs_new_protected:Npn \__cmd_add_expandable_grabber:nn #1#2
1092 {
1093   \tl_put_right:Nx \l__cmd_signature_tl
1094   { \exp_not:c { __cmd_expandable_grab_ #1 :w } #2 }
1095 }

```

(End of definition for __cmd_add_expandable_grabber:nn.)

`__cmd_get_grabber:NN`
`__cmd_get_grabber_auxi:NN`
`__cmd_get_grabber_auxii:NN` Given a token #1, defines an expandable function delimited by that token and stores it in the token list #2. The function is named after the token, unless that function name is already taken by some other grabber (this can happen in the rare case where delimiters with different category codes are used in the same document): in that case use a global counter to get a unique name. Since the grabbers are not named after xparse commands they should not be used to get material from the input stream.

```

1096 \cs_new_protected:Npn \__cmd_get_grabber:NN #1#2
1097 {
1098   \cs_set:Npn \__cmd_tmp:w ##1 #1 {##1}
1099   \exp_args:Nc \__cmd_get_grabber_auxi:NN
1100   { __cmd_grabber_ \token_to_str:N #1 :w } #2
1101 }
1102 \cs_new_protected:Npn \__cmd_get_grabber_auxi:NN #1#2
1103 {
1104   \cs_if_eq:NNTF \__cmd_tmp:w #1
1105   { \tl_set:Nn #2 {#1} }

```

```

1106     {
1107       \cs_if_exist:NTF #1
1108       {
1109         \int_gincr:N \g__cmd_grabber_int
1110         \exp_args:Nc \__cmd_get_grabber_auxi:NN
1111         {
1112           __cmd_grabber_
1113           - \int_use:N \g__cmd_grabber_int :w
1114         }
1115         #2
1116       }
1117       { \__cmd_get_grabber_auxii:NN #1 #2 }
1118     }
1119   }
1120 \cs_new_protected:Npn \__cmd_get_grabber_auxii:NN #1#2
1121   {
1122     \cs_set_eq:NN #1 \__cmd_tmp:w
1123     \tl_set:Nn #2 {#1}
1124   }

```

(End of definition for `__cmd_get_grabber:NN`, `__cmd_get_grabber_auxi:NN`, and `__cmd_get_grabber_auxii:NN`.)

1.7.1 Copying a command and its internal structure

```

1125 <latexrelease>\IncludeInRelease{2021/11/15}{\__cmd_copy:NN}%
1126 <latexrelease> {Support~\NewCommandCopy~in~ltxcmd}

```

Since the 2020-10-01 L^AT_EX 2_ε release, support for copying, and showing the definition of, robust commands has been available, but the specifics of each command are implemented separately. Here we'll add support for copying and showing `ltxcmd` definitions.

To fully support copying, we need two commands: a conditional to test if a command is in fact a `ltxcmd` command, and another command to actually copy the command. The conditional is defined later as `__kernel_cmd_if_xparse:NTF`, so now to the copying:

This macro just branches to the proper copying command by using `__cmd_cmd_type_cases:NnnnnnF`. The copying command takes the names of the commands to be copied to and from, and the actual commands as its four arguments.

```

\__cmd_copy:NN
\__cmd_set_eq_if_exist:NN
\__cmd_set_eq_if_exist:cc
1127 \cs_new_protected:Npn \__cmd_copy:NN #1 #2
1128   {
1129     \use:x
1130     {
1131       \int_set:Nn \tex_escapechar:D { 92 }
1132       \exp_not:N \__cmd_cmd_type_cases:NnnnnnF \exp_not:N #2
1133       { \__cmd_copy_command:nnNN }
1134       { \__cmd_copy_expandable:nnNN }
1135       { \__cmd_copy_optimized:nnNN }
1136       { \__cmd_copy_environment:nnNN }
1137       { \__cmd_copy_environment_end:nnNN }
1138       { \__cmd_cant_copy:nwn { non-ltxcmd } }
1139       { \cs_to_str:N #1 } { \cs_to_str:N #2 }
1140       \exp_not:N #1 \exp_not:N #2
1141       \exp_not:N \__cmd_break_point:n { \cs_to_str:N #2 }
1142       \int_set:Nn \tex_escapechar:D { \int_use:N \tex_escapechar:D }

```

```

1143     }
1144   }
1145   \cs_new_protected:Npn \__cmd_set_eq_if_exist:NN #1 #2
1146   { \cs_if_exist:NTF #2 { \cs_set_eq:NN } { \use_none:nn } #1 #2 }
1147   \cs_generate_variant:Nn \__cmd_set_eq_if_exist:NN { cc }

```

An utility macro similar to `__cmd_bad_def:wn` to abort a command copy. Contrary to `__cmd_bad_def:wn` though, when this happens the issue is most likely internal, because the command was already (supposedly) correctly defined so it should be copyable. Hopefully this macro will never be used ever, but if it does, apologise and give the reason for the failure so the user can report.

```

\__cmd_cant_copy:nwn
1148 \cs_new_protected:Npn \__cmd_cant_copy:nwn #1 #2 \__cmd_break_point:n #3
1149   { \msg_error:nnnn { cmd } { copy-bug } {#1} {#3} }
1150 \msg_new:nnn { cmd } { copy-bug }
1151   {
1152     Error~while~copying~command~\iow_char:N\#2:\
1153     \str_case:nn {#1}
1154     {
1155       { non-ltcmd } { Command~is~not~a~valid~ltcmd~command. }
1156       { unknown-type } { Found~an~unknown~argument~type. }
1157       { invalid-end }
1158       { Target~command~is~not~named~\iow_char:N \end<name>. }
1159     }
1160   }

```

And, of course, add `__kernel_cmd_if_xparse:NTF` and `__cmd_copy:NN` to `\@declarecommandcopylisthook`:

```

1161 \tl_gput_right:Nn \@declarecommandcopylisthook
1162   { { \__kernel_cmd_if_xparse:NTF \__cmd_copy:NN } }

```

(End of definition for `__cmd_copy:NN`, `__cmd_set_eq_if_exist:NN`, and `__cmd_cant_copy:nwn`.)

```

\__cmd_copy_command:nnNN
\__cmd_copy_command:NnNNnnnn

```

A normal (non-expandable) command has a pretty straightforward structure. Its definition is stored in `\(cmd)_code`, its defaults (if any) are stored in `\(cmd)_defaults`, and its top-level definition contains its signature, which can just be copied over. `__cmd_copy_command:nnNN` copies the command code and defaults, and then defines the top-level command using the auxiliary `__cmd_copy_command:NnNNnnnn`. This macro takes the signature of the command being copied from its top-level definition, and replaces the named bits with the new name.

```

1163 \cs_new_protected:Npn \__cmd_copy_command:nnNN #1 #2 #3 #4
1164   {
1165     \cs_set_eq:cc { #1 ~ code } { #2 ~ code }
1166     \__cmd_set_eq_if_exist:cc { #1 ~ defaults } { #2 ~ defaults }
1167     \cs_set_protected_nopar:Npx #3
1168     { \exp_after:wN \__cmd_copy_command:NnNNnnnn #4 {#1} }
1169   }
1170 \cs_new:Npn \__cmd_copy_command:NnNNnnnn #1 #2 #3 #4 #5 #6 #7 #8
1171   {
1172     #1 \exp_not:n { {#2} }
1173     \exp_not:c { #8 ~ } \exp_not:c { #8 ~ code }
1174     \exp_not:n { {#5} {#6} {#7} }
1175   }

```

(End of definition for `__cmd_copy_command:nnNN` and `__cmd_copy_command:NnNNnnnn`.)

```

\__cmd_copy_expandable:nnNN
  \__cmd_copy_expandable:NnNNNNnnn

```

An expandable command is slightly more complicated. Besides the `\<cmd>_code`, and `\<cmd>_defaults`, it also has an auxiliary `\<cmd>_l` for grabbing delimited arguments, and possibly another auxiliary `\<cmd>_ll`, if the command has both long and short arguments. Then, its signature also has several specific bits that are unique to that command; this is in contrast to non-expandable commands, which use a common set of parsing functions.

We start by copying the basics, then call `__cmd_copy_expandable_signature:NnNNNNnnn` to parse the signature of the command and build up the modified copy in a temporary token list, then we call `__cmd_copy_expandable:NnNNNNnnn` that will copy the top-level definition of the command, with the proper internal renames.

```

1176 <latexrelease>\EndIncludeInRelease
1177 <latexrelease>\IncludeInRelease{2020/10/01}{\__cmd_copy:NN}%
1178 <latexrelease> {Support~\NewCommandCopy~in~ltxcmd}
1179 <latexrelease>\EndIncludeInRelease

```

There's one variant: a command begins with `__cmd_start_expandable:nNNNNn` may still be un-expandable/protected if it's defined by `\NewDocumentCommand` and friends, with empty or only m-type arguments.

```

1180 <latexrelease>\IncludeInRelease{2023/06/01}{\__cmd_copy_expandable:nnNN}%
1181 <latexrelease> {Distinguish~non-expandable~document~commands}
1182 \cs_new_protected:Npn \__cmd_copy_expandable:nnNN #1 #2 #3 #4
1183 {
1184   \cs_set_eq:cc { #1 ~ code } { #2 ~ code }
1185   \__cmd_set_eq_if_exist:cc { #1 ~ } { #2 ~ }
1186   \__cmd_set_eq_if_exist:cc { #1 ~ \c_space_tl } { #2 ~ \c_space_tl }
1187   \__cmd_set_eq_if_exist:cc { #1 ~ defaults } { #2 ~ defaults }
1188   \exp_after:wN \__cmd_copy_expandable_signature:NnNNNNnnn #4 {#1} {#2}
1189   \token_if_protected_macro:NTF #4
1190   { \cs_set_protected_nopar:Npx }{ \cs_set_nopar:Npx }
1191   #3
1192   { \exp_after:wN \__cmd_copy_expandable:NnNNNNnnn #4 {#1} {#2} }
1193 }
1194 <latexrelease>\EndIncludeInRelease
1195 <latexrelease>\IncludeInRelease{2021/11/15}{\__cmd_copy_expandable:nnNN}%
1196 <latexrelease> {Support~\NewCommandCopy~in~ltxcmd}
1197 <latexrelease>\cs_new_protected:Npn \__cmd_copy_expandable:nnNN #1 #2 #3 #4
1198 <latexrelease> {
1199 <latexrelease>   \cs_set_eq:cc { #1 ~ code } { #2 ~ code }
1200 <latexrelease>   \__cmd_set_eq_if_exist:cc { #1 ~ } { #2 ~ }
1201 <latexrelease>   \__cmd_set_eq_if_exist:cc { #1 ~ \c_space_tl } { #2 ~ \c_space_tl }
1202 <latexrelease>   \__cmd_set_eq_if_exist:cc { #1 ~ defaults } { #2 ~ defaults }
1203 <latexrelease>   \exp_after:wN \__cmd_copy_expandable_signature:NnNNNNnnn #4 {#1} {#2}
1204 <latexrelease>   \cs_set_nopar:Npx #3
1205 <latexrelease>   { \exp_after:wN \__cmd_copy_expandable:NnNNNNnnn #4 {#1} {#2} }
1206 <latexrelease> }
1207 <latexrelease>\EndIncludeInRelease
1208 <latexrelease>\IncludeInRelease{2020/10/01}{\__cmd_copy_expandable:nnNN}%
1209 <latexrelease> {Support~\NewCommandCopy~in~ltxcmd}
1210 <latexrelease>\EndIncludeInRelease

```

Copy the code, simply define the wrapper.

```

\__cmd_copy_optimized:nnNN
1211 \cs_new_protected:Npn \__cmd_copy_optimized:nnNN #1#2#3#4
1212 {
1213   \cs_set_eq:cc { #1 ~ code } { #2 ~ code }

```

```

1214 \token_if_protected_macro:NTF #4
1215 { \cs_set_protected_nopar:Npe }
1216 { \cs_set_nopar:Npe }
1217 #3
1218 {
1219 \exp_not:N \__cmd_start_optimized:
1220 \exp_not:c { #1 ~ code }
1221 }
1222 }

1223 <latexrelease>\IncludeInRelease{2021/11/15}{\__cmd_copy:NN (part 2)}%
1224 <latexrelease> {Support~\NewCommandCopy~in~ltxcmd}

1225 \cs_new:Npn \__cmd_copy_expandable:NnNNNNnnn #1 #2 #3 #4 #5 #6 #7 #8 #9
1226 {
1227 \exp_not:N #1 \exp_not:n { {#2} }
1228 \exp_not:c { #8 ~ }
1229 \exp_not:c
1230 {
1231 #8 ~
1232 \str_if_eq:eeT
1233 { \exp_not:c { #9 ~ \c_space_tl } } { \exp_not:N #4 }
1234 { \c_space_tl }
1235 }
1236 \exp_not:c { #8 ~ code }
1237 \str_if_eq:eeTF { \exp_not:N #6 } { ? }
1238 { ? }
1239 { \exp_not:c { #8 ~ defaults } }
1240 { \exp_not:V \l__cmd_tmpa_tl }
1241 }

```

A signature for an expandable command contains as many `\expandable_grab_⟨type⟩:w` as there are arguments, and what follows this macro depends on the `⟨type⟩`. We'll start a loop through the signature, and at each argument grabber, we'll step the argument count, and look for the `⟨type⟩` with `__cmd_copy_parse_grabber:w` so that we know which `__cmd_copy_grabber_⟨type⟩:w` to call next.

```

1242 \cs_new_protected:Npn \__cmd_copy_expandable_signature:NnNNNNnnn
1243 #1 #2 #3 #4 #5 #6 #7 #8 #9
1244 {
1245 \int_zero:N \l__cmd_current_arg_int
1246 \tl_clear:N \l__cmd_tmpa_tl
1247 \__cmd_copy_expandable:nnN {#8} {#9} #7
1248 \q_recursion_tail \q_recursion_stop
1249 }
1250 \cs_new_protected:Npn \__cmd_copy_expandable:nnN #1 #2 #3
1251 {
1252 \quark_if_recursion_tail_stop:n {#3}
1253 \int_incr:N \l__cmd_current_arg_int
1254 \exp_after:wN \__cmd_copy_parse_grabber:w \token_to_str:N #3 {#1} {#2}
1255 }
1256 \use:x
1257 {
1258 \cs_new_protected:Npn \exp_not:N \__cmd_copy_parse_grabber:w ##1
1259 \tl_to_str:n { expandable_grab_ } ##2 \tl_to_str:n { :w }

```

```

1260     {
1261       \tl_put_right:Nx \exp_not:N \l__cmd_tmpa_tl
1262       { \exp_not:N \exp_not:c { __cmd_expandable_grab_##2:w } }
1263       \exp_not:N \cs_if_exist_use:cF { __cmd_copy_grabber_##2:w }
1264       { \__cmd_cant_copy:nwn { unknown-type } }
1265     }
1266   }

```

The most complicated is the Delimited argument: each argument has a dedicated grabbing function named after the command that has to be copied over (of the form $\backslash\langle cmd \rangle_{\langle arg \rangle}(\langle num \rangle)$).

```

1267 \cs_new_protected:Npn \__cmd_copy_grabber_D:w #1 #2 #3 #4 #5
1268   {
1269     \tl_put_right:Nx \l__cmd_tmpa_tl
1270     {
1271       \exp_not:c { #1 ~ (arg ~ \int_use:N \l__cmd_current_arg_int ) }
1272       \exp_not:n { #4 #5 }
1273     }
1274     \cs_set_eq:cc
1275     { #1 ~ (arg ~ \int_use:N \l__cmd_current_arg_int ) }
1276     { #2 ~ (arg ~ \int_use:N \l__cmd_current_arg_int ) }
1277     \__cmd_copy_expandable:nnN {#1} {#2}
1278   }

```

D_alt is just a special case of D that uses a single delimiter (used when both delimiters of the argument are identical):

```

1279 \cs_new_protected:Npn \__cmd_copy_grabber_D_alt:w #1 #2 #3 #4
1280   { \__cmd_copy_grabber_D:w {#1} {#2} {#3} {#4} { } }

```

As far as copying is concerned, R is identical to D:

```

1281 \cs_new_eq:NN \__cmd_copy_grabber_R:w \__cmd_copy_grabber_D:w
1282 \cs_new_eq:NN \__cmd_copy_grabber_R_alt:w \__cmd_copy_grabber_D_alt:w

```

E is straightforward: we just copy the embellishments over, and increase the current argument number $\backslash\l__cmd_current_arg_int$ by the number of embellishments (minus one because there is a $\backslash\int_incr:N$ down the line).

```

1283 \cs_new_protected:Npn \__cmd_copy_grabber_E:w #1 #2 #3 #4
1284   {
1285     \tl_put_right:Nn \l__cmd_tmpa_tl { {#3} {#4} }
1286     \int_add:Nn \l__cmd_current_arg_int { \tl_count:n {#4} - 1 }
1287     \__cmd_copy_expandable:nnN {#1} {#2}
1288   }
1289 \cs_new_eq:NN \__cmd_copy_grabber_E_long:w \__cmd_copy_grabber_E:w

```

t just needs copying the token to be tested for:

```

1290 \cs_new_protected:Npn \__cmd_copy_grabber_t:w #1 #2 #3 #4
1291   {
1292     \tl_put_right:Nn \l__cmd_tmpa_tl { #3 #4 }
1293     \__cmd_copy_expandable:nnN {#1} {#2}
1294   }

```

And last but not least, m is the simplest; the grabber is just $\backslash__cmd_expandable_grab_m:w$, which is already added to the new command so here we just resume the loop:

```

1295 \cs_new_protected:Npn \__cmd_copy_grabber_m:w { \__cmd_copy_expandable:nnN }
1296 \cs_new_eq:NN \__cmd_copy_grabber_m_long:w \__cmd_copy_grabber_m:w

```

(End of definition for `_cmd_copy_expandable:nnNN` and others.)

`_cmd_copy_environment:nnNN` Copying an environment's `\begin` part is pretty much like copying a command, except it
`_cmd_copy_environment:Nnnnnnn` has a longer name, and at the end we have to copy `\environment <name>` into `\<name>`.

```
1297 \cs_new_protected:Npn \_cmd_copy_environment:nnNN #1 #2 #3 #4
1298   {
1299     \cs_set_eq:cc { environment~ #1 ~ code } { environment~ #2 ~ code }
1300     \_cmd_set_eq_if_exist:cc
1301       { environment~ #1 ~ defaults } { environment~ #2 ~ defaults }
1302     \cs_set_protected_nopar:cpx { environment~ #1 }
1303       { \exp_after:wN \_cmd_copy_environment:Nnnnnnn #4 {#1} }
1304     \cs_set_eq:cc {#1} { environment~ #1 }
1305   }
1306 \cs_new:Npn \_cmd_copy_environment:Nnnnnnn #1 #2 #3 #4 #5 #6 #7
1307   { #1 \exp_not:n { {#2} } {#7} \exp_not:n { {#4} {#5} {#6} } }
```

(End of definition for `_cmd_copy_environment:nnNN` and `_cmd_copy_environment:Nnnnnnn`.)

`_cmd_copy_environment_end:nnNN` Copying an environment's `\end` part is a bit trickier. We first have to make sure that both
`_cmd_copy_environment_end_aux:nnNN` parts are named `\end<name>` (that's actually not a hard requirement, but an environment
`\end` command makes no sense without the `end` in its name), and strip the leading `end`
from the strings. After that, copying is straightforward.

```
1308 \cs_new_protected:Npn \_cmd_copy_environment_end:nnNN #1 #2
1309   {
1310     \_cmd_check_end:Nn \l__cmd_tmpa_tl {#1}
1311     \_cmd_check_end:Nn \l__cmd_tmpb_tl {#2}
1312     \exp_args:Noo \_cmd_copy_environment_end_aux:nnNN
1313       { \l__cmd_tmpa_tl } { \l__cmd_tmpb_tl }
1314   }
1315 \cs_new_protected:Npn \_cmd_copy_environment_end_aux:nnNN #1 #2 #3 #4
1316   {
1317     \cs_set_nopar:cpx { environment~ #1 ~end }
1318       { \exp_not:c { environment~ #1 ~end~aux } }
1319     \cs_set_eq:cc
1320       { environment~ #1 ~end~aux~ } { environment~ #2 ~end~aux~ }
1321     \cs_set_eq:cc { end #1 } { environment~ #1 ~end }
1322   }
```

To check whether an `\end` command is valid, we look for the string `end` at the beginning of the command name, and if not found, raise an error:

```
\_cmd_check_end:Nn 1323 \cs_new_protected:Npn \_cmd_check_end:Nn #1 #2
\_cmd_check_end:n 1324   {
\_cmd_check_end:w 1325     \tl_set:Nx #1 { \_cmd_check_end:n {#2} }
1326     \token_if_eq_meaning:NNT #1 \q_nil
1327     { \_cmd_cant_copy:nwn { invalid~end } }
1328   }
1329 \cs_set_protected:Npn \_cmd_tmp:w #1
1330   {
1331     \cs_new:Npn \_cmd_check_end:n ##1
1332     {
1333       \exp_after:wN \_cmd_check_end:w \tl_to_str:n {##1}
1334       #1 \q_mark #1 \q_stop
1335     }
1336     \cs_new:Npn \_cmd_check_end:w ##1 #1 ##2 #1 ##3 \q_stop
```



```

1337     { \if_meaning:w ##2 \q_mark \exp_not:N \q_nil \else: ##2 \fi: }
1338   }
1339 \exp_args:No \__cmd_tmp:w { \tl_to_str:n { end } }

```

(End of definition for `__cmd_copy_environment_end:nnNN` and others.)

Not much to do regarding `latexrelease`: we could remove the entries from `\@declarecommandcopylist` but it doesn't seem worth it.

```

1340 \langle latexrelease \rangle \EndIncludeInRelease
1341 \langle latexrelease \rangle \IncludeInRelease{2020/10/01}{\__cmd_copy:NN (part 2)}%
1342 \langle latexrelease \rangle {Support~\NewCommandCopy~in~ltxcmd}
1343 \langle latexrelease \rangle \EndIncludeInRelease

```

1.7.2 Showing the definition of a command

```

1344 \langle latexrelease \rangle \IncludeInRelease{2021/11/15}{\__cmd_show:N}%
1345 \langle latexrelease \rangle {Support~\ShowCommand~in~ltxcmd}

```

To show the definition of a command we need more or less the same building blocks as for copying, except that instead of making a copy, we'll just print stuff to the terminal. This macro just branches to the proper showing command by using `__cmd_cmd_type_cases:NnnnnnF`. The showing command takes the command to be shown as argument.

`__cmd_show:N`

```

1346 \cs_new_protected:Npn \__cmd_show:N #1
1347 {
1348   \use:x
1349   {
1350     \int_set:Nn \tex_escapechar:D { 92 }
1351     \exp_not:N \__cmd_cmd_type_cases:NnnnnnF \exp_not:N #1
1352     { \__cmd_show_command:N }
1353     { \__cmd_show_expandable:N }
1354     { \__cmd_show_optimized:N }
1355     { \__cmd_show_environment:N }
1356     { \__cmd_show_environment_end:N }
1357     { \__cmd_cant_copy:nwn { non-ltxcmd } }
1358     \exp_not:N #1
1359     \exp_not:N \__cmd_break_point:n { \cs_to_str:N #1 }
1360     \int_set:Nn \tex_escapechar:D { \int_use:N \tex_escapechar:D }
1361   }
1362 }

```

(End of definition for `__cmd_show:N`.)

`__cmd_show_command:N`
`__cmd_show_command:NnNNwN`
`__cmd_show_expandable:N`
`__cmd_show_expandable:NnNNNNnN`
`__cmd_show_optimized:N`
`__cmd_show_command_aux:NnNNn`
`__cmd_show_environment:N`
`__cmd_show:x`

These commands just expand the command once to reveal its innards, then pass the type of command, the control sequence, the signature, and the code macro to `__cmd_show_command_aux:NnNNn`.

```

1363 \cs_new_protected:Npn \__cmd_show_command:N #1
1364 { \exp_after:wN \__cmd_show_command:NnNNwN #1 \q__cmd #1 }
1365 \cs_new_protected:Npn \__cmd_show_command:NnNNwN #1 #2 #3 #4 #5 \q__cmd #6
1366 {
1367   \__cmd_show_command_aux:NnNNn \tl_show:x
1368   { document~command } #6 #4 {#2}
1369 }
1370 \cs_new_protected:Npn \__cmd_show_expandable:N #1
1371 { \exp_after:wN \__cmd_show_expandable:NnNNNNnN #1 #1 }

```

```

1372 <latexrelease>\EndIncludeInRelease
1373 <latexrelease>\IncludeInRelease{2020/10/01}{\__cmd_show:N}%
1374 <latexrelease> {Support~\ShowCommand-in~ltxcmd}
1375 <latexrelease>\EndIncludeInRelease

There's one variant: a command begins with \__cmd_start_expandable:nNNNNn
may still be un-expandable/protected if it's defined by \NewDocumentCommand and
friends, with empty or only m-type arguments.

1376 <latexrelease>\IncludeInRelease{2023/06/01}{\__cmd_show_expandable:NnNNNNn}%
1377 <latexrelease> {Distinguish-non-expandable~document~commands}
1378 \cs_new_protected:Npn \__cmd_show_expandable:NnNNNNn #1 #2 #3 #4 #5 #6 #7 #8
1379 {
1380   \exp_args:Nne \__cmd_show_command_aux:NnNNn \tl_show:x
1381   { \token_if_protected_macro:NF #8 { expandable~ } document~command }
1382   #8 #5 {#2}
1383 }
1384 <latexrelease>\EndIncludeInRelease
1385 <latexrelease>\IncludeInRelease{2021/11/15}{\__cmd_show_expandable:NnNNNNn}%
1386 <latexrelease> {Support~\ShowCommand-in~ltxcmd}
1387 <latexrelease>\cs_new_protected:Npn \__cmd_show_expandable:NnNNNNn #1 #2 #3 #4 #5 #6 #7 #8
1388 <latexrelease> {
1389 <latexrelease>   \__cmd_show_command_aux:NnNNn \tl_show:x
1390 <latexrelease>   { expandable~document~command } #8 #5 {#2}
1391 <latexrelease> }
1392 <latexrelease>\EndIncludeInRelease
1393 <latexrelease>\IncludeInRelease{2020/10/01}{\__cmd_show_expandable:NnNNNNn}%
1394 <latexrelease> {Support~\ShowCommand-in~ltxcmd}
1395 <latexrelease>\EndIncludeInRelease

1396 <latexrelease>\IncludeInRelease{2021/11/15}{\__cmd_show:N (part 2)}%
1397 <latexrelease> {Support~\ShowCommand-in~ltxcmd}

```

Now just print everything in the required format. The auxiliary `__cmd_split_signature:n` stores a ready-to-print token list in `\l__cmd_tmpa_tl`, so we use that here:

```

1398 \cs_new_protected:Npn \__cmd_show_command_aux:NnNNn #1 #2 #3 #4 #5
1399 {
1400   \__cmd_split_signature:n {#5}
1401   #1
1402   {
1403     \token_to_str:N #3 = #2: \iow_newline:
1404     \tl_use:N \l__cmd_tmpa_tl
1405     -> \cs_replacement_spec:N #4
1406   }
1407 }

```

Optimized functions need things done a bit differently as we need to reconstruct the argument spec.

```

1408 \cs_new_protected:Npn \__cmd_show_optimized:N #1
1409 {
1410   \exp_args:Nc \__cmd_show_optimized:NN
1411   { \cs_to_str:N #1 \c_space_tl code }
1412   #1
1413 }
1414 \cs_new_protected:Npn \__cmd_show_optimized:NN #1#2

```

```

1415 {
1416   \cs_set:Npe \__cmd_show_optimized_aux:N ##1
1417   {
1418     \c_space_tl \c_space_tl \c_hash_str ##1 :
1419     \bool_lazy_or:nnT
1420     { \token_if_long_macro_p:N #1 }
1421     { \token_if_protected_long_macro_p:N #1 }
1422     { + } m
1423     \iow_newline:
1424   }
1425   \tl_show:e
1426   {
1427     \token_to_str:N #2 =
1428     \bool_lazy_or:nnF
1429     { \token_if_protected_macro_p:N #1 }
1430     { \token_if_protected_long_macro_p:N #1 }
1431     { expandable ~ } document-command:
1432     \iow_newline:
1433     \int_step_function:nN
1434     {
1435       \int_div_truncate:nn
1436       { \tl_count:e { \cs_parameter_spec:N #1 } }
1437       { 2 }
1438     }
1439     \__cmd_show_optimized_aux:N
1440     ->
1441     \cs_replacement_spec:N #1
1442   }
1443 }
1444 \cs_generate_variant:Nn \tl_count:n { e }

```

We can reuse most of the above to show an environment, except that we need to ensure that the proper `\environment ...` are passed to `__cmd_show_command_aux:NnNn`. Additionally, when `\ShowCommand\foo` is used (if `foo` is an environment), we show `\endfoo` as well, and when `\ShowCommand\endfoo` is used, change that to `\ShowCommand\foo` and do the same.

```

1445 \cs_new_protected:Npn \__cmd_show_environment:N #1
1446 {
1447   \exp_after:wN \__cmd_show_environment:Nnnw #1 \q__cmd
1448   \tl_show:x
1449   {
1450     \token_to_str:N \end { \cs_to_str:N #1 } : \iow_newline:
1451     -> \exp_args:Nc \cs_replacement_spec:N
1452     { environment~ \cs_to_str:N #1 ~end-aux~ }
1453   }
1454 }
1455 \cs_new_protected:Npn \__cmd_show_environment:Nnnw #1 #2 #3 #4 \q__cmd
1456 {
1457   \use:x
1458   {
1459     \__cmd_show_command_aux:NnNn \__cmd_show:x { document-environment }
1460     { \exp_not:N \begin {#3} }
1461     \exp_not:c { environment~ #3 ~ code }
1462     {#2}

```

```

1463     }
1464   }
1465   \cs_new_protected:Npn \__cmd_show:x #1
1466   { \iow_term:x { > ~ #1 . \iow_newline: } }
1467   \cs_new_protected:Npn \__cmd_show_environment_end:N #1
1468   {
1469     \exp_args:NNx \__cmd_check_end:Nn \l__cmd_tmpa_tl { \cs_to_str:N #1 }
1470     \exp_args:Nc \__cmd_show_environment:N { \l__cmd_tmpa_tl }
1471   }

```

And, of course, add `__kernel_cmd_if_xparse:NTF` and `__cmd_show:N` to `\@showcommandlisthook` and to `\@showenvironmentlisthook` (`__cmd_show:N` takes care of the environment case as well, so both entries are identical):

```

1472 \tl_gput_right:Nn \@showcommandlisthook
1473   { { \__kernel_cmd_if_xparse:NTF \__cmd_show:N } }
1474 \tl_gput_right:Nn \@showenvironmentlisthook
1475   { { \__kernel_cmd_if_xparse:NTF \__cmd_show:N } }

```

(End of definition for `__cmd_show_command:N` and others.)

`__cmd_split_signature:n` Now we'll try a least-effort adventure into splitting the symbolic user-provided signature for a command into individual parameters for pretty-printing. A counter is used to keep track of the current argument number, and two token lists are used: `\l__cmd_tmpa_tl` holds the final token list to be printed, and `\l__cmd_tmpb_tl` holds just the current item, so that we can make changes to an individual item without having to dissect the whole thing (this is used for e- and E-types).

```

1476 \cs_new_protected:Npn \__cmd_split_signature:n #1
1477   {
1478     \int_set:Nn \l__cmd_current_arg_int { 1 }
1479     \tl_clear:N \l__cmd_tmpa_tl
1480     \tl_clear:N \l__cmd_tmpb_tl
1481     \__cmd_split_signature_loop:Nw #1 \q_recursion_tail \q_recursion_stop
1482   }

```

`__cmd_split_signature_loop:Nw` This is the main chunk of the loop: it starts an item with `__cmd_split_start_item:` (this adds indentation and the argument number to `\l__cmd_tmpb_tl`), then checks if a special token list `\c__cmd_show_type_⟨type⟩_tl` exists. If it doesn't, the current argument is a "simple" type which needs no extra processing. Otherwise, call a specific function depending on the value of said token list.

```

1483 \cs_new_protected:Npn \__cmd_split_signature_loop:Nw #1
1484   {
1485     \quark_if_recursion_tail_stop:N #1
1486     \tl_if_empty:NT \l__cmd_tmpb_tl { \__cmd_split_start_item: }
1487     \tl_if_exist:cTF { c__cmd_show_type_#1_tl }
1488     {
1489       \use:c
1490       {
1491         __cmd_show_
1492         \if_case:w \tl_use:c { c__cmd_show_type_#1_tl } \exp_stop_f:
1493         delim \or: delims \or: delims_opt \or: opt \or:
1494         e \or: E \or: prefix \or: processor \fi: :Nw
1495       } #1
1496     }
1497     { \__cmd_split_end_item:n {#1} \__cmd_split_signature_loop:Nw }

```

1498 }

The token lists `\c__cmd_show_type_⟨type⟩_t1` exist for nontrivial (for printing) *⟨types⟩* that require special parsing (like delimiters or optional arguments). Values from 0 to 7 are assigned to each type:

- `\c__cmd_show_type_t_t1`
- `\c__cmd_show_type_r_t1`
- `\c__cmd_show_type_d_t1` 1. a single delimiter token;
- `\c__cmd_show_type_R_t1` 2. two delimiter tokens;
- `\c__cmd_show_type_D_t1` 3. two delimiter tokens plus a default value;
- `\c__cmd_show_type_0_t1` 4. a default value;
- `\c__cmd_show_type_e_t1` 5. a list of embellishments (exclusive for e-type);
- `\c__cmd_show_type_E_t1` 6. embellishments plus defaults (exclusive for E-type);
- `\c__cmd_show_type+_t1` 7. simple prefixes;
- `\c__cmd_show_type!_t1` 8. prefixes with arguments (argument processors);
- `\c__cmd_show_type>_t1`

```
1499 \cs_set_protected:Npn \__cmd_tmp:w #1 #2
1500   {
1501     \quark_if_nil:nF {#1}
1502     { \tl_const:cn { c__cmd_show_type_#1_t1 } {#2} \__cmd_tmp:w }
1503   }
1504 \__cmd_tmp:w t0 r1 d1 R2 D2 O3 e4 E5 +6 !6 >7 =7 \q_nil \q_nil
```

Now, based on each type we know how to act. In most cases it is just a matter of feeding in the grabbed arguments and resuming the loop. The embellishments require a bit more attention: the e-type loops through the list of embellishments and adds each to the token list as a separate argument. The E-type does more or less the same, but uses `__cmd_tl_mapthread_function:nnN` to map over two lists simultaneously, adding each token and default to the token list for printing.

```
1505 \cs_new_protected:Npn \__cmd_show_delim:Nw #1 #2
1506   { \__cmd_split_end_item:n { #1 #2 } \__cmd_split_signature_loop:Nw }
1507 \cs_new_protected:Npn \__cmd_show_delims:Nw #1 #2 #3
1508   { \__cmd_split_end_item:n { #1 #2 #3 } \__cmd_split_signature_loop:Nw }
1509 \cs_new_protected:Npn \__cmd_show_delims_opt:Nw #1 #2 #3 #4
1510   { \__cmd_split_end_item:n { #1 #2 #3 {#4} } \__cmd_split_signature_loop:Nw }
1511 \cs_new_protected:Npn \__cmd_show_opt:Nw #1 #2
1512   { \__cmd_split_end_item:n { #1 {#2} } \__cmd_split_signature_loop:Nw }
1513 \cs_new_protected:Npn \__cmd_show_e:Nw #1 #2
1514   {
1515     \tl_map_inline:nn {#2}
1516     {
1517       \__cmd_split_start_item:
1518       \__cmd_split_end_item:n { #1 ##1 }
1519     }
1520     \__cmd_split_signature_loop:Nw
1521   }
1522 \cs_set_protected:Npn \__cmd_tmp:w #1
1523   {
1524     \cs_new_protected:Npn \__cmd_show_E:Nw ##1 ##2 ##3
```

```

1525     {
1526       \cs_set_protected:Npn \__cmd_tmp:w #####1 #####2
1527       {
1528         \__cmd_split_start_item:
1529         \__cmd_split_end_item:n { ##1 #####1 {#####2} }
1530       }
1531       \__cmd_tl_mapthread_function:nnN {##2}
1532       { ##3 {#1} {#1} {#1} {#1} {#1} {#1} {#1} {#1} {#1} {#1} } \__cmd_tmp:w
1533       \__cmd_split_signature_loop:Nw
1534     }
1535   }
1536 \exp_args:NV \__cmd_tmp:w \c_novalue_tl

```

Minor wrinkle with the prefixes: they use `__cmd_split_add_item:n` instead of `__cmd_split_end_item:n` (add *vs.* end) because they are followed by an argument, so they can't end the item.

```

1537 \cs_new_protected:Npn \__cmd_show_prefix:Nw #1
1538 { \__cmd_split_add_item:n {#1} \__cmd_split_signature_loop:Nw }
1539 \cs_new_protected:Npn \__cmd_show_processor:Nw #1 #2
1540 { \__cmd_split_add_item:n { #1 {#2} } \__cmd_split_signature_loop:Nw }

```

And now the auxiliaries that store the strings to be printed. `__cmd_split_start_item:` starts an item from scratch, `__cmd_split_add_item:n` adds tokens to an item without adding a newline, and `__cmd_split_end_item:n` adds tokens, terminates the item with a newline, and steps the argument count.

```

1541 \cs_new_protected:Npn \__cmd_split_start_item:
1542 {
1543   \tl_set:Nx \l__cmd_tmpb_tl
1544   { ~ \c_space_tl \c_hash_str \int_use:N \l__cmd_current_arg_int : }
1545 }
1546 \cs_new_protected:Npn \__cmd_split_add_item:n #1
1547 { \tl_put_right:Nx \l__cmd_tmpb_tl { \tl_to_str:n {#1} } }
1548 \cs_new_protected:Npn \__cmd_split_end_item:n #1
1549 {
1550   \tl_put_right:Nx \l__cmd_tmppa_tl
1551   { \l__cmd_tmpb_tl \tl_to_str:n {#1} \iow_newline: }
1552   \tl_clear:N \l__cmd_tmpb_tl
1553   \int_incr:N \l__cmd_current_arg_int
1554 }

```

(End of definition for `__cmd_split_signature:n` and others.)

Not much to do regarding `latexrelease`: we could remove the entries from `\@showcommandlisthook`, but it doesn't seem worth it.

```

\__cmd_split_start_item: 1555 <latexrelease>\EndIncludeInRelease
\__cmd_split_add_item:n  1556 %
\__cmd_split_end_item:n  1557 <latexrelease>\IncludeInRelease{2020/10/01}{\__cmd_show:N (part 2)}%
                          1558 <latexrelease> {Support~\ShowCommand~in~ltxcmd}
                          1559 <latexrelease>\EndIncludeInRelease

```

1.8 Grabbing arguments

All of the grabbers follow the same basic pattern. The initial function stores in `\l__cmd_signature_tl` the code to grab further arguments, defines (the function in) `\l__cmd_fn_tl` that will grab the argument, and calls it.

Defining `\l__cmd_fn_tl` means determining whether to use `\cs_set:Npn` or `\cs_set_nopar:Npn`, and for optional arguments whether to skip spaces. Once the argument is found, `\l__cmd_fn_tl` calls `__cmd_add_arg:n`, responsible for calling processors and grabbing further arguments.

This uses the well-tested code of D-type arguments, skipping the peeking step because the b-type argument is always present, and adding a cleanup stage at the end by hijacking the signature. The clean-up consists of properly dealing with `\l__cmd_args_tl` and also putting back the `\end` that served as an end-delimiter: this `\end` receives the environment name as its argument and is run normally. The D-type code stores the argument found (body of the environment) as a brace group in `\l__cmd_args_tl` and depending on the presence of a prefix `!` we trim spaces or not before adding this braced argument into the saved `\l__cmd_args_tl`. The strange `\begin` control sequence is there for display purposes only: it has to look like `\begin` in the terminal but not to delimited arguments.

```

1560 \cs_new_protected:Npn \__cmd_grab_b:w
1561   { \__cmd_grab_b_aux:NNw \cs_set_protected_nopar:Npn \tl_trim_spaces:n }
1562 \cs_new_protected:Npn \__cmd_grab_b_long:w
1563   { \__cmd_grab_b_aux:NNw \cs_set_protected:Npn \tl_trim_spaces:n }
1564 \cs_new_protected:Npn \__cmd_grab_b_obey_spaces:w
1565   { \__cmd_grab_b_aux:NNw \cs_set_protected_nopar:Npn \exp_not:n }
1566 \cs_new_protected:Npn \__cmd_grab_b_long_obey_spaces:w
1567   { \__cmd_grab_b_aux:NNw \cs_set_protected:Npn \exp_not:n }
1568 \cs_new_protected:Npn \__cmd_grab_b_aux:NNw #1#2#3 \__cmd_run_code:
1569   {
1570     \__cmd_grab_D_aux:NNnNN \begin \end {#3} #1 \use_ii:nn
1571     \tl_put_left:Nn \l__cmd_signature_tl { \__cmd_grab_b_end:Nw #2 }
1572     \tl_set_eq:NN \l__cmd_saved_args_tl \l__cmd_args_tl
1573     \tl_clear:N \l__cmd_args_tl
1574     \exp_args:Nc \l__cmd_fn_tl { begin ~ }
1575   }
1576 \cs_new_protected:Npn \__cmd_grab_b_end:Nw #1#2 \__cmd_run_code:
1577   {
1578     \tl_set:Nx \l__cmd_args_tl
1579     {
1580       \exp_not:V \l__cmd_saved_args_tl
1581       { \exp_after:wN #1 \l__cmd_args_tl }
1582     }
1583     #2
1584     \__cmd_run_code:
1585     \end
1586   }

```

(End of definition for `__cmd_grab_b:w` and others.)

Collecting an environment body verbatim shares some ideas with the v-type grabber, and others with the standard `filecontents` environment. The start is to set the end-of-line to a predictable value and to deactivate the specials.

```

1587 \cs_new_protected:Npn \__cmd_grab_c:w #1 \__cmd_run_code:
1588   {
1589     \bool_set_false:N \l__cmd_obey_spaces_bool
1590     \__cmd_grab_c_start:n {#1}
1591   }
1592 \cs_new_protected:Npn \__cmd_grab_c_obey_spaces:w #1 \__cmd_run_code:

```

```

1593 {
1594   \bool_set_true:N \l__cmd_obey_spaces_bool
1595   \__cmd_grab_c_start:n {#1}
1596 }
1597 \cs_new_protected:Npn \__cmd_grab_c_start:n #1
1598 {
1599   \tl_set:Nn \l__cmd_signature_tl {#1}
1600   \group_begin:
1601     \tl_clear:N \l__cmd_v_arg_tl
1602     \tex_escapechar:D = 92 \scan_stop:
1603     \tex_endlinechar:D = '\^M \scan_stop:
1604     \cs_set_eq:NN \do \char_set_catcode_other:N
1605     \dospecials
1606     \char_set_catcode_other:n { '\^M }
1607     \__cmd_grab_c_first:w
1608 }

```

Notice here and below that we cannot use `\token_to_str:N \end` as that would have the wrong category codes for the letters.

```

1609 \group_begin:
1610   \char_set_catcode_other:N \^M %
1611   \cs_new_protected:Npn \__cmd_grab_c_first:w #1 \^M %
1612     { %
1613       \tl_if_blank:nF {#1} %
1614       { %
1615         \msg_warning:nnee { cmd } { chars-dropped-first-line } %
1616         { \exp_not:n {#1} } %
1617         { \exp_not:V \@currenvir } %
1618       } %
1619       \__cmd_grab_c_loop:w #1 \^M %
1620     } %
1621   \cs_new_protected:Npe \__cmd_grab_c_loop:w #1 \^M %
1622     { %
1623       \exp_not:N \__cmd_grab_c_auxi:w #1 %
1624       \c_backslash_str end %
1625       \scan_stop: %
1626     } %
1627 \group_end:

```

(End of definition for `__cmd_grab_c:w` and others.)

`__cmd_grab_c_auxi:w` We need to see if the current line contains `\end` followed by the name of the current environment. To do that and allow for spaces, we have to work stepwise. First, establish if there is an `\end` at all: remember that here we are dealing with “other” tokens. Whether these is an `\end` or not, the tokens *before* it form part of the line.

```

1628 \use:e
1629 {
1630   \cs_new_protected:Npn \exp_not:N \__cmd_grab_c_auxi:w
1631     #1 \c_backslash_str end #2 \scan_stop:
1632 }
1633 {
1634   \tl_put_right:Nn \l__cmd_v_arg_tl {#1}
1635   \tl_if_empty:nTF {#2}
1636     {

```



```

1637     \tl_put_right:Nn \l__cmd_v_arg_tl { \obeyedline }
1638     \__cmd_grab_c_loop:w
1639   }
1640   { \__cmd_grab_c_auxii:w #2 \scan_stop: }
1641 }

```

There is an `\end`, so we now remove the trailing marker we needed to do the test. This is stripped off, then we need to examine the rest of the line one token at a time: see `verbatim.dtx` for the inspiration. Notice that we use `^^M` here as the end marker: this allows looping to look for multiple `\end` entries in the line.

```

1642 \group_begin:
1643   \char_set_catcode_other:N \^^M %
1644   \use:e %
1645   { %
1646     \cs_new_protected:Npe \exp_not:N \__cmd_grab_c_auxii:w %
1647     #1 \c_backslash_str end \scan_stop: %
1648   } %
1649   { %
1650     \tl_set:Nn \exp_not:N \l__cmd_tmpa_tl
1651     { \c_backslash_str end } %
1652     \exp_not:N \__cmd_grab_c_auxiii:N #1 \^^M %
1653   } %

```

Within the line, we need to collect up the tokens: if we do not find the end-of-environment argument, we will need those to reinsert. There are three special cases here: `^^M` (end of line: tidy up and back to the main loop), `\` (possibly skip over) and `{` (start the inner loop). Anything else means we move back to examine the rest of the line for any more `\end` entries.

```

1654   \cs_new_protected:Npn \__cmd_grab_c_auxiii:N #1 %
1655   { %
1656     \token_case_charcode:NnF #1 %
1657     { %
1658       \^^M %
1659       { \__cmd_grab_c_auxiv: } %
1660       \c_space_token %
1661       { %
1662         \tl_put_right:Nn \l__cmd_tmpa_tl {#1} %
1663         \__cmd_grab_c_auxiii:N %
1664       } %
1665       \c_group_begin_token %
1666       { %
1667         \tl_set:Nn \l__cmd_tmpb_tl {#1} %
1668         \__cmd_grab_c_auxvi:N %
1669       } %
1670     } %
1671     { \__cmd_grab_c_auxv: } %
1672   } %
1673 \group_end:
1674 \cs_new_protected:Npn \__cmd_grab_c_auxiv:
1675 {
1676   \tl_put_right:Ne \l__cmd_v_arg_tl
1677   {
1678     \exp_not:V \l__cmd_tmpa_tl
1679     \exp_not:N \obeyedline

```

```

1680     }
1681     \__cmd_grab_c_loop:w
1682   }
1683 \cs_new_protected:Npn \__cmd_grab_c_auxv:
1684   {
1685     \tl_put_right:NV \l__cmd_v_arg_tl \l__cmd_tmpa_tl
1686     \__cmd_grab_c_loop:w
1687   }

```

In the inner loop, we again have only a few special cases. First, we could again have $\sim M$, in which case we tidy up using a common auxiliary. Second, we check for the escape char: this cannot happen inside the end of an environment and means we loop, re-inserting the token. Finally, we have $\}$, where we need to move on to check what has been collected. Otherwise, collect up and loop. Notice here that the inner loop needs to collect tokens separately: this leaves any spaces after \end in $\l__cmd_tmpa_tl$, so we can test $\l__cmd_tmpb_tl$ directly.

```

1688 \group_begin:
1689   \char_set_catcode_other:N \sim M %
1690   \cs_new_protected:Npe \__cmd_grab_c_auxvii:N #1 %
1691   { %
1692     \exp_not:N \token_case_charcode:NnF #1 %
1693     { %
1694       \sim M %
1695       {
1696         \exp_not:N \__cmd_grab_c_auxviii: %
1697         \exp_not:N \__cmd_grab_c_auxv: %
1698       }%
1699       \c_backslash_str %
1700       { %
1701         \exp_not:N \__cmd_grab_c_auxviii: %
1702         \exp_not:N \__cmd_grab_c_auxv: #1
1703       } %
1704       \c_group_end_token %
1705       { %
1706         \tl_put_right:Nn \exp_not:N \l__cmd_tmpb_tl {#1} %
1707         \exp_not:N \__cmd_grab_c_auxviiii: %
1708       } %
1709     } %
1710     { %
1711       \tl_put_right:Nn \exp_not:N \l__cmd_tmpb_tl {#1} %
1712       \exp_not:N \__cmd_grab_c_auxvii:N %
1713     } %
1714   } %
1715 \group_end: %
1716 \cs_new_protected:Npn \__cmd_grab_c_auxviii:
1717   { \tl_put_right:NV \l__cmd_tmpa_tl \l__cmd_tmpb_tl }
1718 \cs_new_protected:Npn \__cmd_grab_c_auxviiii:
1719   {
1720     \str_if_eq:eeTF { \exp_not:N \l__cmd_tmpb_tl } { { \@currenvir } }
1721     { \__cmd_grab_c_end:w }
1722     {
1723       \__cmd_grab_c_auxviii:
1724       \__cmd_grab_c_auxv:
1725     }

```

1726 }

(End of definition for `__cmd_grab_c_auxi:w` and others.)

`__cmd_grab_c_end:w` To end the collection, we clean up the last line: once again we need to find `^^M`. Once
`__cmd_grab_c_end:n` that is done, we can warn if there is anything left behind.

```
\__cmd_grab_c_end_auxi:w 1727 \group_begin:
\__cmd_grab_c_end_auxii:w 1728 \char_set_catcode_other:N \^^M %
\__cmd_grab_c_end_auxiii:w 1729 \cs_new_protected:Npn \__cmd_grab_c_end:w #1 \^^M %
1730 { %
1731 \tl_if_blank:nF {#1} %
1732 { %
1733 \msg_warning:nnee { cmd } { chars-dropped-last-line } %
1734 { \exp_not:n {#1} } %
1735 { \exp_not:V \@currenenvir } %
1736 } %
1737 \exp_args:NNNo \group_end: %
1738 \tl_set:Nn \l__cmd_v_arg_tl { \l__cmd_v_arg_tl } %
1739 \__cmd_add_arg:x %
1740 { %
1741 \bool_if:NTF \l__cmd_obey_spaces_bool %
1742 { \exp_not:V } %
1743 { \exp_args:NV \__cmd_grab_c_end:n } %
1744 \l__cmd_v_arg_tl %
1745 } %
1746 \exp_args:NV \end \@currenenvir %
1747 } %
1748 \group_end: %
```

Look for line markers at each end and tidy up if required.

```
1749 \cs_new:Npn \__cmd_grab_c_end:n #1
1750 {
1751 \__cmd_grab_c_end_auxi:w \q_nil #1 \q_nil
1752 \obeyedline \obeyedline \q_nil \q_stop
1753 }
1754 \cs_new:Npn \__cmd_grab_c_end_auxi:w #1 \q_nil \obeyedline
1755 { \__cmd_grab_c_end_auxii:w #1 \q_nil }
1756 \cs_new:Npn \__cmd_grab_c_end_auxii:w \q_nil #1 \obeyedline \q_nil
1757 { \__cmd_grab_c_end_auxiii:w #1 \q_nil }
1758 \cs_new:Npn \__cmd_grab_c_end_auxiii:w #1 \q_nil #2 \q_stop
1759 { \exp_not:n {#1} }
```

(End of definition for `__cmd_grab_c_end:w` and others.)

`__cmd_grab_D:w` The generic delimited argument grabber. The auxiliary function does a peek test before
`__cmd_grab_D_no_strip:w` calling `__cmd_grab_D_call:Nw`, so that the optional nature of the argument works as
`__cmd_grab_D_obey_spaces:w` expected.

```
\__cmd_grab_D_obey_spaces_no_strip:w 1760 \tl_map_inline:nn { { } { _long } }
\__cmd_grab_D_long:w 1761 {
\__cmd_grab_D_long_no_strip:w 1762 \tl_map_inline:nn { { } { _obey_spaces } }
\__cmd_grab_D_long_obey_spaces:w 1763 {
\__cmd_grab_D_long_obey_spaces_no_strip:w 1764 \tl_map_inline:nn { { } { _no_strip } }
1765 {
1766 \tl_map_inline:nn { { } { _verb_safe } }
1767 {
```

```

1768 \cs_new_protected:cpe { __cmd_grab_D #1 ##1 #####1 #####1 :w }
1769 #####1#####2#####3
1770 \__cmd_run_code:
1771 {
1772 \exp_not:N \__cmd_grab_D_aux:NNnNNNN
1773 #####1 #####2 {#####3}
1774 \str_if_eq:nnTF {#1} { _long }
1775 \cs_set_protected:Npn
1776 \cs_set_protected_nopar:Npn
1777 \str_if_eq:nnTF {##1} { _obey_spaces }
1778 { \exp_not:N \__cmd_peek_meaning_remove:NTF }
1779 { \exp_not:N \__cmd_peek_nonspace_remove:NTF }
1780 \str_if_eq:nnTF {####1} { _no_strip }
1781 { \exp_not:N \use_none:n }
1782 { \exp_not:N \use_ii:nn }
1783 \str_if_eq:nnTF {#####1} { _verb_safe }
1784 { \exp_not:N \use:n }
1785 { \exp_not:N \use_none:n }
1786 }
1787 }
1788 }
1789 }
1790 }

```

This is a bit complicated. The idea is that, in order to check for nested optional argument tokens ([...] and so on) the argument needs to be grabbed without removing any braces at all. If this is not done, then cases like [{}] fail. So after testing for an optional argument, it is collected piece-wise. Inserting a quark prevents loss of braces, and there is then a test to see if there are nested delimiters to handle.

```

1791 \group_begin:
1792 \char_set_catcode_other:N \^^M
1793 \cs_new_protected:Npn \__cmd_grab_D_aux:NNnNNNN #1#2#3#4#5#6#7
1794 {
1795 \__cmd_grab_D_aux:NNnNN #1#2 {#3} #4 #6
1796 #7
1797 {
1798 \group_begin:
1799 \__cmd_grab_D_verb_safe:NN #1#5
1800 }
1801 #5 #1
1802 {
1803 #7 { \group_end: }
1804 \__cmd_grab_D_call:Nw #1
1805 }
1806 {
1807 #7 { \group_end: }
1808 \__cmd_add_arg:o \c_novalue_tl
1809 }
1810 }
1811 \group_end:

```

The only awkwardness here is the need to preserve the catcode of the search token: this is done low-level for performance reasons. Only values that are realistic are included. As this does not cover spaces when they are skipped that has to be covered separately.

```

1812 \cs_new_protected:Npn \__cmd_grab_D_verb_safe:NN #1#2
1813 {
1814   \cs_set_eq:NN \do \char_set_catcode_other:N
1815   \dospecials
1816   \char_set_catcode_other:N \^^M
1817   \token_if_eq_meaning:NNT #2 \__cmd_peek_nonspace_remove:NTF
1818   { \char_set_catcode_space:n { '\ } }
1819   \use:c
1820   {
1821     char_set_catcode_
1822     \if_catcode:w \c_math_toggle_token #1 math_toggle \else:
1823     \if_catcode:w ^ #1 math_superscript \else:
1824     \if_catcode:w \c_math_subscript_token #1 math_subscript \else:
1825     \if_catcode:w A #1 letter \else:
1826     other \fi: \fi: \fi: \fi:
1827     :N
1828   }
1829   #1
1830 }

```

Inside the “standard” grabber, there is a test to see if the grabbed argument is entirely enclosed by braces. There are a couple of extra factors to allow for: the argument might be entirely empty, and spaces at the start and end of the input must be retained around a brace group. Also notice that a *blank* argument might still contain spaces. To allow for suppression of brace stripping, the business end is passed here as #5.

```

1831 \cs_new_protected:Npn \__cmd_grab_D_aux:NNnN #1#2#3#4#5
1832 {
1833   \tl_set:Nn \l__cmd_signature_tl {#3}
1834   \exp_after:wN #4 \l__cmd_fn_tl ##1 #2
1835   {
1836     \tl_if_in:nnTF {##1} {#1}
1837     { \__cmd_grab_D_nested:NNnN #1 #2 {##1} #4 }
1838     {
1839       \tl_if_blank:oTF { \use_none:n ##1 }
1840       { \__cmd_add_arg:o { \use_none:n ##1 } }
1841       {
1842         \str_if_eq:eeTF
1843         { \exp_not:o { \use_none:n ##1 } }
1844         { { \exp_not:o { \use_ii:nnn ##1 \q_nil } } }
1845         { \__cmd_add_arg:o { #5 ##1 } }
1846         { \__cmd_add_arg:o { \use_none:n ##1 } }
1847       }
1848     }
1849   }
1850 }

```

(End of definition for `__cmd_grab_D:w` and others.)

<pre> __cmd_grab_D_nested:NNnN __cmd_grab_D_nested:w \l__cmd_nesting_a_tl \l__cmd_nesting_b_tl \q__cmd </pre>	<p>Catching nested optional arguments means more work. The aim here is to collect up each pair of optional tokens without T_EX helping out, and without counting anything. The code above will already have removed the leading opening token and a closing token, but the wrong one. The aim is then to work through the material grabbed so far and divide it up on each opening token, grabbing a closing token to match (thus working in pairs). Once there are no opening tokens, then there is a second check to see if there</p>
---	---

are any opening tokens in the second part of the argument (for things like `[] []`). Once everything has been found, the entire collected material is added to the output as a single argument. The only tricky part here is ensuring that any grabbing function that might run away is named after the function currently being parsed and not after `xparse`. That leads to some rather complex nesting! There is also a need to prevent the loss of any braces, hence the insertion and removal of quarks along the way.

```

1851 \tl_new:N \l__cmd_nesting_a_tl
1852 \tl_new:N \l__cmd_nesting_b_tl
1853 \quark_new:N \q__cmd
1854 \cs_new_protected:Npn \__cmd_grab_D_nested:NNnN #1#2#3#4
1855 {
1856   \tl_clear:N \l__cmd_nesting_a_tl
1857   \tl_clear:N \l__cmd_nesting_b_tl
1858   \exp_after:wN #4 \l__cmd_fn_tl ##1 #1 ##2 \q__cmd ##3 #2
1859   {
1860     \tl_put_right:No \l__cmd_nesting_a_tl { \use_none:n ##1 #1 }
1861     \tl_put_right:No \l__cmd_nesting_b_tl { \use_i:nn #2 ##3 }
1862     \tl_if_in:nnTF {##2} {#1}
1863     {
1864       \l__cmd_fn_tl
1865       \q_nil ##2 \q__cmd \ERROR
1866     }
1867     {
1868       \tl_put_right:Nx \l__cmd_nesting_a_tl
1869       { \__cmd_grab_D_nested:w \q_nil ##2 \q_stop }
1870       \tl_if_in:NnTF \l__cmd_nesting_b_tl {#1}
1871       {
1872         \tl_set_eq:NN \l__cmd_tmpa_tl \l__cmd_nesting_b_tl
1873         \tl_clear:N \l__cmd_nesting_b_tl
1874         \exp_after:wN \l__cmd_fn_tl \exp_after:wN
1875         \q_nil \l__cmd_tmpa_tl \q_nil \q__cmd \ERROR
1876       }
1877       {
1878         \tl_put_right:No \l__cmd_nesting_a_tl
1879         \l__cmd_nesting_b_tl
1880         \__cmd_add_arg:V \l__cmd_nesting_a_tl
1881       }
1882     }
1883   }
1884   \l__cmd_fn_tl #3 \q_nil \q__cmd \ERROR
1885 }
1886 \cs_new:Npn \__cmd_grab_D_nested:w #1 \q_nil \q_stop
1887 { \exp_not:o { \use_none:n #1 } }

```

(End of definition for `__cmd_grab_D_nested:NNnN` and others.)

`__cmd_grab_D_call:Nw` For D and R-type arguments, to avoid losing any braces, a token needs to be inserted before the argument to be grabbed. If the argument runs away because the closing token is missing then this inserted token shows up in the terminal. Ideally, `#1` would therefore be used directly, but that is no good as it will mess up the rest of the grabber. Instead, a copy of `#1` with an altered category code is used, as this will look right in the terminal but will not mess up the grabber. The only issue then is that the category code of `#1` is unknown. So there is a quick test to ensure that the inserted token can never be

matched by the grabber. (This assumes that the open and close delimiters are not the same character with different category codes, but that really should not happen in any sensible document-level syntax.) An exception is when #1 is a control sequence token, in which case the character-token treatment is no good because if hit with `\token_to_str:N` it would add sputios tokens to the argument. In this case a different branch is taken. The token inserted is then the same $\langle csname \rangle$ as #1, but with a space appended, so that the grabber don't see it as another of the same delimiter.

```

1888 \cs_new_protected_nopar:Npn \__cmd_grab_D_call:Nw #1
1889   {
1890     \token_if_eq_catcode:NNTF + #1
1891     {
1892       \exp_after:wN \exp_after:wN \exp_after:wN
1893       \l__cmd_fn_tl \char_generate:nn { '#1 } { 11 }
1894     }
1895     {
1896       \__cmd_token_if_cs:NNTF #1
1897       {
1898         \exp_after:wN \l__cmd_fn_tl
1899         \cs:w \cs_to_str:N #1 ~ \cs_end:
1900       }
1901       {
1902         \exp_after:wN \l__cmd_fn_tl
1903         \token_to_str:N #1
1904       }
1905     }
1906   }

```

(End of definition for __cmd_grab_D_call:Nw.)

`__cmd_grab_E:w` Everything here needs to point to a loop.

```

\__cmd_grab_E_long:w
\__cmd_grab_E_obey_spaces:w
\__cmd_grab_E_long_obey_spaces:w
\__cmd_grab_E:nnNN
\__cmd_grab_E_loop:NnN
\__cmd_grab_E_finalise:
1907 \cs_new_protected:Npn \__cmd_grab_E:w #1#2 \__cmd_run_code:
1908   {
1909     \__cmd_grab_E:nnNN {#1} {#2}
1910     \cs_set_protected_nopar:Npn
1911     \__cmd_peek_nonspace_remove:NNTF
1912   }
1913 \cs_new_protected:Npn \__cmd_grab_E_long:w #1#2 \__cmd_run_code:
1914   {
1915     \__cmd_grab_E:nnNN {#1} {#2}
1916     \cs_set_protected:Npn
1917     \__cmd_peek_nonspace_remove:NNTF
1918   }
1919 \cs_new_protected:Npn \__cmd_grab_E_obey_spaces:w #1#2 \__cmd_run_code:
1920   {
1921     \__cmd_grab_E:nnNN {#1} {#2}
1922     \cs_set_protected_nopar:Npn
1923     \__cmd_peek_meaning_remove:NNTF
1924   }
1925 \cs_new_protected:Npn \__cmd_grab_E_long_obey_spaces:w #1#2 \__cmd_run_code:
1926   {
1927     \__cmd_grab_E:nnNN {#1} {#2}
1928     \cs_set_protected:Npn
1929     \__cmd_peek_meaning_remove:NNTF
1930   }

```

A loop is needed here to allow a random ordering of keys. These are searched for one at a time, with any not found needing to be tracked: they can appear later. The grabbed values are held in a property list which is then turned into an ordered list to be passed back to the user.

```

1931 \cs_new_protected:Npn \__cmd_grab_E:nmNN #1#2#3#4
1932 {
1933   \exp_after:wN #3 \l__cmd_fn_tl ##1##2##3
1934   {
1935     \prop_put:Nnn \l__cmd_tmp_prop {##1} {##3}
1936     \__cmd_grab_E_loop:NnN #4 { } ##2 \q_recursion_stop
1937   }
1938   \prop_clear:N \l__cmd_tmp_prop
1939   \tl_set:Nn \l__cmd_signature_tl {#2}
1940   \cs_set_protected:Npn \__cmd_grab_E_finalise:
1941   {
1942     \tl_map_inline:nn {#1}
1943     {
1944       \prop_get:NnNF \l__cmd_tmp_prop {####1} \l__cmd_tmpb_tl
1945       { \tl_set_eq:NN \l__cmd_tmpb_tl \c_novalue_tl }
1946       \tl_put_right:Nx \l__cmd_args_tl
1947       { { \exp_not:V \l__cmd_tmpb_tl } }
1948     }
1949     \l__cmd_signature_tl \__cmd_run_code:
1950   }
1951   \__cmd_grab_E_loop:NnN #4 { } #1 \q_recursion_tail \q_recursion_stop
1952 }
1953 \cs_new_protected:Npn \__cmd_grab_E_loop:NnN #1#2#3#4 \q_recursion_stop
1954 {
1955   \cs_if_eq:NNTF #3 \q_recursion_tail
1956   { \__cmd_grab_E_finalise: }
1957   {
1958     #1 #3
1959     { \l__cmd_fn_tl #3 {#2#4} }
1960     { \__cmd_grab_E_loop:NnN #1 {#2#3} #4 \q_recursion_stop }
1961   }
1962 }
1963 \cs_new_protected:Npn \__cmd_grab_E_finalise: { }

```

(End of definition for __cmd_grab_E:w and others.)

```

\__cmd_grab_m:w Collecting a single mandatory argument is quite easy.
\__cmd_grab_m_long:w
1964 \cs_new_protected:Npn \__cmd_grab_m:w #1 \__cmd_run_code:
1965 {
1966   \tl_set:Nn \l__cmd_signature_tl {#1}
1967   \exp_after:wN \cs_set_protected_nopar:Npn \l__cmd_fn_tl ##1
1968   { \__cmd_add_arg:n {##1} }
1969   \l__cmd_fn_tl
1970 }
1971 \cs_new_protected:Npn \__cmd_grab_m_long:w #1 \__cmd_run_code:
1972 {
1973   \tl_set:Nn \l__cmd_signature_tl {#1}
1974   \exp_after:wN \cs_set_protected:Npn \l__cmd_fn_tl ##1
1975   { \__cmd_add_arg:n {##1} }
1976   \l__cmd_fn_tl

```


1977 }

(End of definition for _cmd_grab_m:w and _cmd_grab_m_long:w.)

_cmd_grab_m_1:w Grabbing 1–8 mandatory arguments is done by giving 8–1 known arguments to a 9-
_cmd_grab_m_2:w argument function that stores them in \l__cmd_args_tl. For simplicity, grabbing 9
_cmd_grab_m_3:w mandatory arguments is done by grabbing 5 then 4 arguments.

_cmd_grab_m_4:w 1978 \cs_new_protected_nopar:Npn _cmd_grab_m_aux:Nnnnnnnnn #1#2#3#4#5#6#7#8#9

_cmd_grab_m_5:w 1979 {
_cmd_grab_m_6:w 1980 \tl_put_right:No \l__cmd_args_tl
_cmd_grab_m_7:w 1981 { #1 {#2} {#3} {#4} {#5} {#6} {#7} {#8} {#9} }

_cmd_grab_m_8:w 1982 \l__cmd_signature_tl _cmd_run_code:

_cmd_grab_m_9:w 1983 }

_cmd_grab_m_aux:Nnnnnnnnn 1984 \cs_new_protected:cpn { __cmd_grab_m_1:w } #1 _cmd_run_code:

1985 {
1986 \tl_set:Nn \l__cmd_signature_tl {#1}
1987 \exp_after:wN \cs_set_eq:NN \l__cmd_fn_tl _cmd_grab_m_aux:Nnnnnnnnn
1988 \l__cmd_fn_tl \use_none:nnnnnn { } { } { } { } { } { } { }
1989 }

1990 \cs_new_protected:cpn { __cmd_grab_m_2:w } #1 _cmd_run_code:

1991 {
1992 \tl_set:Nn \l__cmd_signature_tl {#1}
1993 \exp_after:wN \cs_set_eq:NN \l__cmd_fn_tl _cmd_grab_m_aux:Nnnnnnnnn
1994 \l__cmd_fn_tl \use_none:nnnnnn { } { } { } { } { } { } { }
1995 }

1996 \cs_new_protected:cpn { __cmd_grab_m_3:w } #1 _cmd_run_code:

1997 {
1998 \tl_set:Nn \l__cmd_signature_tl {#1}
1999 \exp_after:wN \cs_set_eq:NN \l__cmd_fn_tl _cmd_grab_m_aux:Nnnnnnnnn
2000 \l__cmd_fn_tl \use_none:nnnnnn { } { } { } { } { } { }
2001 }

2002 \cs_new_protected:cpn { __cmd_grab_m_4:w } #1 _cmd_run_code:

2003 {
2004 \tl_set:Nn \l__cmd_signature_tl {#1}
2005 \exp_after:wN \cs_set_eq:NN \l__cmd_fn_tl _cmd_grab_m_aux:Nnnnnnnnn
2006 \l__cmd_fn_tl \use_none:nnnn { } { } { } { }
2007 }

2008 \cs_new_protected:cpn { __cmd_grab_m_5:w } #1 _cmd_run_code:

2009 {
2010 \tl_set:Nn \l__cmd_signature_tl {#1}
2011 \exp_after:wN \cs_set_eq:NN \l__cmd_fn_tl _cmd_grab_m_aux:Nnnnnnnnn
2012 \l__cmd_fn_tl \use_none:nnn { } { } { }
2013 }

2014 \cs_new_protected:cpn { __cmd_grab_m_6:w } #1 _cmd_run_code:

2015 {
2016 \tl_set:Nn \l__cmd_signature_tl {#1}
2017 \exp_after:wN \cs_set_eq:NN \l__cmd_fn_tl _cmd_grab_m_aux:Nnnnnnnnn
2018 \l__cmd_fn_tl \use_none:nn { } { }
2019 }

2020 \cs_new_protected:cpn { __cmd_grab_m_7:w } #1 _cmd_run_code:

2021 {
2022 \tl_set:Nn \l__cmd_signature_tl {#1}
2023 \exp_after:wN \cs_set_eq:NN \l__cmd_fn_tl _cmd_grab_m_aux:Nnnnnnnnn
2024 \l__cmd_fn_tl \use_none:n { }

```

2025 }
2026 \cs_new_protected:cpn { __cmd_grab_m_8:w } #1 \__cmd_run_code:
2027 {
2028   \tl_set:Nn \l__cmd_signature_tl {#1}
2029   \exp_after:wN \cs_set_eq:NN \l__cmd_fn_tl \__cmd_grab_m_aux:Nnnnnnnn
2030   \l__cmd_fn_tl \prg_do_nothing:
2031 }
2032 \cs_new_protected:cpx { __cmd_grab_m_9:w }
2033 {
2034   \exp_not:c { __cmd_grab_m_5:w }
2035   \exp_not:c { __cmd_grab_m_4:w }
2036 }

```

(End of definition for __cmd_grab_m_1:w and others.)

__cmd_grab_R:w The grabber for R-type arguments is basically the same as that for D-type ones, but
 __cmd_grab_R_long:w always skips spaces (as it is mandatory) and has a hard-coded error message.
 __cmd_grab_R_aux:NNnN

```

2037 \cs_new_protected:Npn \__cmd_grab_R:w #1#2#3 \__cmd_run_code:
2038 { \__cmd_grab_R_aux:NNnN #1 #2 {#3} \cs_set_protected_nopar:Npn }
2039 \cs_new_protected:Npn \__cmd_grab_R_long:w #1#2#3 \__cmd_run_code:
2040 { \__cmd_grab_R_aux:NNnN #1 #2 {#3} \cs_set_protected:Npn }
2041 \cs_new_protected:Npn \__cmd_grab_R_aux:NNnN #1#2#3#4
2042 {
2043   \__cmd_grab_D_aux:NNnN #1 #2 {#3} #4 \use_ii:nn
2044   \__cmd_peek_nonspace_remove:NTF #1
2045   { \__cmd_grab_D_call:Nw #1 }
2046   {
2047     \msg_error:nxxx { cmd } { missing-required }
2048     { \__cmd_environment_or_command: }
2049     { \token_to_str:N #1 }
2050     \__cmd_add_arg:o \c_novalue_tl
2051   }
2052 }

```

(End of definition for __cmd_grab_R:w, __cmd_grab_R_long:w, and __cmd_grab_R_aux:NNnN.)

__cmd_grab_t:w Dealing with a token is quite easy. Check the match, remove the token if needed and add
 __cmd_grab_t_obey_spaces:w a flag to the output.
 __cmd_grab_t_aux:NNw

```

2053 \cs_new_protected:Npn \__cmd_grab_t:w
2054 { \__cmd_grab_t_aux:NNw \__cmd_peek_nonspace_remove:NTF }
2055 \cs_new_protected:Npn \__cmd_grab_t_obey_spaces:w
2056 { \__cmd_grab_t_aux:NNw \__cmd_peek_meaning_remove:NTF }
2057 \cs_new_protected:Npn \__cmd_grab_t_aux:NNw #1#2#3 \__cmd_run_code:
2058 {
2059   \tl_set:Nn \l__cmd_signature_tl {#3}
2060   \exp_after:wN \cs_set_protected:Npn \l__cmd_fn_tl
2061   {
2062     #1 #2
2063     { \__cmd_add_arg:n { \BooleanTrue } }
2064     { \__cmd_add_arg:n { \BooleanFalse } }
2065   }
2066   \l__cmd_fn_tl
2067 }

```

(End of definition for `__cmd_grab_t:w`, `__cmd_grab_t_obey_spaces:w`, and `__cmd_grab_t_aux:NNw`.)

`\l__cmd_v_arg_tl`

2068 `\tl_new:N \l__cmd_v_arg_tl`

`__cmd_grab_v:w`
`__cmd_grab_v_long:w`
`__cmd_grab_v_aux:w`
`__cmd_grab_v_group_end:`

Firstly, it is necessary to change `\tex_endlinechar:D` so that newlines in different catcode regimes (e.g., `\ExplSyntaxOn`) are not misinterpreted as spaces. The opening delimiter is the first non-space token, and is never read verbatim. This is required by consistency with the case where the preceding argument was optional and absent: then \TeX has already read and tokenized that token when looking for the optional argument. The first thing is thus to check is that this delimiter is a character, and to distinguish the case of a left brace (in that case, `\group_align_safe_end:` is needed to compensate for the begin-group character that was just seen). Then set verbatim catcodes with `__cmd_grab_v_aux_catcodes:`.

The group keep catcode changes local, and `\group_align_safe_begin/end:` allow to use a character with category code 4 (normally `&`) as the delimiter (all commands do `\group_align_safe_begin/end:`, so there's no need to do that again here). It is ended by `__cmd_grab_v_group_end:`, which smuggles the collected argument out of the group.

```

2069 \cs_new_protected:Npn \__cmd_grab_v:w
2070   {
2071     \bool_set_false:N \l__cmd_long_bool
2072     \__cmd_grab_v_aux:w
2073   }
2074 \cs_new_protected:Npn \__cmd_grab_v_long:w
2075   {
2076     \bool_set_true:N \l__cmd_long_bool
2077     \__cmd_grab_v_aux:w
2078   }
2079 \cs_new_protected:Npn \__cmd_grab_v_aux:w #1 \__cmd_run_code:
2080   {
2081     \tl_set:Nn \l__cmd_signature_tl {#1}
2082     \group_begin:
2083       \tex_escapechar:D = 92 \scan_stop:
2084       \tex_endlinechar:D = '\^~M \scan_stop:
2085       \tl_clear:N \l__cmd_v_arg_tl
2086       \peek_remove_spaces:n
2087         {
2088           \peek_meaning_remove:NTF \c_group_begin_token
2089             {
2090               \group_align_safe_end:
2091               \__cmd_grab_v_bgroup:
2092             }
2093           {
2094             \peek_N_type:TF
2095               { \__cmd_grab_v_aux_test:N }
2096               { \__cmd_grab_v_aux_abort:n { } }
2097           }
2098         }
2099   }
2100 \cs_new_protected:Npn \__cmd_grab_v_group_end:
2101   {

```

```

2102     \exp_args:NNNo
2103     \group_end:
2104     \tl_set:Nn \l__cmd_v_arg_tl { \l__cmd_v_arg_tl }
2105 }

```

(End of definition for `__cmd_grab_v:w` and others.)

`__cmd_grab_v_aux_test:N` Check that the opening delimiter is a character, setup category codes, then start reading tokens one by one, keeping the delimiter as an argument. If the verbatim was not nested, we will be grabbing one character at each step. Unfortunately, it can happen that what follows the verbatim argument is already tokenized. Thus, we check at each step that the next token is indeed a “nice” character, *i.e.*, is not a character with category code 1 (begin-group), 2 (end-group) or 6 (macro parameter), nor the space character, with category code 10 and character code 32, nor a control sequence. The partially built argument is stored in `\l__cmd_v_arg_tl`. If we ever meet a token which we cannot grab (non-N-type), or which is not a character according to `__cmd_grab_v_token_if_char:NTF`, then we bail out with `__cmd_grab_v_aux_abort:n`. Otherwise, we stop at the first character matching the delimiter.

```

2106 \cs_new_protected:Npn \__cmd_grab_v_aux_test:N #1
2107 {
2108     \__cmd_grab_v_token_if_char:NTF #1
2109     {
2110         \__cmd_grab_v_aux_put:N #1
2111         \__cmd_grab_v_aux_catcodes:
2112         \__cmd_grab_v_aux_loop:N #1
2113     }
2114     { \__cmd_grab_v_aux_abort:n {#1} #1 }
2115 }
2116 \cs_new_protected:Npn \__cmd_grab_v_aux_loop:N #1
2117 {
2118     \peek_N_type:TF
2119     { \__cmd_grab_v_aux_loop:NN #1 }
2120     { \__cmd_grab_v_aux_abort:n { } }
2121 }
2122 \cs_new_protected:Npn \__cmd_grab_v_aux_loop:NN #1#2
2123 {
2124     \__cmd_grab_v_token_if_char:NTF #2
2125     {
2126         \token_if_eq_charcode:NNTF #1 #2
2127         { \__cmd_grab_v_aux_loop_end: }
2128         {
2129             \__cmd_grab_v_aux_put:N #2
2130             \__cmd_grab_v_aux_loop:N #1
2131         }
2132     }
2133     { \__cmd_grab_v_aux_abort:n {#2} #2 }
2134 }
2135 \cs_new_protected:Npn \__cmd_grab_v_aux_loop_end:
2136 {
2137     \__cmd_grab_v_group_end:
2138     \__cmd_add_arg:x { \tl_tail:N \l__cmd_v_arg_tl }
2139 }

```

(End of definition for `__cmd_grab_v_aux_test:N` and others.)

\l__cmd_v_nesting_int

2140 \int_new:N \l__cmd_v_nesting_int

__cmd_grab_v_bgroup: If the opening delimiter is a left brace, we keep track of how many left and right braces
__cmd_grab_v_bgroup_loop: were encountered so far in \l__cmd_v_nesting_int (the methods used for optional ar-
__cmd_grab_v_bgroup_loop:N guments cannot apply here), and stop as soon as it reaches 0.

Some care was needed when removing the opening delimiter, which has already been assigned category code 1: using \peek_meaning_remove:NTF in the __cmd_grab_v_aux:w function would break within alignments. Instead, we first convert that token to a string, and remove the result as a normal undelimited argument.

```
2141 \cs_new_protected:Npx \__cmd_grab_v_bgroup:
2142 {
2143   \exp_not:N \__cmd_grab_v_aux_catcodes:
2144   \exp_not:n { \int_set:Nn \l__cmd_v_nesting_int { 1 } }
2145   \exp_not:N \__cmd_grab_v_aux_put:N \iow_char:N \{
2146   \exp_not:N \__cmd_grab_v_bgroup_loop:
2147 }
2148 \cs_new_protected:Npn \__cmd_grab_v_bgroup_loop:
2149 {
2150   \peek_N_type:TF
2151   { \__cmd_grab_v_bgroup_loop:N }
2152   { \__cmd_grab_v_aux_abort:n { } }
2153 }
2154 \cs_new_protected:Npn \__cmd_grab_v_bgroup_loop:N #1
2155 {
2156   \__cmd_grab_v_token_if_char:NTF #1
2157   {
2158     \token_if_eq_charcode:NNTF \c_group_end_token #1
2159     {
2160       \int_decr:N \l__cmd_v_nesting_int
2161       \int_compare:nNnTF \l__cmd_v_nesting_int > 0
2162       {
2163         \__cmd_grab_v_aux_put:N #1
2164         \__cmd_grab_v_bgroup_loop:
2165       }
2166       { \__cmd_grab_v_aux_loop_end: }
2167     }
2168     {
2169       \token_if_eq_charcode:NNT \c_group_begin_token #1
2170       { \int_incr:N \l__cmd_v_nesting_int }
2171       \__cmd_grab_v_aux_put:N #1
2172       \__cmd_grab_v_bgroup_loop:
2173     }
2174   }
2175   { \__cmd_grab_v_aux_abort:n {#1} #1 }
2176 }
```

(End of definition for __cmd_grab_v_bgroup:, __cmd_grab_v_bgroup_loop:, and __cmd_grab_v_bgroup_loop:N.)

__cmd_grab_v_aux_catcodes: The approach for short verbatim arguments is to make the end-line character a macro
__cmd_grab_v_aux_abort:n parameter character: this is forbidden by the rest of the code. Then the error branch can check what caused the bail out and give the appropriate error message.

```

2177 <latexrelease> \IncludeInRelease{2025/06/01}{\__cmd_grab_v_aux_catcodes}%
2178 <latexrelease> {Active-spaces-and-tabs}
2179 \cs_new_protected:Npn \__cmd_grab_v_aux_catcodes:
2180 {
2181   \cs_set_eq:NN \do \char_set_catcode_other:N
2182   \dospecials
2183   \char_set_catcode_active:n { \ }
2184   \char_set_catcode_active:n { \^I }
2185   \bool_if:NTF \l__cmd_long_bool
2186     { \char_set_catcode_other:n { \tex_endlinechar:D } }
2187     { \char_set_catcode_parameter:n { \tex_endlinechar:D } }
2188 }
2189 <latexrelease> \EndIncludeInRelease
2190 <latexrelease> \IncludeInRelease{2020/10/01}{\__cmd_grab_v_aux_catcodes}%
2191 <latexrelease> {Active-spaces-and-tabs}
2192 <latexrelease> \cs_new_protected:Npn \__cmd_grab_v_aux_catcodes:
2193 <latexrelease> {
2194 <latexrelease>   \cs_set_eq:NN \do \char_set_catcode_other:N
2195 <latexrelease>   \dospecials
2196 <latexrelease>   \bool_if:NTF \l__cmd_long_bool
2197 <latexrelease>     { \char_set_catcode_other:n { \tex_endlinechar:D } }
2198 <latexrelease>     { \char_set_catcode_parameter:n { \tex_endlinechar:D } }
2199 <latexrelease> }
2200 <latexrelease> \EndIncludeInRelease
2201 \cs_new_protected:Npn \__cmd_grab_v_aux_abort:n #1
2202 {
2203   \__cmd_grab_v_group_end:
2204   \exp_after:wN \exp_after:wN \exp_after:wN
2205     \peek_meaning_remove:NTF \char_generate:nn { \tex_endlinechar:D } { 6 }
2206     {
2207       \msg_error:nnxxx { cmd } { verbatim-nl }
2208       { \__cmd_environment_or_command: }
2209       { \tl_to_str:N \l__cmd_v_arg_tl }
2210       { \tl_to_str:n {#1} }
2211       \__cmd_add_arg:o \c_novalue_tl
2212     }
2213     {
2214       \msg_error:nnxxx { cmd } { verbatim-tokenized }
2215       { \__cmd_environment_or_command: }
2216       { \tl_to_str:N \l__cmd_v_arg_tl }
2217       { \tl_to_str:n {#1} }
2218       \__cmd_add_arg:o \c_novalue_tl
2219     }
2220 }

```

(End of definition for __cmd_grab_v_aux_catcodes: and __cmd_grab_v_aux_abort:n.)

__cmd_grab_v_aux_put:N Storing one token in the collected argument: everything as-is except for end-of-lines, with \exp_not:N to handle actives.

```

2221 <latexrelease> \IncludeInRelease{2025-06-01}{\__cmd_grab_v_aux_put:N}%
2222 <latexrelease> {Use-more-std-catcodes}
2223 \cs_new_protected:Npn \__cmd_grab_v_aux_put:N #1
2224 {
2225   \tl_put_right:Nx \l__cmd_v_arg_tl

```

```

2226 {
2227   \int_compare:nNnTF {'#1} = \tex_endlinechar:D
2228     { \exp_not:N \obeyedline }
2229     { \exp_not:N #1 }
2230 }
2231 }
2232 \langle latexrelease \rangle \EndIncludeInRelease
2233 \langle latexrelease \rangle \IncludeInRelease{2024/06/01}{\__cmd_grab_v_aux_put:N}%
2234 \langle latexrelease \rangle {Endlines~as~\obeyedline}
2235 \langle latexrelease \rangle \cs_new_protected:Npn \__cmd_grab_v_aux_put:N #1
2236 \langle latexrelease \rangle {
2237   \tl_put_right:Nx \l__cmd_v_arg_tl
2238   {
2239     \token_if_active:NTF #1
2240       { \exp_not:N #1 }
2241       {
2242         \int_compare:nNnTF {'#1} = \tex_endlinechar:D
2243           { \exp_not:N \obeyedline }
2244           { \token_to_str:N #1 }
2245         }
2246       }
2247   }
2248 \langle latexrelease \rangle \EndIncludeInRelease
2249 \langle latexrelease \rangle \IncludeInRelease{2020/10/01}{\__cmd_grab_v_aux_put:N}%
2250 \langle latexrelease \rangle {Endlines~as~\obeyedline}
2251 \langle latexrelease \rangle \cs_new_protected:Npn \__cmd_grab_v_aux_put:N #1
2252 \langle latexrelease \rangle {
2253   \tl_put_right:Nx \l__cmd_v_arg_tl
2254   {
2255     \token_if_active:NTF #1
2256       { \exp_not:N #1 } { \token_to_str:N #1 }
2257   }
2258 \langle latexrelease \rangle }
2259 \langle latexrelease \rangle \EndIncludeInRelease

```

(End of definition for __cmd_grab_v_aux_put:N.)

__cmd_grab_v_token_if_char:NTF

This function assumes that the escape character is printable. Then the string representation of control sequences is at least two characters, and \str_tail:n only removes the escape character. Macro parameter characters are doubled by \tl_to_str:n, and will also yield a non-empty result, hence are not considered as characters.

```

2260 \cs_new_protected:Npn \__cmd_grab_v_token_if_char:NTF #1
2261   { \str_if_eq:eeTF { } { \str_tail:n {#1} } }

```

(End of definition for __cmd_grab_v_token_if_char:NTF.)

__cmd_add_arg:n

When an argument is found it is stored, then further arguments are grabbed by calling \l__cmd_signature_tl.

__cmd_add_arg:V

__cmd_add_arg:o

__cmd_add_arg:x

```

2262 \cs_new_protected:Npn \__cmd_add_arg:n #1
2263   {
2264     \tl_put_right:Nn \l__cmd_args_tl { {#1} }
2265     \l__cmd_signature_tl \__cmd_run_code:
2266   }
2267 \cs_generate_variant:Nn \__cmd_add_arg:n { V , o , x }

```

(End of definition for __cmd_add_arg:n.)

1.9 Grabbing arguments expandably

```

\__cmd_expandable_grab_D:w
  \_cmd_expandable_grab_D:NNwNNn
    \_cmd_expandable_grab_D:NNwNNnnn
\__cmd_expandable_grab_D:Nw
  \_cmd_expandable_grab_D:nnNNwNN

```

The first step is to grab the first token or group. The generic grabbers $\langle function \rangle_{\sqcup}$ and $\langle function \rangle_{\sqcup}$ are just after $\backslash q_{_} _ cmd$, we go and find them (and use the long one).

```

2268 \cs_new:Npn \__cmd_expandable_grab_D:w #1 \q_ _cmd #2#3
2269   { #2 { \__cmd_expandable_grab_D:NNwNNn #1 \q_ _cmd #2 #3 } }

```

We then wish to test whether $\#7$, which we just grabbed, is exactly $\#2$. A preliminary test is whether their string representations coincide, then expand the only grabber function we have, $\#1$, once: the two strings below are equal if and only if $\#7$ matches $\#2$ exactly.² The preliminary test is needed as $\#7$ could validly contain $\backslash par$ (because a later mandatory argument could be long) and our grabber may be short. If $\#7$ does not match $\#2$, then the optional argument is missing, we use the default $-NoValue-$, and put back the argument $\#7$ in the input stream.

If it does match, then interesting things need to be done. We will grab the argument piece by piece, with the following pattern:

```

\grabber \{tokens\}
\q_nil \{piece 1\} \{piece 2\} \ERROR \q_ _cmd
\q_nil \input stream

```

The $\langle grabber \rangle$ will find an opening delimiter in $\langle piece 2 \rangle$, take the $\backslash q_{_} _ cmd$ as a second delimiter, and find more material delimited by the closing delimiter in the $\langle input stream \rangle$. We then move the part before the opening delimiter from $\langle piece 2 \rangle$ to $\langle piece 1 \rangle$, and the material taken from the $\langle input stream \rangle$ to the $\langle piece 2 \rangle$. Thus, the argument moves gradually from the $\langle input stream \rangle$ to the $\langle piece 2 \rangle$, then to the $\langle piece 1 \rangle$ when we have made sure to find all opening and closing delimiters. This two-step process ensures that nesting works: the number of opening delimiters minus closing delimiters in $\langle piece 1 \rangle$ is always equal to the number of closing delimiters in $\langle piece 2 \rangle$. We stop grabbing arguments once the $\langle piece 2 \rangle$ contains no opening delimiter any more, hence the balance is reached, and the final argument is $\langle piece 1 \rangle \langle piece 2 \rangle$. The indirection via $\backslash _ cmd _ tmp:w$ allows to insert $-NoValue-$ expanded.

```

2270 \cs_set_protected:Npn \__cmd_tmp:w #1
2271   {
2272     \cs_new:Npn \__cmd_expandable_grab_D:NNwNNn ##1##2##3##4 \q_ _cmd ##5##6##7
2273     {
2274       \str_if_eq:nnTF {##2} {##7}
2275       {
2276         \str_if_eq:onTF
2277           { ##1 { } { } ##7 ##2 \q_ _cmd ##3 }
2278           { { } {##2} { } }
2279       }
2280       { \use_ii:nn }
2281     }
2282     ##1
2283     { \__cmd_expandable_grab_D:NNwNNnnn ##1##2##3##4 \q_ _cmd ##5##6 }

```

²It is obvious that if $\#7$ matches $\#2$ then the strings are equal. We must check the converse. The right-hand-side of $\backslash str_if_eq:onTF$ does not end with $\#3$, implying that the grabber function took everything as its arguments. The first brace group can only be empty if $\#7$ starts with $\#2$, otherwise the brace group preceding $\#7$ would not vanish. The third brace group is empty, thus the $\backslash q_{_} _ cmd$ that was used by our grabber $\#1$ must be the one that we inserted (not some token in $\#7$), hence the second brace group contains the end of $\#7$ followed by $\#2$. Since this is $\#2$ on the right-hand-side, and no brace can be lost there, $\#7$ must contain nothing else than its leading $\#2$.


```

2284         \q_nil { } ##2 \ERROR \q__cmd \ERROR
2285     }
2286     { ##4 {#1} \q__cmd ##5 ##6 {##7} }
2287 }
2288 }
2289 \exp_args:No \__cmd_tmp:w { \c_novalue_tl }

```

At this stage, #7 is `\q_nil {<piece 1>}` *<more for piece 1>*, and we want to concatenate all that, removing `\q_nil`, and keeping the opening delimiter #2. Simply use `\use_ii:nn`. Also, #8 is *<remainder of piece 2>* `\ERROR`, and #9 is `\ERROR` *<more for piece 2>*. We concatenate those, replacing the two `\ERROR` by the closing delimiter #3.

```

2290 \cs_new:Npn \__cmd_expandable_grab_D:NNNwNNnnn #1#2#3#4 \q__cmd #5#6#7#8#9
2291 {
2292     \exp_args:Nof \__cmd_expandable_grab_D:nnNNwNN
2293     { \use_ii:nn #7 #2 }
2294     { \__cmd_expandable_grab_D:Nw #3 \exp_stop_f: #8 #9 }
2295     #1#2#3 #4 \q__cmd #5 #6
2296 }
2297 \cs_new:Npn \__cmd_expandable_grab_D:Nw #1#2 \ERROR \ERROR { #2 #1 }

```

Armed with our two new *<pieces>*, we are ready to loop. However, we must first see if *<piece 2>* (here #2) contains any opening delimiter #4. Again, we expand #3, this time removing its whole output with `\use_none:nnn`. The test is similar to `\tl_if_in:nnTF`. The token list is empty if and only if #2 does not contain the opening delimiter. In that case, we are done, and put the argument (from which we remove a spurious pair of delimiters coming from how we started the loop). Otherwise, we go back to looping with `__cmd_expandable_grab_D:NNNwNNnnn`. The code to deal with brace stripping is much the same as for the non-expandable case.

```

2298 \cs_new:Npn \__cmd_expandable_grab_D:nnNNwNN #1#2#3#4#5#6 \q__cmd #7#8
2299 {
2300     \exp_args:No \tl_if_empty:oTF
2301     { #3 { \use_none:nnn } #2 \q__cmd #5 #4 \q__cmd #5 }
2302     {
2303         \tl_if_blank:oTF { \use_none:nn #1#2 }
2304         { \__cmd_put_arg_expandable:ow { \use_none:nn #1#2 } }
2305         {
2306             \str_if_eq:eeTF
2307             { \exp_not:o { \use_none:nn #1#2 } }
2308             { { \exp_not:o { \use_iii:nnnn #1#2 \q_nil } } }
2309             { \__cmd_put_arg_expandable:ow { \use_iii:nnn #1#2 } }
2310             { \__cmd_put_arg_expandable:ow { \use_none:nn #1#2 } }
2311         }
2312         #6 \q__cmd #7 #8
2313     }
2314     {
2315         #3
2316         { \__cmd_expandable_grab_D:NNNwNNnnn #3#4#5#6 \q__cmd #7 #8 }
2317         \q_nil {#1} #2 \ERROR \q__cmd \ERROR
2318     }
2319 }

```

(End of definition for __cmd_expandable_grab_D:w and others.)

`_cmd_expandable_grab_D_alt:w` When the delimiters are identical, nesting is not possible and a simplified approach is used. The test concept here is the same as for the case where the delimiters are different
`_cmd_expandable_grab_D_alt:NNwNNn` but there cannot be any nesting.
`_cmd_expandable_grab_D_alt:Nwn`

```

2320 \cs_new:Npn \_cmd_expandable_grab_D_alt:w #1 \q__cmd #2#3
2321   { #2 { \_cmd_expandable_grab_D_alt:NNwNNn #1 \q__cmd #2 #3 } }
2322 \cs_set_protected:Npn \_cmd_tmp:w #1
2323   {
2324     \cs_new:Npn \_cmd_expandable_grab_D_alt:NNwNNn ##1##2##3 \q__cmd ##4##5##6
2325     {
2326       \str_if_eq:nnTF {##6} {##2}
2327       {
2328         \str_if_eq:onTF
2329           { ##1 { } ##6 ##2 ##2 }
2330           { { } ##2 }
2331       }
2332       { \use_ii:nn }
2333       {
2334         ##1
2335         { \_cmd_expandable_grab_D_alt:NNwn ##4 ##5 ##3 \q__cmd }
2336         ##6 \ERROR
2337       }
2338       { ##3 {#1} \q__cmd ##4 ##5 {##6} }
2339     }
2340   }
2341 \exp_args:No \_cmd_tmp:w { \c_novalue_tl }
2342 \cs_new:Npn \_cmd_expandable_grab_D_alt:NNwn #1#2#3 \q__cmd #4
2343   {
2344     \tl_if_blank:oTF { \use_none:n #4 }
2345     { \_cmd_put_arg_expandable:ow { \use_none:n #4 } }
2346     {
2347       \str_if_eq:eeTF
2348         { \exp_not:o { \use_none:n #4 } }
2349         { { \exp_not:o { \use_ii:nnn #4 \q__nil } } }
2350         { \_cmd_put_arg_expandable:ow { \use_ii:nn #4 } }
2351         { \_cmd_put_arg_expandable:ow { \use_none:n #4 } }
2352     }
2353     #3 \q__cmd #1 #2
2354   }

```

(End of definition for `_cmd_expandable_grab_D_alt:w`, `_cmd_expandable_grab_D_alt:NNwNNn`, and `_cmd_expandable_grab_D_alt:Nwn`.)

`_cmd_expandable_grab_E:w` We keep track of long/short by placing the appropriate grabber as the third token after `\q__cmd`; it is eventually removed by the `end:nnw` auxiliary. The `aux:w` auxiliary will be called repeatedly with two arguments: the set of pairs $\langle \textit{parser} \rangle \langle \textit{token} \rangle$, and the set of arguments found so far (initially all `\NoValue-`). At each step, grab what follows in the input stream then call the `loop:nnnNNw` auxiliary to compare it with each possible embellishment in turn. This auxiliary's #1 is what was found in the input, #2 collects $\langle \textit{parser} \rangle \langle \textit{token} \rangle$ pairs that did not match, #3 collects the corresponding arguments found previously, #4 and #5 is the current pair, #6 is the remaining pairs, #7 is empty or two `\q__nil`, and #8 is the current argument. If none of the pairs matched (determined by `\quark_if_nil:NTF`) then call the `end` auxiliary to stop looking for embellishments, remembering to put what was grabbed in the input back where it belongs, and storing

the arguments found just before `\q__cmd`. If the current argument #8 is not `-NoValue-` or if the input #1 does not match #5 (see `t`-type arguments below for a similar `\str_if_eq:onTF` test) then carry on the loop. Otherwise, we found a new embellishment: grab the corresponding argument in the input using the `find:w` auxiliary. To avoid losing braces around that auxiliary's argument we include a space, which will be eliminated in the next loop through embellishments.

```

2355 \cs_new:Npn \__cmd_expandable_grab_E:w #1 \q__cmd #2#3
2356   { \__cmd_expandable_grab_E_aux:w #1 \q__cmd #2 #3 #3 }
2357 \cs_new:Npn \__cmd_expandable_grab_E_long:w #1 \q__cmd #2#3
2358   { \__cmd_expandable_grab_E_aux:w #1 \q__cmd #2 #3 #2 }
2359 \cs_new:Npn \__cmd_expandable_grab_E_aux:w #1 \q__cmd #2#3#4
2360   { #2 { \__cmd_expandable_grab_E_test:nw #1 \q__cmd #2 #3 #4 } }
2361 \cs_new:Npn \__cmd_expandable_grab_E_test:nw #1#2#3 \q__cmd #4#5#6#7
2362   {
2363     \__cmd_expandable_grab_E_loop:nnnNw {#7} { } { }
2364     #1 \q_nil \q_nil \q_nil \q_mark #2 \q_nil
2365     #3 \q__cmd #4 #5 #6
2366   }
2367 \cs_new:Npn \__cmd_expandable_grab_E_loop:nnnNw
2368   #1#2#3#4#5#6 \q_nil #7 \q_mark #8
2369   {
2370     \quark_if_nil:NTF #4
2371     { \__cmd_expandable_grab_E_end:nw {#1} {#3} }
2372     {
2373       \tl_if_novalue:NTF {#8}
2374       { \str_if_eq:onTF { #4 { } #1 #5 } {#5} }
2375       { \use_ii:nn }
2376       { \__cmd_expandable_grab_E_find:w { #2 #4 #5 #6 } {#3} ~ }
2377       {
2378         \__cmd_expandable_grab_E_loop:nnnNw
2379         {#1} { #2 #4 #5 } { #3 {#8} }
2380         #6 \q_nil #7 \q_mark
2381       }
2382     }
2383   }
2384 \cs_new:Npn \__cmd_expandable_grab_E_find:w #1 \q__cmd #2#3#4
2385   { #4 { \__cmd_expandable_grab_E_find:nw #1 \q__cmd #2 #3 #4 } }
2386 \cs_new:Npn \__cmd_expandable_grab_E_find:nw #1#2#3 \q_nil #4 \q__cmd #5#6#7#8
2387   { \__cmd_expandable_grab_E_aux:w {#1} { #2 {#8} #3 } #4 \q__cmd #5 #6 #7 }
2388 \cs_new:Npn \__cmd_expandable_grab_E_end:nw #1#2#3 \q__cmd #4#5#6
2389   { #3 #2 \q__cmd #4 #5 {#1} }

```

(End of definition for `__cmd_expandable_grab_E:w` and others.)

```

\__cmd_expandable_grab_m:w
  \__cmd_expandable_grab_m_long:w
  \__cmd_expandable_grab_m_aux:wNn

```

The mandatory case is easy: find the auxiliary after the `\q__cmd`, and use it directly to grab the argument, then correctly position the argument before `\q__cmd`.

```

2390 \cs_new:Npn \__cmd_expandable_grab_m:w #1 \q__cmd #2#3
2391   { #3 { \__cmd_expandable_grab_m_aux:wNn #1 \q__cmd #2 #3 } }
2392 \cs_new:Npn \__cmd_expandable_grab_m_long:w #1 \q__cmd #2#3
2393   { #2 { \__cmd_expandable_grab_m_aux:wNn #1 \q__cmd #2 #3 } }
2394 \cs_new:Npn \__cmd_expandable_grab_m_aux:wNn #1 \q__cmd #2#3#4
2395   { #1 {#4} \q__cmd #2 #3 }

```

(End of definition for `__cmd_expandable_grab_m:w`, `__cmd_expandable_grab_m_long:w`, and `__cmd_expandable_grab_m_aux:wNn`.)

`_cmd_expandable_grab_R:w` Much the same as for the D-type argument, with only the lead-off function varying.

```
\_cmd_expandable_grab_R_aux:NNwNNn
2396 \cs_new:Npn \_cmd_expandable_grab_R:w #1 \q__cmd #2#3
2397   { #2 { \_cmd_expandable_grab_R_aux:NNwNNn #1 \q__cmd #2#3 } }
2398 \cs_set_protected:Npn \_cmd_tmp:w #1
2399   {
2400     \cs_new:Npn \_cmd_expandable_grab_R_aux:NNwNNn ##1##2##3##4 \q__cmd ##5##6##7
2401     {
2402       \str_if_eq:nnTF {##7} {##2}
2403       {
2404         \str_if_eq:onTF
2405           { ##1 { } { } ##7 ##2 \q__cmd ##3 }
2406           { { } {##2} { } }
2407       }
2408       { \use_ii:nn }
2409       {
2410         ##1
2411         { \_cmd_expandable_grab_D:NNwNNnn ##1##2##3##4 \q__cmd ##5##6 }
2412         \q_nil { } ##2 \ERROR \q__cmd \ERROR
2413       }
2414       {
2415         \msg_expandable_error:nfff { cmd } { missing-required }
2416         { \exp_args:Nf \tl_trim_spaces:n { \token_to_str:N ##5 } }
2417         { \tl_to_str:n {##2} }
2418         ##4 {#1} \q__cmd ##5 ##6 {##7}
2419       }
2420     }
2421   }
2422 \exp_args:No \_cmd_tmp:w { \c_novalue_tl }
```

(End of definition for `_cmd_expandable_grab_R:w` and `_cmd_expandable_grab_R_aux:NNwNNn`.)

`_cmd_expandable_grab_R_alt:w` When the delimiters are identical, nesting is not possible and a simplified approach is used. The test concept here is the same as for the case where the delimiters are different.

```
\_cmd_expandable_grab_R_alt_aux:NNwNNn
2423 \cs_new:Npn \_cmd_expandable_grab_R_alt:w #1 \q__cmd #2#3
2424   { #2 { \_cmd_expandable_grab_R_alt_aux:NNwNNn #1 \q__cmd #2#3 } }
2425 \cs_set_protected:Npn \_cmd_tmp:w #1
2426   {
2427     \cs_new:Npn \_cmd_expandable_grab_R_alt_aux:NNwNNn ##1##2##3 \q__cmd ##4##5##6
2428     {
2429       \str_if_eq:nnTF {##6} {##2}
2430       {
2431         \str_if_eq:onTF
2432           { ##1 { } ##6 ##2 ##2 }
2433           { { } ##2 }
2434       }
2435       { \use_ii:nn }
2436       {
2437         ##1
2438         { \_cmd_expandable_grab_D_alt:NNwn ##4 ##5 ##3 \q__cmd }
2439         ##6 \ERROR
2440       }
2441       {
2442         \msg_expandable_error:nfff { cmd } { missing-required }
2443         { \exp_args:Nf \tl_trim_spaces:n { \token_to_str:N ##4 } }
2444       }
2445     }
2446   }
```

```

2444         { \tl_to_str:n {##2} }
2445         ##3 {#1} \q__cmd ##4 ##5 {##6}
2446     }
2447 }
2448 }
2449 \exp_args:No \__cmd_tmp:w { \c_novalue_tl }

```

*(End of definition for __cmd_expandable_grab_R_alt:w and
__cmd_expandable_grab_R_alt_aux:NNwNNn.)*

__cmd_expandable_grab_t:w As for a D-type argument, here we compare the grabbed tokens using the only parser we
__cmd_expandable_grab_t_aux:NNwn have in order to work out if #2 is exactly equal to the output of the grabber.

```

2450 \cs_new:Npn \__cmd_expandable_grab_t:w #1 \q__cmd #2#3
2451 { #2 { \__cmd_expandable_grab_t_aux:NNwn #1 \q__cmd #2 #3 } }
2452 \cs_new:Npn \__cmd_expandable_grab_t_aux:NNwn #1#2#3 \q__cmd #4#5#6
2453 {
2454   \str_if_eq:onTF { #1 { } #6 #2 } {#2}
2455   { #3 { \BooleanTrue } \q__cmd #4 #5 }
2456   { #3 { \BooleanFalse } \q__cmd #4 #5 {#6} }
2457 }

```

(End of definition for __cmd_expandable_grab_t:w and __cmd_expandable_grab_t_aux:NNwn.)

__cmd_put_arg_expandable:nw A useful helper, to store arguments when they are ready.

```

\__cmd_put_arg_expandable:ow 2458 \cs_new:Npn \__cmd_put_arg_expandable:nw #1#2 \q__cmd { #2 {#1} \q__cmd }
2459 \cs_generate_variant:Nn \__cmd_put_arg_expandable:nw { o }

```

(End of definition for __cmd_put_arg_expandable:nw.)

1.10 Argument processors

__cmd_bool_reverse:N A simple reversal.

```

2460 \cs_new_protected:Npn \__cmd_bool_reverse:N #1
2461 {
2462   \bool_if:NTF #1
2463   { \tl_set:Nn \ProcessedArgument { \c_false_bool } }
2464   { \tl_set:Nn \ProcessedArgument { \c_true_bool } }
2465 }

```

(End of definition for __cmd_bool_reverse:N.)

```

\l__cmd_split_list_seq
\l__cmd_split_list_tl
\__cmd_split_list_multi:nn
\__cmd_split_list_multi:nV
\__cmd_split_list_single:Nn
2466 \seq_new:N \l__cmd_split_list_seq
2467 \tl_new:N \l__cmd_split_list_tl
2468 \cs_new_protected:Npn \__cmd_split_list:nn #1#2
2469 {
2470   \tl_if_single:nTF {#1}
2471   {
2472     \token_if_cs:NTF #1
2473     { \__cmd_split_list_multi:nn {#1} {#2} }
2474     { \__cmd_split_list_single:Nn #1 {#2} }
2475   }
2476   { \__cmd_split_list_multi:nn {#1} {#2} }
2477 }
2478 \cs_new_protected:Npn \__cmd_split_list_multi:nn #1#2
2479 {
2480   \seq_set_split:Nnn \l__cmd_split_list_seq {#1} {#2}
2481   \tl_clear:N \ProcessedArgument
2482   \seq_map_inline:Nn \l__cmd_split_list_seq
2483   { \tl_put_right:Nn \ProcessedArgument { ##1} } }
2484 }
2485 \cs_generate_variant:Nn \__cmd_split_list_multi:nn { nV }
2486 \group_begin:
2487 \char_set_catcode_active:N ^^@
2488 \cs_new_protected:Npn \__cmd_split_list_single:Nn #1#2
2489 {
2490   \tl_set:Nn \l__cmd_split_list_tl {#2}
2491   \group_begin:
2492   \char_set_lccode:nn { ^^@ } { '#1 }
2493   \tex_lowercase:D
2494   {
2495     \group_end:
2496     \tl_replace_all:Nnn \l__cmd_split_list_tl { ^^@ }
2497     } {#1}
2498   \__cmd_split_list_multi:nV {#1} \l__cmd_split_list_tl
2499 }
2500 \group_end:

```

(End of definition for `__cmd_split_list:nn`, `__cmd_split_list_multi:nn`, and `__cmd_split_list_single:Nn`.)

```

\__cmd_split_argument:nnn
\__cmd_split_argument_aux:nnnn
\__cmd_split_argument_aux:n
\__cmd_split_argument_aux:wn

```

Splitting to a known number of items is a special version of splitting a list, in which the limit is hard-coded and where there will always be exactly the correct number of output items. An auxiliary function is used to save on working out the token list length several times.

```

2501 \cs_new_protected:Npn \__cmd_split_argument:nnn #1#2#3
2502 {
2503   \__cmd_split_list:nn {#2} {#3}
2504   \exp_args:Nf \__cmd_split_argument_aux:nnnn
2505   { \tl_count:N \ProcessedArgument }
2506   {#1} {#2} {#3}
2507 }

```

```

2508 \cs_new_protected:Npn \__cmd_split_argument_aux:nnnn #1#2#3#4
2509 {
2510   \int_compare:nNnF {#1} = { #2 + 1 }
2511   {
2512     \int_compare:nNnTF {#1} > { #2 + 1 }
2513     {
2514       \tl_set:Nx \ProcessedArgument
2515       {
2516         \exp_last_unbraced:NnNo
2517         \__cmd_split_argument_aux:n
2518         { #2 + 1 }
2519         \use_none_delimit_by_q_stop:w
2520         \ProcessedArgument
2521         \q_stop
2522       }
2523       \msg_error:nnxxx { cmd } { arg-split }
2524       { \tl_to_str:n {#3} } { \int_eval:n { #2 + 1 } }
2525       { \tl_to_str:n {#4} }
2526     }
2527   {
2528     \tl_put_right:Nx \ProcessedArgument
2529     {
2530       \prg_replicate:nn { #2 + 1 - (#1) }
2531       { { \exp_not:V \c_novalue_tl } }
2532     }
2533   }
2534 }
2535 }

```

Auxiliaries to leave exactly the correct number of arguments in `\ProcessedArgument`.

```

2536 \cs_new:Npn \__cmd_split_argument_aux:n #1
2537 { \prg_replicate:nn {#1} { \__cmd_split_argument_aux:wn } }
2538 \cs_new:Npn \__cmd_split_argument_aux:wn #1 \use_none_delimit_by_q_stop:w #2
2539 {
2540   \exp_not:n { {#2} }
2541   #1
2542   \use_none_delimit_by_q_stop:w
2543 }

```

(End of definition for `__cmd_split_argument:nnn` and others.)

`__cmd_trim_spaces:n` This one is almost trivial.

```

2544 \cs_new_protected:Npn \__cmd_trim_spaces:n #1
2545 { \tl_set:Nx \ProcessedArgument { \tl_trim_spaces:n {#1} } }

```

(End of definition for `__cmd_trim_spaces:n`.)

1.11 Conversion to key–value form

This is implemented as a process but with no public interfaces, hence is treated separately from the others: it's a feature of `ltxcmd` which just happens to use the same mechanism as a processor.

```

\__cmd_arg_to_keyvalue:nn If the entire argument is braced, we treat as free text and return as the value for the text
\__cmd_arg_to_keyvalue_braces:nnn key. Alternatively, if the start of the input is =, then it is forced to be key–value. To
\__cmd_arg_to_keyvalue_auxi:Nnn avoid needing to worry about catcodes for this, and to allow spaces around the =, we use
\__cmd_arg_to_keyvalue_auxii:Nnnn a series of steps rather than a delimited argument.
\__cmd_arg_to_keyvalue_auxiii:Nnnn 2546 \cs_new_protected:Npn \__cmd_arg_to_keyvalue:nn #1#2
\__cmd_arg_to_keyvalue_auxiv:Nnnn 2547 {
\__cmd_arg_to_keyvalue_auxv:nn 2548 \tl_trim_spaces_apply:nN {#2} \__cmd_arg_to_keyvalue_braces:nnn
\__cmd_arg_to_keyvalue_loop:w 2549 {#1} {#2}
\__cmd_arg_to_keyvalue_loop_group:n 2550 }
\__cmd_arg_to_keyvalue_loop_space:w 2551 \cs_new_protected:Npn \__cmd_arg_to_keyvalue_braces:nnn #1#2#3
\__cmd_arg_to_keyvalue_loop_N_type:N 2552 {
\__cmd_arg_to_keyvalue_math:w 2553 \tl_if_head_is_group:nT {#1}
\__cmd_arg_to_keyvalue_math_N_type:N 2554 {
\__cmd_arg_to_keyvalue_math_group:n 2555 \tl_if_blank:oT { \use_none:n #1 }
\__cmd_arg_to_keyvalue_math_space:w 2556 {
\__cmd_arg_to_keyvalue_set_default:nn 2557 \tl_set:Nx \ProcessedArgument { #2 = { \exp_not:n #1 } }
\__cmd_arg_to_keyvalue_set_keyvalue:nn 2558 \use_none:nnnn
\__cmd_arg_to_keyvalue_set_keyvalue:nn 2559 }
\__cmd_split_N_head_apply:Nn 2560 }
\__cmd_split_N_head_apply_aux:NNw 2561 \__cmd_arg_to_keyvalue_auxi:nnn {#1} {#2} {#3}
2562 }
2563 \cs_new:Npn \__cmd_arg_to_keyvalue_auxi:nnn #1
2564 {
2565 \tl_if_head_is_N_type:nTF {#1}
2566 { \__cmd_split_N_head_apply:Nn \__cmd_arg_to_keyvalue_auxii:Nnnn {#1} }
2567 { \__cmd_arg_to_keyvalue_auxv:nn }
2568 }
2569 \cs_new:Npn \__cmd_arg_to_keyvalue_auxii:Nnnn #1#2
2570 {
2571 \str_if_eq:eeTF { \exp_not:n {#1} } { = }
2572 { \tl_trim_spaces_apply:nN {#2} \__cmd_arg_to_keyvalue_auxiii:nnn }
2573 { \__cmd_arg_to_keyvalue_auxv:nn }
2574 }
2575 \cs_new:Npn \__cmd_arg_to_keyvalue_auxiii:nnn #1
2576 {
2577 \tl_if_head_is_N_type:nTF {#1}
2578 { \__cmd_split_N_head_apply:Nn \__cmd_arg_to_keyvalue_auxiv:Nnnn {#1} }
2579 { \__cmd_arg_to_keyvalue_auxv:nn }
2580 }
2581 \cs_new:Npn \__cmd_arg_to_keyvalue_auxiv:Nnnn #1#2
2582 {
2583 \str_if_eq:eeTF { \exp_not:n {#1} } { , }
2584 { \tl_set:Nn \ProcessedArgument {#2} \use_none:nn }
2585 { \__cmd_arg_to_keyvalue_auxv:nn }
2586 }

```

The two clear-cut cases have been eliminated, and we therefore have to deal with a search for = signs. We need an “action” loop here so we do not get misled by for example {=}. As the code here is for very much predictable types of input, we hard-code what constitutes math mode opening and closing. At the very beginning, the default key (#1) and the argument as given by the user (#2) are placed right after the `__cmd_recursion_stop`, so that when the recursion ends, the macros `__cmd_arg_to_keyvalue_set_default:nn` or `__cmd_arg_to_keyvalue_set_keyvalue:nn` can be used to grab these two items and

set the `\ProcessedArgument` accordingly.

```
2587 \cs_new_protected:Npn \__cmd_arg_to_keyvalue_auxv:nn #1#2
2588 {
2589   \__cmd_arg_to_keyvalue_loop:w #2
2590   \q__cmd_recursion_tail \q__cmd_recursion_stop {#1} {#2}
2591 }
2592 \cs_new_protected:Npn \__cmd_arg_to_keyvalue_loop:w #1 \q__cmd_recursion_stop
2593 {
2594   \tl_if_head_is_N_type:nTF {#1}
2595   { \__cmd_arg_to_keyvalue_loop_N_type:N }
2596   {
2597     \tl_if_head_is_group:nTF {#1}
2598     { \__cmd_arg_to_keyvalue_loop_group:n }
2599     { \__cmd_arg_to_keyvalue_loop_space:w }
2600   }
2601   #1 \q__cmd_recursion_stop
2602 }
2603 \cs_new_protected:Npn \__cmd_arg_to_keyvalue_loop_group:n #1
2604 { \__cmd_arg_to_keyvalue_loop:w }
2605 \use:n { \cs_new_protected:Npn \__cmd_arg_to_keyvalue_loop_space:w } ~
2606 { \__cmd_arg_to_keyvalue_loop:w }
2607 \cs_new_protected:Npn \__cmd_arg_to_keyvalue_loop_N_type:N #1
2608 {
2609   \__cmd_if_recursion_tail_stop_do:Nn #1
2610   { \__cmd_arg_to_keyvalue_set_default:nn }
2611   \str_if_eq:nnTF {#1} { = }
2612   {
2613     \__cmd_use_i_delimit_by_q_recursion_stop:nw
2614     { \__cmd_arg_to_keyvalue_set_keyvalue:nn }
2615   }
2616   {
2617     \bool_lazy_or:nnTF
2618     { \token_if_math_toggle_p:N #1 }
2619     { \str_if_eq_p:nn {#1} { \ ( } }
2620     { \__cmd_arg_to_keyvalue_math:w }
2621     { \__cmd_arg_to_keyvalue_loop:w }
2622   }
2623 }
2624 \cs_new_protected:Npn \__cmd_arg_to_keyvalue_math:w #1 \q__cmd_recursion_stop
2625 {
2626   \tl_if_head_is_N_type:nTF {#1}
2627   { \__cmd_arg_to_keyvalue_math_N_type:N }
2628   {
2629     \tl_if_head_is_group:nTF {#1}
2630     { \__cmd_arg_to_keyvalue_math_group:n }
2631     { \__cmd_arg_to_keyvalue_math_space:w }
2632   }
2633   #1 \q__cmd_recursion_stop
2634 }
2635 \cs_new_protected:Npn \__cmd_arg_to_keyvalue_math_N_type:N #1
2636 {
2637   \__cmd_if_recursion_tail_stop_do:Nn #1
2638   { \__cmd_arg_to_keyvalue_set_default:nn }
2639   \bool_lazy_or:nnTF
```

```

2640     { \token_if_math_toggle_p:N #1 }
2641     { \str_if_eq_p:nn {#1} { \ } } }
2642     { \__cmd_arg_to_keyvalue_loop:w }
2643     { \__cmd_arg_to_keyvalue_math:w }
2644 }
2645 \cs_new_protected:Npn \__cmd_arg_to_keyvalue_math_group:n #1
2646 { \__cmd_arg_to_keyvalue_math:w }
2647 \use:n { \cs_new_protected:Npn \__cmd_arg_to_keyvalue_math_space:w } ~
2648 { \__cmd_arg_to_keyvalue_math:w }
2649 \cs_new_protected:Npn \__cmd_arg_to_keyvalue_set_default:nn #1#2
2650 { \tl_set:Nn \ProcessedArgument { #1 = {#2} } }
2651 \cs_new_protected:Npn \__cmd_arg_to_keyvalue_set_keyvalue:nn #1#2
2652 { \tl_set:Nn \ProcessedArgument {#2} }

```

A utility to allow us to grab the first N-type token without risking brace stripping the rest of the input.

```

2653 \cs_new:Npn \__cmd_split_N_head_apply:Nn #1#2
2654 { \exp:w \if_false: { \fi: \__cmd_split_N_head_apply_aux:NNw #1#2 } }
2655 \cs_new:Npn \__cmd_split_N_head_apply_aux:NNw #1#2
2656 {
2657     \exp_after:wN \exp_end:
2658     \exp_after:wN #1 \exp_after:wN #2 \exp_after:wN { \if_false: } \fi:
2659 }
2660

```

(End of definition for __cmd_arg_to_keyvalue:nn and others.)

1.12 Utilities

```

\__cmd_check_definable:nNT
  \__cmd_check_definable_aux:nN

```

Check that a token list is appropriate as a first argument of `\NewDocumentCommand` and similar functions and otherwise produce an error. First trim whitespace to allow for spaces around the actual command to be defined. If the result has multiple tokens, it is not a valid argument. The single token is a control sequence exactly if its string representation has more than one character (using `\token_to_str:N` rather than `\tl_to_str:n` to avoid problems with macro parameter characters, and setting `\tex_escapechar:D` to prevent it from being non-printable). Finally, check for an active character: this is done by lowercasing the token to fix its character code (arbitrarily to that of `?`) and comparing the result to an active `?`. Both control sequences and active characters are valid arguments, and non-active character tokens are not. In all cases, the group opened to keep assignments local must be closed.

```

2661 \cs_new_protected:Npn \__cmd_check_definable:nNT #1
2662 { \tl_trim_spaces_apply:nN {#1} \__cmd_check_definable_aux:nN }
2663 \group_begin:
2664   \char_set_catcode_active:n { '?' }
2665   \cs_new_protected:Npn \__cmd_check_definable_aux:nN #1#2
2666   {
2667     \group_begin:
2668     \tl_if_single_token:nTF {#1}
2669     {
2670       \int_set:Nn \tex_escapechar:D { 92 }
2671       \exp_args:Nx \tl_if_empty:nTF
2672       { \exp_args:No \str_tail:n { \token_to_str:N #1 } }
2673       {

```

```

2674         \exp_args:Nx \char_set_lccode:nn
2675         { ' \str_head:n {#1} } { '?' }
2676         \tex_lowercase:D { \tl_if_eq:nnTF {#1} } { ? }
2677         { \group_end: \use_iii:nnn }
2678         { \group_end: \use_i:nnn }
2679     }
2680     { \group_end: \use_iii:nnn }
2681 }
2682 { \group_end: \use_ii:nnn }
2683 {
2684     \msg_error:nxxx { cmd } { not-definable }
2685     { \tl_to_str:n {#1} } { \token_to_str:N #2 }
2686 }
2687 {
2688     \msg_error:nxxx { cmd } { not-one-token }
2689     { \tl_to_str:n {#1} } { \token_to_str:N #2 }
2690 }
2691 }
2692 \group_end:

```

(End of definition for `__cmd_check_definable:nNT` and `__cmd_check_definable_aux:nN`.)

`__cmd_token_if_cs:NTF` Based on the definition of `__cmd_check_definable_aux:nN` above, but only checks for an actual control sequence (*i.e.*, `\langle anything \rangle`). `\tex_escapechar:D` is temporarily changed to a known value and then it checks if `\string#1` contains more than one character: if it does, it's a control sequence. This test differs from `\token_if_cs:NTF` for example in `\token_if_cs:NTF \c_group_begin_token {T}{F}`, where `\token_if_cs:NTF` returns false.

```

2693 \cs_new_protected:Npn \__cmd_token_if_cs:NTF #1
2694 {
2695     \group_begin:
2696     \int_set:Nn \tex_escapechar:D { 92 }
2697     \exp_args:Nx \tl_if_empty:NTF
2698     { \exp_args:No \str_tail:n { \token_to_str:N #1 } }
2699     { \group_end: \use_ii:nn }
2700     { \group_end: \use_i:nn }
2701 }

```

(End of definition for `__cmd_token_if_cs:NTF`.)

`__cmd_tl_mapthread_function:NNN` Analogue of `\seq_mapthread_function:NNN` for token lists.

```

\__cmd_tl_mapthread_function:nnN 2702 \cs_new:Npn \__cmd_tl_mapthread_function:NNN #1#2#3
\__cmd_tl_mapthread_loop:w 2703 {
2704     \exp_after:wN \exp_after:wN
2705     \exp_after:wN \__cmd_tl_mapthread_loop:w
2706     \exp_after:wN \exp_after:wN
2707     \exp_after:wN #3
2708     \exp_after:wN #1
2709     \exp_after:wN \q_recursion_tail
2710     \exp_after:wN \q_mark
2711     #2
2712     \q_recursion_tail
2713     \q_recursion_stop
2714 }

```

```

2715 \cs_new:Npn \__cmd_tl_mapthread_function:nnN #1#2#3
2716   {
2717     \__cmd_tl_mapthread_loop:w #3
2718     #1 \q_recursion_tail \q_mark
2719     #2 \q_recursion_tail \q_recursion_stop
2720   }
2721 \cs_new:Npn \__cmd_tl_mapthread_loop:w #1#2#3 \q_mark #4
2722   {
2723     \quark_if_recursion_tail_stop:n {#2}
2724     \quark_if_recursion_tail_stop:n {#4}
2725     #1 {#2} {#4}
2726     \__cmd_tl_mapthread_loop:w #1#3 \q_mark
2727   }

```

(End of definition for __cmd_tl_mapthread_function:NNN, __cmd_tl_mapthread_function:nnN, and __cmd_tl_mapthread_loop:w.)

```

\__kernel_cmd_if_xparse:NTF
  \__cmd_cmd_type_cases:NnnnnnF
\__cmd_cmd_if_xparse_aux:N

```

To determine whether the command is an xparse command check that its `arg_spec` is empty (this also excludes non-macros) and that its `replacement_spec` starts with either `__cmd_start:nNNnnn` (non-expandable command) or `__cmd_start_expandable:nNNNNn` (expandable command) or `__cmd_start_optimized:` (optimized command) or `__cmd_start_env:nnnnn` (environment) or `\environment #1 end aux` (environment end).

This conditional is needed in several kernel modules and is therefore has a kernel-internal name.

```

2728 \cs_new_protected:Npn \__cmd_cmd_type_cases:NnnnnnF #1 #2 #3 #4 #5 #6 #7
2729   {
2730     \exp_args:Ne \str_case_e:nnF
2731     {
2732       \exp_args:Nf \tl_if_empty:nT { \__kernel_cs_parameter_spec:N #1 }
2733       { \exp_not:N \exp_not:n { \exp_not:e { \tl_head:N #1 } } }
2734     }
2735     {
2736       { \exp_not:N \__cmd_start:nNNnnn } {#2}
2737       { \exp_not:N \__cmd_start_expandable:nNNNNn } {#3}
2738       { \exp_not:N \__cmd_start_optimized: } {#4}
2739       { \exp_not:N \__cmd_start_env:nnnnn } {#5}
2740     }
2741     \exp_after:wN \exp_not:N
2742     \cs:w environment~
2743     \exp_last_unbraced:Ne \use_none:nnn
2744     { \cs_to_str:N #1 } ~end~aux \cs_end:
2745     } {#6}
2746   }
2747   {#7}
2748 }
2749 \cs_new_protected:Npn \__kernel_cmd_if_xparse:NTF #1
2750   {
2751     \__cmd_cmd_type_cases:NnnnnnF #1
2752     { } { } { } { } { } { } { } { \use_iii:nnn }
2753     \use_i:nn
2754   }

```

(End of definition for __kernel_cmd_if_xparse:NTF, __cmd_cmd_type_cases:NnnnnnF, and __cmd_cmd_if_xparse_aux:N.)

`__cmd_peek_nospace:NTF` Collect spaces in a loop, and put the collected spaces back in the false branch of a call to `\peek_meaning:NTF` or `\peek_meaning_remove:NTF`.

`__cmd_peek_nospace_remove:NTF`
`__cmd_peek_nospace_aux:nNNTF`

```

2755 \cs_new_protected:Npn \__cmd_peek_nospace:NTF
2756   { \__cmd_peek_nospace_aux:nNNTF { } \__cmd_peek_meaning:NTF }
2757 \cs_new_protected:Npn \__cmd_peek_nospace_remove:NTF
2758   { \__cmd_peek_nospace_aux:nNNTF { } \__cmd_peek_meaning_remove:NTF }
2759 \cs_new_protected:Npn \__cmd_peek_nospace_aux:nNNTF #1#2#3#4#5
2760   {
2761     \peek_meaning_remove:NTF \c_space_token
2762     { \__cmd_peek_nospace_aux:nNNTF { #1 ~ } #2 #3 {#4} {#5} }
2763     { #2 #3 { #4 } { #5 #1 } }
2764   }

```

(End of definition for `__cmd_peek_nospace:NTF`, `__cmd_peek_nospace_remove:NTF`, and `__cmd_peek_nospace_aux:nNNTF`.)

`__cmd_peek_meaning:NTF` Peek ahead for a token with a given meaning. In case the search token is a control sequence, also check that the `<csname>` is the same as the control sequence peeked at. This extra verification is necessary when the command is delimited by control sequence tokens (as opposed to character tokens), and we want the exact same control sequence to match.

`__cmd_peek_meaning_remove:NTF`
`__cmd_peek_cs_check_equal:NNN`
`__cmd_peek_meaning_aux:NNTF`
`__cmd_peek_true_remove:NNw`

```

2765 \cs_new_protected:Npn \__cmd_peek_meaning:NTF
2766   { \__cmd_peek_meaning_aux:NNTF \c_false_bool }
2767 \cs_new_protected:Npn \__cmd_peek_meaning_remove:NTF
2768   { \__cmd_peek_meaning_aux:NNTF \c_true_bool }
2769 \cs_new_protected:Npn \__cmd_peek_meaning_aux:NNTF #1#2#3#4
2770   {
2771     \tl_set:Nn \l__cmd_tmpa_tl {#3}
2772     \tl_set:Nn \l__cmd_tmpb_tl {#4}
2773     \peek_meaning:NTF #2
2774     {
2775       \token_if_eq_meaning:NNTF #2 \c_group_begin_token
2776       { \__cmd_peek_true_remove:Nw #1 }
2777       {
2778         \__cmd_token_if_cs:NTF #2
2779         { \__cmd_peek_cs_check_equal:NNN #1 #2 }
2780         { \__cmd_peek_true_remove:Nw #1 }
2781       }
2782     }
2783     { \l__cmd_tmpb_tl }
2784   }
2785 \cs_new_protected:Npn \__cmd_peek_cs_check_equal:NNN #1#2#3
2786   {
2787     \str_if_eq:nnTF {#2} {#3}
2788     { \__cmd_peek_true_remove:Nw #1 }
2789     { \l__cmd_tmpb_tl }
2790     #3
2791   }
2792 \cs_new_protected:Npn \__cmd_peek_true_remove:Nw #1
2793   {
2794     \bool_if:NTF #1
2795     {
2796       \tex_afterassignment:D \l__cmd_tmpa_tl
2797       \cs_set_eq:NN \__cmd_tmp:w

```

```

2798     }
2799     { \l__cmd_tmpa_tl }
2800 }

```

(End of definition for `__cmd_peek_meaning:NTF` and others.)

1.13 Messages

`__cmd_ignore_def_tl`

```

2801 \tl_const:Nn \c__cmd_ignore_def_tl
2802 { \ \ \ LaTeX-will-ignore~this~entire-definition. }

```

`__cmd_environment_or_command:`

Two texts used in several messages.

```

2803 \cs_new:Npn \__cmd_environment_or_command:
2804 {
2805   \bool_if:NTF \l__cmd_environment_bool
2806     { environment ~ ' \l__cmd_environment_str ' }
2807     {
2808       command ~
2809         ' \c_backslash_str \tl_to_str:N \l__cmd_function_tl '
2810     }
2811 }

```

(End of definition for `__cmd_environment_or_command:`)

Some messages intended as errors when defining commands/environments.

```

2812 \msg_new:nnnn { cmd } { arg-after-body }
2813 { Argument~type~'#1'~must~be~last~in~#2. }
2814 {
2815   The~'#1'~argument~type~must~come~last~but~it~is~followed~
2816   by~'#3'~in~the~argument~specification.~This~is~not~allowed.
2817   \c__cmd_ignore_def_tl
2818 }
2819 \msg_new:nnnn { cmd } { bad-arg-spec }
2820 { Bad~argument~specification~'#2'~for~#1. }
2821 {
2822   The~argument~specification~provided~is~not~valid:~
2823   one~or~more~mandatory~parts~are~missing.
2824   \c__cmd_ignore_def_tl
2825 }
2826 \msg_new:nnnn { cmd } { already-defined }
2827 { Command~'#1'~already~defined. }
2828 {
2829   You~have~used~#2~
2830   with~a~command~that~already~has~a~definition. \ \
2831   The~existing~definition~of~'#1'~will~not~be~altered.
2832 }
2833 \msg_new:nnnn { cmd } { undefined }
2834 { Command ~'#1'~undefined. }
2835 {
2836   You~have~used~#2~
2837   with~a~command~that~was~never~defined.
2838   \c__cmd_ignore_def_tl
2839 }

```

```

2840 \msg_new:nnnn { cmd } { chars-dropped-first-line }
2841   { Characters~'#1'~dropped-on-first-line-of~#2-environment. }
2842   {
2843     LaTeX~was~collecting~a~verbatim-like~environment,~and~the~characters~
2844     '#1'~were~found~after~'\begin{#2}~on~the~first~line:~this~is~not~supported.
2845   }
2846 \msg_new:nnnn { cmd } { chars-dropped-last-line }
2847   { Characters~'#1'~dropped-after-end-of~#2-environment. }
2848   {
2849     LaTeX~was~collecting~a~verbatim-like~environment,~and~the~characters~
2850     '#1'~were~found~after~'\end{#2}~on~the~last~line:~this~is~not~supported.
2851   }
2852 \msg_new:nnnn { cmd } { env-already-defined }
2853   { Environment~'#1'~already~defined. }
2854   {
2855     You~have~used~\NewDocumentEnvironment
2856     with~an~environment~that~already~has~a~definition. \ \ \
2857     The~existing~definition~of~'#1'~will~not~be~altered.
2858   }
2859 \msg_new:nnnn { cmd } { env-end-already-defined }
2860   { End-of-environment~'#1'~already~defined. }
2861   {
2862     You~have~used~\NewDocumentEnvironment
2863     with~an~environment~that~already~has~a~definition~for~'end#1'. \ \ \
2864     The~existing~definition~of~'#1'~will~not~be~altered.
2865   }
2866 \msg_new:nnnn { cmd } { env-undefined }
2867   { Environment~'#1'~undefined. }
2868   {
2869     You~have~used~\RenewDocumentEnvironment
2870     with~an~environment~that~was~never~defined.
2871     \c__cmd_ignore_def_tl
2872   }
2873 \msg_new:nnnn { cmd } { expandable-ending-optional }
2874   { Bad~argument~specification~'#2'~for~#1. }
2875   {
2876     Expandable~commands~must~have~a~final~mandatory~argument~
2877     (or~no~arguments~at~all).~You~cannot~have~a~terminal~optional~
2878     argument~with~expandable~commands.
2879   }
2880 \msg_new:nnnn { cmd } { long-short-mix }
2881   { Invalid~argument~prefix~'+~'~in~command~'#1'. }
2882   {
2883     The~arguments~for~an~expandable~command~must~not~involve~short~
2884     arguments~after~long~arguments.~You~have~tried~to~mix~the~two~types~
2885     when~defining~'#1'.
2886   }
2887 \msg_new:nnnn { cmd } { invalid-command-arg }
2888   { Invalid~argument~type~'#2'~in~#1. }
2889   {
2890     The~letter~'#2'~can~only~be~used~in~environment~argument~
2891     specifications,~but~not~for~commands.
2892     \ \ \
2893     LaTeX~will~ignore~the~entire~definition.

```

```

2894 }
2895 \msg_new:nnnn { cmd } { invalid-expandable-arg }
2896 { Invalid-argument-type-#2'-in-#1. }
2897 {
2898   The-letter-#2'-specifies-an-argument-type-which-cannot-be-used-
2899   in-an-expandable-command.
2900   \c__cmd_ignore_def_tl
2901 }
2902 \msg_new:nnnn { cmd } { invalid-after-optional-expandably }
2903 { Argument-#2'-invalid-after-optional-arg-in-#1. }
2904 {
2905   The-letter-#2'-specifies-an-argument-type-which-cannot-be-used-
2906   in-an-expandable-command-after-an-optional-argument.
2907   \c__cmd_ignore_def_tl
2908 }
2909 \msg_new:nnnn { cmd } { invalid-bang }
2910 { Invalid-argument-prefix-'!'-in-#1. }
2911 {
2912   The-prefix-'!'-is-only-allowed-for-trailing-optional-arguments.-
2913   You-tried-to-apply-it-to-#2.
2914   \c__cmd_ignore_def_tl
2915 }
2916 \msg_new:nnnn { cmd } { not-definable }
2917 { First-argument-of-#2'-must-be-a-command. }
2918 {
2919   The-first-argument-of-#2'-should-be-the-document-command-that-will-
2920   be-defined.-The-provided-argument-#1'-is-a-character.-Perhaps-a-
2921   backslash-is-missing?
2922   \c__cmd_ignore_def_tl
2923 }
2924 \msg_new:nnnn { cmd } { not-one-token }
2925 { First-argument-of-#2'-must-be-a-command. }
2926 {
2927   The-first-argument-of-#2'-should-be-the-document-command-that-will-
2928   be-defined.-The-provided-argument-#1'-contains-more-than-one-
2929   token.-Perhaps-a-backslash-is-missing?
2930   \c__cmd_ignore_def_tl
2931 }
2932 \msg_new:nnnn { cmd } { not-single-token }
2933 { Argument-delimiter-#2'-invalid-in-#1. }
2934 {
2935   The-argument-specification-contains-
2936   \tl_if_empty:nTF{#2}{nothing}{#2}'-
2937   in-a-place-
2938   where-a-single-token-is-required.
2939   \c__cmd_ignore_def_tl
2940 }
2941 \msg_new:nnnn { cmd } { forbidden-group-token }
2942 { Argument-delimiter-#2'-invalid-in-#1. }
2943 {
2944   The-argument-specification-contains-the-implicit-
2945   #3-group-token-#2'-which-is-not-allowed-as-an-argument-delimiter.
2946   \c__cmd_ignore_def_tl
2947 }

```



```

2948 \msg_new:nnnn { cmd } { processor-in-expandable }
2949 { Invalid-argument-prefix-'}>'~in-command~'#1'. }
2950 {
2951   The~argument~specification~for~'#1'~contains~the~processor~function~'>{#2}''.~
2952   This~is~only~supported~for~robust~commands,~but~not~for~expandable~ones.
2953   \c__cmd_ignore_def_tl
2954 }
2955 \msg_new:nnnn { cmd } { keyval-in-expandable }
2956 { Invalid-argument-prefix~'='~in-command~'#1'. }
2957 {
2958   The~argument~specification~for~'#1'~contains~a~key--value~marker~'={#2}''.~
2959   This~is~only~supported~for~robust~commands,~but~not~for~expandable~ones.
2960   \c__cmd_ignore_def_tl
2961 }
2962 \msg_new:nnnn { cmd } { too-many-args }
2963 { Too-many-arguments-for~'#1'. }
2964 {
2965   The~argument~specification~'#2'~asks~for~more~than~9~arguments.~
2966   This~cannot~be~implemented.
2967   \c__cmd_ignore_def_tl
2968 }
2969 \msg_new:nnnn { cmd } { two-markers }
2970 { Invalid-argument-prefix~'#2'~in~#1. }
2971 {
2972   The~argument~specification~provided~for~#1~has~two~'#2'~markers~applied~
2973   to~the~same~argument;~one~is~redundant.
2974 }
2975 \msg_new:nnnn { cmd } { unknown-argument-type } % should be unkown-arg-type but dep in xparse
2976 { Invalid-argument-type~'#2'~in~#1. }
2977 {
2978   The~letter~'#2'~does~not~specify~a~known~argument~type.
2979   \c__cmd_ignore_def_tl
2980 }
2981 \msg_new:nnnn { cmd } { xparse-arg-type }
2982 { Invalid-argument-type~'#2'~in~#1~(requires~xparse). }
2983 {
2984   The~letter~'#2'~specifies~a~known~but~deprecated~argument~type.~
2985   If~you~really~need~it~you~have~to~load~the~xparse~package.
2986   \c__cmd_ignore_def_tl
2987 }

```

Errors when using commands/environments. The if-boolean message is always used in expandable errors. The default-loop and missing-required messages can be expandable or not expandable.

```

2988 \msg_new:nnn { cmd } { if-boolean }
2989 { Invalid-argument~{#1}~to~\iow_char:N\\IfBoolean... }
2990 \msg_new:nnnn { cmd } { default-loop }
2991 { Circular-dependency-in-defaults-of~#1. }
2992 {
2993   The~default~values~of~two~or~more~arguments~of~the~#1~
2994   depend~on~each~other~in~a~way~that~cannot~be~resolved.
2995 }
2996 \msg_new:nnnn { cmd } { missing-required }
2997 { Required-argument-missing-for~#1. }

```

```

2998 {
2999   The~#1~expects~one~of~its~arguments~to~start~with~'#2'~.~
3000   LaTeX~did~not~find~this~argument~and~will~insert~a~default~value~
3001   for~further~processing.
3002 }
3003 \msg_new:nnnn { cmd } { arg-split }
3004 { Too~many~'#1'~separators~in~argument. }
3005 {
3006   LaTeX~was~asked~to~split~the~input~'#3'~
3007   at~each~occurrence~of~the~separator~'#1'~into~#2~parts.~
3008   Too~many~separators~were~found.
3009 }
3010 \msg_new:nnnn { cmd } { verbatim-nl }
3011 { Verbatim~like~#1~ended~by~end~of~line. }
3012 {
3013   The~verbatim~argument~of~the~#1~cannot~contain~more~than~one~line,~
3014   but~the~end~
3015   of~the~current~line~has~been~reached.~You~may~have~forgotten~the~
3016   closing~delimiter.
3017   \\ \\
3018   LaTeX~will~ignore~'#2'~and~you~may~get~some~additional~
3019   (low-level)~errors.
3020 }
3021 \msg_new:nnnn { cmd } { verbatim-tokenized }
3022 { Verbatim~like~#1~illegal~in~argument. }
3023 {
3024   The~#1~takes~a~verbatim~argument~and~should~therefore~normally~
3025   not~be~used~in~arguments~of~other~commands~or~environments.~
3026   LaTeX~found~an~illegal~token~ \tl_if_empty:nF {#3} { (#3)~ }
3027   after~'#2'~and~will~drop~everything~up~to~this~point.
3028   \\ \\
3029   Expect~further~(low-level)~errors.
3030 }
    Intended more for information.
3031 \msg_new:nnn { cmd } { define-command } % should be just "define" but dep in xparse
3032 {
3033   Defining~command~#1~
3034   with~sig.~'#2'~\msg_line_context:.
3035 }
3036 \msg_new:nnn { cmd } { define-env }
3037 {
3038   Defining~environment~'#1'~
3039   with~sig.~'#2'~\msg_line_context:.
3040 }
3041 \msg_new:nnn { cmd } { redefine }
3042 {
3043   Redefining~command~#1~
3044   with~sig.~'#2'~\msg_line_context:.
3045 }
3046 \msg_new:nnn { cmd } { redefine-env }
3047 {
3048   Redefining~environment~'#1'~
3049   with~sig.~'#2'~\msg_line_context:.
3050 }

```

```

3051 \msg_new:nnn { cmd } { optional-mandatory }
3052 {
3053   Optional~and~mandatory~argument~with~same~delimiter~'#2'.
3054   \\ \\
3055   The~mandatory~argument~specified~with~
3056   '\str_case:nnF{#1}{ {R/r}{r~or~R} }{#1}'~has~the~
3057   same~delimiter~'#2'~as~an~earlier~optional~argument.~
3058   It~will~therefore~not~be~possible~to~omit~all~the~earlier~
3059   optional~arguments~when~calling~this~command.
3060   \\ \\
3061   This~may~be~intentional,~but~then~it~might~be~a~mistake.
3062 }
3063 \msg_new:nnn { cmd } { unsupported-let }
3064 {
3065   The~command~'#1'~was~undefined~but~not~the~associated~commands~
3066   '#1~code'~and/or~'#1~defaults'.~Maybe~you~tried~using~
3067   \iow_char:N\let.~This~may~lead~to~an~infinite~loop.
3068 }

```

1.14 User functions

The user functions are more or less just the internal functions renamed.

`\BooleanFalse` Design-space names for the Boolean values.

```

\BooleanTrue 3069 \cs_new_eq:NN \BooleanFalse \c_false_bool
3070 \cs_new_eq:NN \BooleanTrue \c_true_bool

```

(End of definition for \BooleanFalse and \BooleanTrue.)

`\NewDocumentCommand` The user macros are pretty simple wrappers around the internal ones. There is however
`\RenewDocumentCommand` a check that the first argument is a single token, possibly surrounded by spaces (hence
`\ProvideDocumentCommand` the strange `\use:nnn`), and is definable.
`\DeclareDocumentCommand`

```

3071 \cs_new_protected:Npn \NewDocumentCommand #1#2#3
3072 {
3073   \__cmd_check_definable:nNT {#1} \NewDocumentCommand
3074   {
3075     \cs_if_exist:NTF #1
3076     {
3077       \msg_error:nxxx { cmd } { already-defined }
3078       { \use:nnn \token_to_str:N #1 { } }
3079       { \token_to_str:N \NewDocumentCommand }
3080     }
3081     { \__cmd_declare_cmd:Nnn #1 {#2} {#3} }
3082   }
3083 }
3084 \cs_new_protected:Npn \RenewDocumentCommand #1#2#3
3085 {
3086   \__cmd_check_definable:nNT {#1} \RenewDocumentCommand
3087   {
3088     \cs_if_exist:NTF #1
3089     { \__cmd_declare_cmd:Nnn #1 {#2} {#3} }
3090     {
3091       \msg_error:nxxx { cmd } { undefined }
3092       { \use:nnn \token_to_str:N #1 { } }

```

```

3093         { \token_to_str:N \RenewDocumentCommand }
3094     }
3095 }
3096 }
3097 \cs_new_protected:Npn \ProvideDocumentCommand #1#2#3
3098 {
3099     \__cmd_check_definable:nNT {#1} \ProvideDocumentCommand
3100     { \cs_if_exist:NF #1 { \__cmd_declare_cmd:Nnn #1 {#2} {#3} } }
3101 }
3102 \cs_new_protected:Npn \DeclareDocumentCommand #1#2#3
3103 {
3104     \__cmd_check_definable:nNT {#1} \DeclareDocumentCommand
3105     { \__cmd_declare_cmd:Nnn #1 {#2} {#3} }
3106 }

```

(End of definition for `\NewDocumentCommand` and others.)

`\NewDocumentEnvironment` Very similar for environments. Trim spaces from user-specified `<envname>`, do existence check then hand off to `__cmd_declare_env:nnnn`.

```

\RenewDocumentEnvironment
\ProvideDocumentEnvironment
\DeclareDocumentEnvironment
3107 <latexrelease> \IncludeInRelease{2024/11/01}{\NewDocumentEnvironment}%
3108 <latexrelease> {Trim~spaces~from~envname~first}
3109 \cs_new_protected:Npn \NewDocumentEnvironment #1#2#3#4
3110 {
3111     \__cmd_new_env:nnnn { \tl_trim_spaces:e {#1} } {#2} {#3} {#4}
3112 }
3113 \cs_new_protected:Npn \RenewDocumentEnvironment #1#2#3#4
3114 {
3115     \__cmd_renew_env:ennn { \tl_trim_spaces:e {#1} } {#2} {#3} {#4}
3116 }
3117 \cs_new_protected:Npn \ProvideDocumentEnvironment #1#2#3#4
3118 {
3119     \__cmd_provide_env:ennn { \tl_trim_spaces:e {#1} } {#2} {#3} {#4}
3120 }
3121 \cs_new_protected:Npn \DeclareDocumentEnvironment #1#2#3#4
3122 {
3123     \__cmd_declare_env:ennn { \tl_trim_spaces:e {#1} } {#2} {#3} {#4}
3124 }

```

Each of `__cmd_(new|renew|provide)_env:nnnn` is curried.

```

3125 \cs_new_protected:Npn \__cmd_new_env:nnnn #1
3126 {
3127     \cs_if_exist:cTF {#1}
3128     {
3129         \msg_error:nnx { cmd } { env-already-defined } {#1}
3130         \use_none:nnn
3131     }
3132     {
3133         \cs_if_exist:cTF { end #1 }
3134         {
3135             \msg_error:nnx { cmd } { env-end-already-defined } {#1}
3136             \use_none:nnn
3137         }
3138         { \__cmd_declare_env:nnnn {#1} }
3139     }
3140 }

```

```

3141 \cs_new_protected:Npn \__cmd_renew_env:nmmm #1
3142 {
3143   \cs_if_exist:cTF {#1}
3144   { \__cmd_declare_env:nmmm {#1} }
3145   {
3146     \msg_error:nxx { cmd } { env-undefined } {#1}
3147     \use_none:nnn
3148   }
3149 }
3150 \cs_new_protected:Npn \__cmd_provide_env:nmmm #1
3151 {
3152   \cs_if_exist:cTF {#1}
3153   { \use_none:nnn }
3154   { \__cmd_declare_env:nmmm {#1} }
3155 }
3156 \cs_generate_variant:Nn \__cmd_new_env:nmmm { e }
3157 \cs_generate_variant:Nn \__cmd_renew_env:nmmm { e }
3158 \cs_generate_variant:Nn \__cmd_provide_env:nmmm { e }
3159 <latexrelease>\EndIncludeInRelease
3160 <latexrelease>\IncludeInRelease{2024/06/01}{\NewDocumentEnvironment}%
3161 <latexrelease>
3162 <latexrelease>\cs_new_protected:Npn \NewDocumentEnvironment #1#2#3#4
3163 <latexrelease> {
3164 <latexrelease>   \cs_if_exist:cTF {#1}
3165 <latexrelease>     { \msg_error:nxx { cmd } { env-already-defined } {#1} }
3166 <latexrelease>     {
3167 <latexrelease>       \cs_if_exist:cTF { end #1 }
3168 <latexrelease>         { \msg_error:nxx { cmd } { env-end-already-defined } {#1} }
3169 <latexrelease>         { \__cmd_declare_env:nmmm {#1} {#2} {#3} {#4} }
3170 <latexrelease>       }
3171 <latexrelease>     }
3172 <latexrelease>\cs_new_protected:Npn \RenewDocumentEnvironment #1#2#3#4
3173 <latexrelease> {
3174 <latexrelease>   \cs_if_exist:cTF {#1}
3175 <latexrelease>     { \__cmd_declare_env:nmmm {#1} {#2} {#3} {#4} }
3176 <latexrelease>     { \msg_error:nxx { cmd } { env-undefined } {#1} }
3177 <latexrelease>   }
3178 <latexrelease>\cs_new_protected:Npn \ProvideDocumentEnvironment #1#2#3#4
3179 <latexrelease> { \cs_if_exist:cF {#1} { \__cmd_declare_env:nmmm {#1} {#2} {#3} {#4} } }
3180 <latexrelease>\cs_new_protected:Npn \DeclareDocumentEnvironment #1#2#3#4
3181 <latexrelease> { \__cmd_declare_env:nmmm {#1} {#2} {#3} {#4} }
3182 <latexrelease>\cs_undefine:N \__cmd_new_env:nmmm
3183 <latexrelease>\cs_undefine:N \__cmd_new_env:enmm
3184 <latexrelease>\cs_undefine:N \__cmd_renew_env:nmmm
3185 <latexrelease>\cs_undefine:N \__cmd_renew_env:enmm
3186 <latexrelease>\cs_undefine:N \__cmd_provide_env:nmmm
3187 <latexrelease>\cs_undefine:N \__cmd_provide_env:enmm
3188 <latexrelease>
3189 <latexrelease>\EndIncludeInRelease

```

(End of definition for \NewDocumentEnvironment and others.)

\NewExpandableDocumentCommand The expandable versions are essentially the same as the basic functions. The strange
\RenewExpandableDocumentCommand \use:nnn is there in case #1 is surrounded with spaces, as can happen with usual docu-
\ProvideExpandableDocumentCommand
\DeclareExpandableDocumentCommand

ment catcodes in `\RenewExpandableDocumentCommand { \! } ...`

```
3190 \cs_new_protected:Npn \NewExpandableDocumentCommand #1#2#3
3191 {
3192   \__cmd_check_definable:nNT {#1} \NewExpandableDocumentCommand
3193   {
3194     \cs_if_exist:NTF #1
3195     {
3196       \msg_error:nxxx { cmd } { already-defined }
3197       { \use:nnn \token_to_str:N #1 { } }
3198       { \token_to_str:N \NewExpandableDocumentCommand }
3199     }
3200     { \__cmd_declare_expandable_cmd:Nnn #1 {#2} {#3} }
3201   }
3202 }
3203 \cs_new_protected:Npn \RenewExpandableDocumentCommand #1#2#3
3204 {
3205   \__cmd_check_definable:nNT {#1} \RenewExpandableDocumentCommand
3206   {
3207     \cs_if_exist:NTF #1
3208     { \__cmd_declare_expandable_cmd:Nnn #1 {#2} {#3} }
3209     {
3210       \msg_error:nxxx { cmd } { undefined }
3211       { \use:nnn \token_to_str:N #1 { } }
3212       { \token_to_str:N \RenewExpandableDocumentCommand }
3213     }
3214   }
3215 }
3216 \cs_new_protected:Npn \ProvideExpandableDocumentCommand #1#2#3
3217 {
3218   \__cmd_check_definable:nNT {#1} \ProvideExpandableDocumentCommand
3219   {
3220     \cs_if_exist:NF #1
3221     { \__cmd_declare_expandable_cmd:Nnn #1 {#2} {#3} }
3222   }
3223 }
3224 \cs_new_protected:Npn \DeclareExpandableDocumentCommand #1#2#3
3225 {
3226   \__cmd_check_definable:nNT {#1} \DeclareExpandableDocumentCommand
3227   { \__cmd_declare_expandable_cmd:Nnn #1 {#2} {#3} }
3228 }
```

(End of definition for `\NewExpandableDocumentCommand` and others.)

`\IfBooleanT` The logical `<true>` and `<false>` statements are just the normal `\c_true_bool` and `\c_false_bool` so `\bool_if:NTF` is almost enough. However, this code-level function blows up badly when passed invalid input. We want `\IfBooleanTF` to accept a single (non-space) token equal to `\c_true_bool` or `\c_false_bool`, possibly surrounded by spaces. If the input is blank or multiple items, jump to the error and pick the false branch. If the input, ignoring spaces (we do this by omitting braces in the `\tl_if_single_token:nF` test), is not a single token then jump to the error as well. It is then safe to compare the token to the two booleans, picking the appropriate branch. If neither matches, we jump to the error as well.

```
3229 \cs_new:Npn \IfBooleanTF #1
```

```

3230 {
3231   \tl_if_single:nF {#1}
3232     { \prg_break:n { \use:n } }
3233   \tl_if_single_token:nF #1
3234     { \prg_break:n { \use:n } }
3235   \token_if_eq_meaning:NNT #1 \c_true_bool
3236     { \prg_break:n { \use_ii:nnn } }
3237   \token_if_eq_meaning:NNT #1 \c_false_bool
3238     { \prg_break:n { \use_iii:nnn } }
3239   \prg_break:n { \use:n }
3240   \prg_break_point:
3241     {
3242       \msg_expandable_error:nnn { cmd } { if-boolean } {#1}
3243       \use_ii:nn
3244     }
3245 }
3246 \cs_new:Npn \IfBooleanT #1#2 { \IfBooleanTF {#1} {#2} { } }
3247 \cs_new:Npn \IfBooleanF #1 { \IfBooleanTF {#1} { } }

```

(End of definition for `\IfBooleanT`, `\IfBooleanF`, and `\IfBooleanTF`.)

`\IfNoValueT` Simple re-naming.

```

\IfNoValueF 3248 \cs_new_eq:NN \IfNoValueF \tl_if_novalue:nF
\IfNoValueTF 3249 \cs_new_eq:NN \IfNoValueT \tl_if_novalue:nT
3250 \cs_new_eq:NN \IfNoValueTF \tl_if_novalue:nTF

```

(End of definition for `\IfNoValueT`, `\IfNoValueF`, and `\IfNoValueTF`.)

`\IfValueT` Inverted logic.

```

\IfValueF 3251 \cs_new:Npn \IfValueF { \tl_if_novalue:nT }
\IfValueTF 3252 \cs_new:Npn \IfValueT { \tl_if_novalue:nF }
3253 \cs_new:Npn \IfValueTF #1#2#3 { \tl_if_novalue:nTF {#1} {#3} {#2} }

```

(End of definition for `\IfValueT`, `\IfValueF`, and `\IfValueTF`.)

`\IfBlankT` Another simple re-naming.

```

\IfBlankF 3254 <latexrelease>\IncludeInRelease{2022/06/01}%
\IfBlankTF 3255 <latexrelease> { \IfBlankTF } { Testing-for-empty-or-blank }%
3256 \cs_new_eq:NN \IfBlankF \tl_if_blank:nF
3257 \cs_new_eq:NN \IfBlankT \tl_if_blank:nT
3258 \cs_new_eq:NN \IfBlankTF \tl_if_blank:nTF
3259 <latexrelease>\EndIncludeInRelease
3260 <latexrelease>\IncludeInRelease{2021/11/15}%
3261 <latexrelease> { \IfBlankTF } { Testing-for-empty-or-blank }%
3262 <latexrelease>\cs_undefine:N \IfBlankF
3263 <latexrelease>\cs_undefine:N \IfBlankT
3264 <latexrelease>\cs_undefine:N \IfBlankTF
3265 <latexrelease>
3266 <latexrelease>\EndIncludeInRelease

```

(End of definition for `\IfBlankT`, `\IfBlankF`, and `\IfBlankTF`.)

`\ProcessedArgument` Processed arguments are returned using this name, which is reserved here although the definition will change.

```

3267 \tl_new:N \ProcessedArgument

```

(End of definition for `\ProcessedArgument`.)

`\ReverseBoolean` Simple copies.

```
\SplitArgument 3268 \cs_new_eq:NN \ReverseBoolean \__cmd_bool_reverse:N
  \SplitList    3269 \cs_new_eq:NN \SplitArgument \__cmd_split_argument:nnn
  \TrimSpaces   3270 \cs_new_eq:NN \SplitList \__cmd_split_list:nn
  \TrimSpaces   3271 \cs_new_eq:NN \TrimSpaces \__cmd_trim_spaces:n
```

(End of definition for `\ReverseBoolean` and others.)

`\ProcessList` To support `\SplitList`.

```
3272 \cs_new_eq:NN \ProcessList \tl_map_tokens:nn
```

(End of definition for `\ProcessList`.)

Finally as promised, restore `__kernel_chk_if_free_cs:N`:

```
3273 <latexrelease> \cs_gset_eq:NN \__kernel_chk_if_free_cs:N \__cmd_chk_if_free_cs:N
3274 <latexrelease> \cs_undefine:N \__cmd_chk_if_free_cs:N
3275 <latexrelease>
3276 <latexrelease> \IncludeInRelease{0000/00/00}{ltxcmd}%
3277 <latexrelease> \Document~command~parser}%
3278 <latexrelease>
3279 <latexrelease> \EndModuleRelease
3280 \ExplSyntaxOff
```

Now in `latexrelease` mode, redefine `\NewDocumentCommand` to not complain on commands already defined.

```
3281 <latexrelease> \@ifundefined{ExplSyntaxOff}{}{\latexrelease@postltxcmd}
3282 <latexrelease> \catcode'\^@=\@latexrelease@catcode@null\relax
3283 </2ekernel | latexrelease>
```

We need to stop `DocStrip` treating `@@` in a special way at this point.

```
3284 <@@= >
```


File 08

lthooks.dtx

1 Introduction

Hooks are points in the code of commands or environments where it is possible to add processing code into existing commands. This can be done by different packages that do not know about each other, and to allow for hopefully safe processing it is necessary to sort different chunks of code added by different packages into a suitable processing order.

This is done by the packages adding chunks of code (via `\AddToHook`) and labeling their code with some label by default using the package name as a label.

At `\begin{document}` all code for a hook is then sorted according to some rules (given by `\DeclareHookRule`) for fast execution without processing overhead. If the hook code is modified afterwards (or the rules are changed), a new version for fast processing is generated.

Some hooks are used already in the preamble of the document. If that happens then the hook is prepared for execution (and sorted) already at that point.

2 Package writer interface

The hook management system is offered as a set of CamelCase commands for traditional $\text{\LaTeX} 2_{\epsilon}$ packages (and for use in the document preamble if needed) as well as `expl3` commands for modern packages, that use the L3 programming layer of \LaTeX . Behind the scenes, a single set of data structures is accessed so that packages from both worlds can coexist and access hooks in other packages.

2.1 $\text{\LaTeX} 2_{\epsilon}$ interfaces

2.1.1 Declaring hooks

With a few exceptions, hooks have to be declared before they can be used. The exceptions are the generic hooks for commands and environments (executed at `\begin` and `\end`), and the generic hooks run when loading files (see section 3.1).

`\NewHook` `\NewHook {<hook>}`

Creates a new `<hook>`. If this hook is declared within a package it is suggested that its name is always structured as follows: `<package-name>/<hook-name>`. If necessary you can further subdivide the name by adding more `/` parts. If a hook name is already taken, an error is raised and the hook is not created.

The `<hook>` can be specified using the dot-syntax to denote the current package name. See section 2.1.5. The string `??` can't be used as a hook name because it has a special significance as a placeholder in hook rules.

`\NewReversedHook` `\NewReversedHook {<hook>}`

Like `\NewHook` declares a new `<hook>`. the difference is that the code chunks for this hook are in reverse order by default (those added last are executed first). Any rules for the hook are applied after the default ordering. See sections 2.3 and 2.4 for further details.

The `<hook>` can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

`\NewMirroredHookPair` `\NewMirroredHookPair {<hook-1>} {<hook-2>}`

A shorthand for `\NewHook{<hook-1>}\NewReversedHook{<hook-2>}`.

The `<hook>` can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

`\NewHookWithArguments` `\NewHookWithArguments {<hook>} {<number>}`

New: 2023-06-01

Creates a new `<hook>` whose code takes `<number>` arguments, and otherwise works exactly like `\NewHook`. Section 2.7 explains hooks with arguments.

The `<hook>` can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

`\NewReversedHookWithArguments` `\NewReversedHookWithArguments {<hook>} {<number>}`

New: 2023-06-01

Like `\NewReversedHook`, but creates a hook whose code takes `<number>` arguments. Section 2.7 explains hooks with arguments.

The `<hook>` can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

`\NewMirroredHookPairWithArguments` `\NewMirroredHookPairWithArguments {<hook-1>} {<hook-2>} {<number>}`

New: 2023-06-01

A shorthand for `\NewHookWithArguments{<hook-1>}{<number>}`

`\NewReversedHookWithArguments{<hook-2>}{<number>}`. Section 2.7 explains hooks with arguments.

The `<hook>` can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

2.1.2 Special declarations for generic hooks

The declarations here should normally not be used. They are available to provide support for special use cases mainly involving generic command hooks.

`\DisableGenericHook` `\DisableGenericHook {<hook>}`

After this declaration³ the `<hook>` is no longer usable: Any further attempt to add code to it will result in an error and any use, e.g., via `\UseHook`, will simply do nothing.

This is intended to be used with generic command hooks (see `lthcmdhooks-doc`) as depending on the definition of the command such generic hooks may be unusable. If that is known, a package developer can disable such hooks up front.

The `<hook>` can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

³In the 2020/06 release this command was called `\DisableHook`, but that name was misleading as it shouldn't be used to disable non-generic hooks.

`\ActivateGenericHook` `\ActivateGenericHook` $\langle hook \rangle$

This declaration activates a generic hook provided by a package/class (e.g., one used in code with `\UseHook` or `\UseOneTimeHook`) without it being explicitly declared with `\NewHook`). If the hook is already activated, this command does nothing.

Note that this command does not undo the effect of `\DisableGenericHook`. See section 2.6 for a discussion of when this declaration is appropriate.

2.1.3 Using hooks in code

Using a hook that is executing the code that has been associated with it is only allowed if the hook has been previously declared with `\NewHook`. For performance reason there are no runtime checks for this and it is the responsibility of the programmer of a package to ensure that all hooks that are used in a package (with one of the commands in this section) are declared first.

`\UseHook` `\UseHook` $\langle hook \rangle$

Execute the code stored in the $\langle hook \rangle$.

Before `\begin{document}` the fast execution code for a hook is not set up, so in order to use a hook there it is explicitly initialized first. As that involves assignments using a hook at those times is not 100% the same as using it after `\begin{document}`.

The $\langle hook \rangle$ *cannot* be specified using the dot-syntax. A leading `.` is treated literally.

`\UseHookWithArguments` `\UseHookWithArguments` $\langle hook \rangle$ $\langle number \rangle$ $\langle arg_1 \rangle$... $\langle arg_n \rangle$

New: 2023-06-01

Execute the code stored in the $\langle hook \rangle$ and pass the arguments $\langle arg_1 \rangle$ through $\langle arg_n \rangle$ to the $\langle hook \rangle$. Otherwise, it works exactly like `\UseHook`. The $\langle number \rangle$ should be the number of arguments declared for the hook. If the hook is not declared, this command does nothing and it will remove $\langle number \rangle$ items from the input. Section 2.7 explains hooks with arguments.

The $\langle hook \rangle$ *cannot* be specified using the dot-syntax. A leading `.` is treated literally.

`\UseOneTimeHook` `\UseOneTimeHook` $\langle hook \rangle$

Some hooks are only used (and can be only used) in one place, for example, those in `\begin{document}` or `\end{document}`. From that point onwards, adding to the hook through a defined `\langle addto-cmd \rangle` command (e.g., `\AddToHook` or `\AtBeginDocument`, etc.) would have no effect (as would the use of such a command inside the hook code itself). It is therefore customary to redefine `\langle addto-cmd \rangle` to simply process its argument, i.e., essentially make it behave like `\@firstofone`.

`\UseOneTimeHook` does that: it records that the hook has been consumed and any further attempt to add to it will result in executing the code to be added immediately.

Using `\UseOneTimeHook` several times with the same $\langle hook \rangle$ means that it only executes the first time it is used. For example, if it is used in a command that can be called several times then the hook executes during only the *first* invocation of that command; this allows its use as an “initialization hook”.

Mixing `\UseHook` and `\UseOneTimeHook` for the same $\langle hook \rangle$ should be avoided, but if this is done then neither will execute after the first `\UseOneTimeHook`.

The $\langle hook \rangle$ *cannot* be specified using the dot-syntax. A leading `.` is treated literally. See section 2.1.5 for details.

`\UseOneTimeHookWithArguments` `\UseOneTimeHookWithArguments` $\langle hook \rangle$ $\langle number \rangle$ $\langle arg_1 \rangle$... $\langle arg_n \rangle$

New: 2023-06-01

Works exactly like `\UseOneTimeHook`, but passes arguments $\langle arg_1 \rangle$ through $\langle arg_n \rangle$ to the $\langle hook \rangle$. The $\langle number \rangle$ should be the number of arguments declared for the hook. If the hook is not declared, this command does nothing and it will remove $\langle number \rangle$ items from the input.

It should be noted that after a one-time hook is used, it is no longer possible to use `\AddToHookWithArguments` or similar with that hook. `\AddToHook` continues to work as normal. Section 2.7 explains hooks with arguments.

The $\langle hook \rangle$ *cannot* be specified using the dot-syntax. A leading `.` is treated literally. See section 2.1.5 for details.

2.1.4 Updating code for hooks

In contrast to the commands from the previous section, declarations such as `\AddToHook` or `\DeclareHookRule` can be used even when the hook is not yet declared. The rationale is that the hook declaration may be in some package that is loaded later, or perhaps not loaded at all.

A side effect of this design is that misspellings do not raise an error but are simply regarded as declarations for hooks with a different name.

`\AddToHook` `\AddToHook` $\langle hook \rangle$ [$\langle label \rangle$] $\langle code \rangle$

Adds $\langle code \rangle$ to the $\langle hook \rangle$ labeled by $\langle label \rangle$. When the optional argument $\langle label \rangle$ is not provided, the $\langle default label \rangle$ is used (see section 2.1.5). If `\AddToHook` is used in a package/class, the $\langle default label \rangle$ is the package/class name, otherwise it is `top-level` (the `top-level` label is treated differently: see section 2.1.6).

If there already exists code under the $\langle label \rangle$ then the new $\langle code \rangle$ is appended to the existing one (even if this is a reversed hook). If you want to replace existing code under the $\langle label \rangle$, first apply `\RemoveFromHook`.

The hook doesn't have to exist for code to be added to it. However, if it is not declared, then obviously the added $\langle code \rangle$ will never be executed. This allows for hooks to work regardless of package loading order and enables packages to add to hooks from other packages without worrying whether they are actually used in the current document. See section 2.1.8.

The $\langle hook \rangle$ and $\langle label \rangle$ can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

`\AddToHookWithArguments` `\AddToHookWithArguments {<hook>} [<label>] {<code>}`

New: 2023-06-01

Works exactly like `\AddToHook`, except that the `<code>` can access the arguments passed to the hook using `#1`, `#2`, ..., `#n` (up to the number of arguments declared for the hook). If the `<code>` should contain *parameter tokens* (`#`) that are not supposed to be understood as the arguments of the hook, such tokens should be doubled. For example, with `\AddToHook` one can write:

```
\AddToHook{myhook}{\def\foo#1{Hello, #1!}}
```

but to achieve the same with `\AddToHookWithArguments`, one should write:

```
\AddToHookWithArguments{myhook}{\def\foo##1{Hello, ##1!}}
```

because in the latter case, `#1` refers to the first argument of the hook `myhook`. Section 2.7 explains hooks with arguments.

The `<hook>` and `<label>` can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

`\RemoveFromHook` `\RemoveFromHook {<hook>} [<label>]`

Removes any code labeled by `<label>` from the `<hook>`. When the optional argument `<label>` is not provided, the `<default label>` is used (see section 2.1.5).

If there is no code under the `<label>` in the `<hook>`, or if the `<hook>` does not exist, a warning is issued when you attempt to `\RemoveFromHook`, and the command is ignored. `\RemoveFromHook` should be used only when you know exactly what labels are in a hook. Typically this will be when some code gets added to a hook by a package, then later this code is removed by that same package. If you want to prevent the execution of code from another package, use the `voids` rule instead (see section 2.1.7).

If the optional `<label>` argument is `*`, then all code chunks are removed. This is rather dangerous as it may well drop code from other packages (that one may not know about); it should therefore not be used in packages but only in document preambles!

The `<hook>` and `<label>` can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

In contrast to the `voids` relationship between two labels in a `\DeclareHookRule` this is a destructive operation as the labeled code is removed from the hook data structure, whereas the relationship setting can be undone by providing a different relationship later.

A useful application for this declaration inside the document body is when one wants to temporarily add code to hooks and later remove it again, e.g.,

```
\AddToHook{env/quote/begin}{\small}
\begin{quote}
  A quote set in a smaller typeface
\end{quote}
...
\RemoveFromHook{env/quote/begin}
... now back to normal for further quotes
```

Note that you can't cancel the setting with

```
\AddToHook{env/quote/begin}{}
```

Important:
The `\RemoveFromHook` command should be only used if one has full control over the code chunk to be removed. In particular it should not be used to remove code chunks from other packages! For this the `voids` relation is provided.

because that only “adds” a further empty chunk of code to the hook. Adding `\normalsize` would work but that means the hook then contained `\small\normalsize` which means two font size changes for no good reason.

The above is only needed if one wants to typeset several quotes in a smaller typeface. If the hook is only needed once then `\AddToHookNext` is simpler, because it resets itself after one use.

`\AddToHookNext` `\AddToHookNext {<hook>} {<code>}`

Adds `<code>` to the next invocation of the `<hook>`. The code is executed after the normal hook code has finished and it is executed only once, i.e. it is deleted after it was used.

Using this declaration is a global operation, i.e., the code is not lost even if the declaration is used inside a group and the next invocation of the hook happens after the end of that group. If the declaration is used several times before the hook is executed then all code is executed in the order in which it was declared.⁴

If this declaration is used with a one-time hook then the code is only ever used if the declaration comes before the hook’s invocation. This is because, in contrast to `\AddToHook`, the code in this declaration is not executed immediately in the case when the invocation of the hook has already happened—in other words, this code will truly execute only on the next invocation of the hook (and in the case of a one-time hook there is no such “next invocation”). This gives you a choice: should my code execute always, or should it execute only at the point where the one-time hook is used (and not at all if this is impossible)? For both of these possibilities there are use cases.

It is possible to nest this declaration using the same hook (or different hooks): e.g.,

`\AddToHookNext{<hook>}{<code-1>\AddToHookNext{<hook>}{<code-2>}}`

will execute `<code-1>` next time the `<hook>` is used and at that point puts `<code-2>` into the `<hook>` so that it gets executed on following time the hook is run.

A hook doesn’t have to exist for code to be added to it. This allows for hooks to work regardless of package loading order. See section 2.1.8.

The `<hook>` can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

`\AddToHookNextWithArguments` `\AddToHookNextWithArguments {<hook>} {<code>}`

New: 2023-06-01

Works exactly like `\AddToHookNext`, but the `<code>` can contain references to the arguments of the `<hook>` as described for `\AddToHookWithArguments` above. Section 2.7 explains hooks with arguments.

The `<hook>` can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

`\ClearHookNext` `\ClearHookNext {<hook>}`

Normally `\AddToHookNext` is only used when you know precisely where it will apply and why you want some extra code at that point. However, there are a few use cases in which such a declaration needs to be canceled, for example, when discarding a page with `\DiscardShipoutBox` (but even then not always), and in such situations `\ClearHookNext` can be used.

⁴There is no mechanism to reorder such code chunks (or delete them).

2.1.5 Hook names and default labels

It is best practice to use `\AddToHook` in packages or classes *without specifying a `<label>`* because then the package or class name is automatically used, which is helpful if rules are needed, and avoids mistyping the `<label>`.

Using an explicit `<label>` is only necessary in very specific situations, e.g., if you want to add several chunks of code into a single hook and have them placed in different parts of the hook (by providing some rules).

The other case is when you develop a larger package with several sub-packages. In that case you may want to use the same `<label>` throughout the sub-packages in order to avoid that the labels change if you internally reorganize your code.

Except for `\UseHook`, `\UseOneTimeHook` and `\IfHookEmptyTF` (and their `expl3` interfaces `\hook_use:n`, `\hook_use_once:n` and `\hook_if_empty:nTF`), all `<hook>` and `<label>` arguments are processed in the same way: first, spaces are trimmed around the argument, then it is fully expanded until only character tokens remain. If the full expansion of the `<hook>` or `<label>` contains a non-expandable non-character token, a low-level `TeX` error is raised (namely, the `<hook>` is expanded using `TeX`'s `\csname... \endcsname`, as such, Unicode characters are allowed in `<hook>` and `<label>` arguments). The arguments of `\UseHook`, `\UseOneTimeHook`, and `\IfHookEmptyTF` are processed much in the same way except that spaces are not trimmed around the argument, for better performance.

It is not enforced, but highly recommended that the hooks defined by a package, and the `<labels>` used to add code to other hooks contain the package name to easily identify the source of the code chunk and to prevent clashes. This should be the standard practice, so this hook management code provides a shortcut to refer to the current package in the name of a `<hook>` and in a `<label>`. If the `<hook>` name or the `<label>` consist just of a single dot (`.`), or starts with a dot followed by a slash (`./`) then the dot denotes the `<default label>` (usually the current package or class name—see `\SetDefaultHookLabel`). A “.” or “./” anywhere else in a `<hook>` or in `<label>` is treated literally and is not replaced.

For example, inside the package `mypackage.sty`, the default label is `mypackage`, so the instructions:

```
\NewHook    {./hook}
\AddToHook {./hook}[.]{code}      % Same as \AddToHook{./hook}{code}
\AddToHook {./hook}[./sub]{code}
\DeclareHookRule{begindocument}{.}{before}{babel}
\AddToHook {file/foo.tex/after}{code}
```

are equivalent to:

```
\NewHook    {mypackage/hook}
\AddToHook {mypackage/hook}[mypackage]{code}
\AddToHook {mypackage/hook}[mypackage/sub]{code}
\DeclareHookRule{begindocument}{mypackage}{before}{babel}
\AddToHook {file/foo.tex/after}{code}                % unchanged
```

The `<default label>` is automatically set equal to the name of the current package or class at the time the package is loaded. If the hook command is used outside of a package, or the current file wasn't loaded with `\usepackage` or `\documentclass`, then the `top-level` is used as the `<default label>`. This may have exceptions—see `\PushDefaultHookLabel`.

This syntax is available in all $\langle label \rangle$ arguments and most $\langle hook \rangle$ arguments, both in the L^AT_EX 2_ε interface, and the L^AT_EX 3 interface described in section 2.2.

Important:

The dot-syntax is not available with \UseHook and some other commands that are typically used within code!

Note, however, that the replacement of . by the $\langle default label \rangle$ takes place when the hook command is executed, so actions that are somehow executed after the package ends will have the wrong $\langle default label \rangle$ if the dot-syntax is used. For that reason, this syntax is not available in \UseHook (and \hook_use:n) because the hook is most of the time used outside of the package file in which it was defined. This syntax is also not available in the hook conditionals \IfHookEmptyTF (and \hook_if_empty:nTF), because these conditionals are used in some performance-critical parts of the hook management code, and because they are usually used to refer to other package's hooks, so the dot-syntax doesn't make much sense.

In some cases, for example in large packages, one may want to separate the code in logical parts, but still use the main package name as the $\langle label \rangle$, then the $\langle default label \rangle$ can be set using \PushDefaultHookLabel{...}\PopDefaultHookLabel or \SetDefaultHookLabel{...}.

<code>\PushDefaultHookLabel</code>	<code>\PushDefaultHookLabel {$\langle default label \rangle$}</code>
<code>\PopDefaultHookLabel</code>	<code>$\langle code \rangle$</code>
	<code>\PopDefaultHookLabel</code>

\PushDefaultHookLabel sets the current $\langle default label \rangle$ to be used in $\langle label \rangle$ arguments, or when replacing a leading “.” (see above). \PopDefaultHookLabel reverts the $\langle default label \rangle$ to its previous value.

Inside a package or class, the $\langle default label \rangle$ is equal to the package or class name, unless explicitly changed. Everywhere else, the $\langle default label \rangle$ is top-level (see section 2.1.6) unless explicitly changed.

The effect of \PushDefaultHookLabel holds until the next \PopDefaultHookLabel. \usepackage (and \RequirePackage and \documentclass) internally use

```
\PushDefaultHookLabel{ $\langle package name \rangle$ }
 $\langle package code \rangle$ 
\PopDefaultHookLabel
```

to set the $\langle default label \rangle$ for the package or class file. Inside the $\langle package code \rangle$ the $\langle default label \rangle$ can also be changed with \SetDefaultHookLabel. \input and other file input-related commands from the L^AT_EX kernel do not use \PushDefaultHookLabel, so code within files loaded by these commands does *not* get a dedicated $\langle label \rangle$! (that is, the $\langle default label \rangle$ is the current active one when the file was loaded.)

Packages that provide their own package-like interfaces (TikZ's \usetikzlibrary, for example) can use \PushDefaultHookLabel and \PopDefaultHookLabel to set dedicated labels and to emulate \usepackage-like hook behavior within those contexts.

The top-level label is treated differently, and is reserved to the user document, so it is not allowed to change the $\langle default label \rangle$ to top-level.

`\SetDefaultHookLabel` `\SetDefaultHookLabel {⟨default label⟩}`

Similarly to `\PushDefaultHookLabel`, sets the current `⟨default label⟩` to be used in `⟨label⟩` arguments, or when replacing a leading “.”. The effect holds until the label is changed again or until the next `\PopDefaultHookLabel`. The difference between `\PushDefaultHookLabel` and `\SetDefaultHookLabel` is that the latter does not save the current `⟨default label⟩`.

This command is useful when a large package is composed of several smaller packages, but all should have the same `⟨label⟩`, so `\SetDefaultHookLabel` can be used at the beginning of each package file to set the correct label.

`\SetDefaultHookLabel` is not allowed in the main document, where the `⟨default label⟩` is `top-level` and there is no `\PopDefaultHookLabel` to end its effect. It is also not allowed to change the `⟨default label⟩` to `top-level`.

2.1.6 The `top-level` label

The `top-level` label, assigned to code added from the main document, is different from other labels. Code added to hooks (usually `\AtBeginDocument`) in the preamble is almost always to change something defined by a package, so it should go at the very end of the hook.

Therefore, code added in the `top-level` is always executed at the end of the hook, regardless of where it was declared. If the hook is reversed (see `\NewReversedHook`), the `top-level` chunk is executed at the very beginning instead.

Rules regarding `top-level` have no effect: if a user wants to have a specific set of rules for a code chunk, they should use a different label to said code chunk, and provide a rule for that label instead.

The `top-level` label is exclusive for the user, so trying to add code with that label from a package results in an error.

2.1.7 Defining relations between hook code

The default assumption is that code added to hooks by different packages are independent and the order in which they are executed is irrelevant. While this is true in many cases it is obviously false in others.

Before the hook management system was introduced packages had to take elaborate precautions to determine whether some other package had also been loaded (before or after) and then to find some ways to alter its behavior accordingly. In addition it was often the user’s responsibility to load packages in the right order so that alterations made by packages were done in that same order; and in some cases even altering the loading order wouldn’t resolve the conflicts.

With the new hook management system it is now possible to define rules (i.e., relationships) between code chunks added by different packages and to specify explicitly the order in which they should be processed.

The rules can be declared for hooks before the hook has been declared with `\NewHook` and they are allowed to refer to code labels that do not yet exist, e.g., because a package defining the code chunk with that label has not yet been loaded. When the hook code is finally sorted for fast execution, all rules that apply are acted on and the others are ignored.

This offers the flexibility needed to handle complicated relationships between code from different packages and to set this up beforehand in a way that is independent of whether or not the packages are actually loaded in a specific document. The downside

of this is that misspellings of hook names or code labels will not raise any error, instead the rule will simply never apply!

`\DeclareHookRule` `\DeclareHookRule` $\langle hook \rangle$ $\langle label1 \rangle$ $\langle relation \rangle$ $\langle label2 \rangle$

Defines a relation between $\langle label1 \rangle$ and $\langle label2 \rangle$ for a given $\langle hook \rangle$. If $\langle hook \rangle$ is ?? this defines a default relation for all hooks that use the two labels, i.e., that have chunks of code labeled with $\langle label1 \rangle$ and $\langle label2 \rangle$.

Currently, the supported relations are the following:

before or **<** Code for $\langle label1 \rangle$ comes before code for $\langle label2 \rangle$.

after or **>** Code for $\langle label1 \rangle$ comes after code for $\langle label2 \rangle$.

incompatible-warning Only code for either $\langle label1 \rangle$ or $\langle label2 \rangle$ can appear for that hook (a way to say that two packages—or parts of them—are incompatible). A warning is raised if both labels appear in the same hook.

incompatible-error Like **incompatible-error** but instead of a warning a L^AT_EX error is raised, and the code for both labels are dropped from that hook until the conflict is resolved.

voids Code for $\langle label1 \rangle$ overwrites code for $\langle label2 \rangle$. More precisely, code for $\langle label2 \rangle$ is dropped for that hook. This can be used, for example if one package is a superset in functionality of another one and therefore wants to undo code in some hook and replace it with its own version.

unrelated The order of code for $\langle label1 \rangle$ and $\langle label2 \rangle$ is irrelevant. This rule is there to undo an incorrect rule specified earlier.

There can only be a single relation between two labels for a given hook, i.e., a later `\DeclareHookRule` overwrites any previous declaration. In all cases rules specific to a given hook take precedence over default rules that use ?? as the $\langle hook \rangle$.

If a default rule is applied, it is done before reversing the label order in a reversed hook, e.g., **before** in a default rule effectively becomes **after** in such a hook. In contrast, a rule for a specific hook is always applied to the state after any reversal (i.e., the state you see when using `\ShowHook` on that hook).

The $\langle hook \rangle$ and $\langle label \rangle$ can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

`\ClearHookRule` `\ClearHookRule` $\langle hook \rangle$ $\langle label1 \rangle$ $\langle label2 \rangle$

Syntactic sugar for saying that $\langle label1 \rangle$ and $\langle label2 \rangle$ are unrelated for the given $\langle hook \rangle$.

`\DeclareDefaultHookRule <label1> <relation> <label2>`

This sets up a relation between `<label1>` and `<label2>` for all hooks unless overwritten by a specific rule for a hook. Useful for cases where one package has a specific relation to some other package, e.g., is `incompatible` or always needs a special ordering `before` or `after`. (Technically it is just a shorthand for using `\DeclareHookRule` with `??` as the hook name.)

If such a rule is applied to a reversed hook it behaves as if the rule is reversed (e.g., `after` becomes `before`) because those rules are applied first and then the order is reversed. The rationale is that in hook pairs (in which the ordering in one is reversed) default rules have to be reversed too in nearly all scenarios. If this is not the case, a default rule can't be used or has to be overwritten with an explicit `\DeclareHookRule` for that specific hook.

Declaring default rules is only supported in the document preamble.⁵

The `<label>` can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

2.1.8 Querying hooks

Simpler data types, like token lists, have three possible states; they can:

- exist and be empty;
- exist and be non-empty; and
- not exist (in which case emptiness doesn't apply);

Hooks are a bit more complicated: a hook may exist or not, and independently it may or may not be empty. This means that even a hook that doesn't exist may be non-empty and it can also be disabled.

This seemingly strange state may happen when, for example, package *A* defines hook `A/foo`, and package *B* adds some code to that hook. However, a document may load package *B* before package *A*, or may not load package *A* at all. In both cases some code is added to hook `A/foo` without that hook being defined yet, thus that hook is said to be non-empty, whereas it doesn't exist. Therefore, querying the existence of a hook doesn't imply its emptiness, neither does the other way around.

Given that code or rules can be added to a hook even if it doesn't physically exist yet, means that a querying its existence has no real use case (in contrast to other variables that can only be update if they have already been declared). For that reason only the test for emptiness has a public interface.

A hook is said to be empty when no code was added to it, either to its permanent code pool, or to its "next" token list. The hook doesn't need to be declared to have code added to its code pool. A hook is said to exist when it was declared with `\NewHook` or some variant thereof. Generic hooks such as `file` and `env` hooks are automatically declared when code is added to them.

⁵Trying to do so, e.g., via `\DeclareHookRule` with `??` has bad side-effects and is not supported (though not explicitly caught for performance reasons).

<code>\IfHookEmptyTF</code>	*	<code>\IfHookEmptyTF {<hook>} {<true code>} {<false code>}</code>
<code>\IfHookEmptyT</code>	*	Tests if the <code><hook></code> is empty (<i>i.e.</i> , no code was added to it using either <code>\AddToHook</code> or <code>\AddToHookNext</code>) or such code was removed again (via <code>\RemoveFromHook</code>), and branches to either <code><true code></code> or <code><false code></code> depending on the result.
<code>\IfHookEmptyF</code>	*	

The `<hook>` *cannot* be specified using the dot-syntax. A leading `.` is treated literally.

2.1.9 Displaying hook code

If one has to adjust the code execution in a hook using a hook rule it is helpful to get some information about the code associated with a hook, its current order and the existing rules.

<code>\ShowHook</code>	<code>\ShowHook {<hook>}</code>
<code>\LogHook</code>	<code>\LogHook {<hook>}</code>

Displays information about the `<hook>` such as

- the code chunks (and their labels) added to it,
- any rules set up to order them,
- the computed order in which the chunks are executed,
- any code executed on the next invocation only.

`\LogHook` prints the information to the `.log` file, and `\ShowHook` prints them to the terminal/command window and starts TeX's prompt (only in `\errorstopmode`) to wait for user action.

The `<hook>` can be specified using the dot-syntax to denote the current package name. See section [2.1.5](#).

Suppose a hook `example-hook` whose output of `\ShowHook{example-hook}` is:

```

1  -> The hook 'example-hook':
2  > Code chunks:
3  >   foo -> [code from package 'foo']
4  >   bar -> [from package 'bar']
5  >   baz -> [package 'baz' is here]
6  > Document-level (top-level) code (executed last):
7  >   -> [code from 'top-level']
8  > Extra code for next invocation:
9  >   -> [one-time code]
10 > Rules:
11 >   foo|baz with relation >
12 >   baz|bar with default relation <
13 > Execution order (after applying rules):
14 >   baz, foo, bar.
```

In the listing above, lines 3 to 5 show the three code chunks added to the hook and their respective labels in the format

`<label> -> <code>`

Line 7 shows the code chunk added by the user in the main document (labeled `top-level`) in the format

```
Document-level (top-level) code (executed  $\langle first/last \rangle$ ):
->  $\langle top-level code \rangle$ 
```

This code will be either the first or last code executed by the hook (`last` if the hook is normal, `first` if it is reversed). This chunk is not affected by rules and does not take part in sorting.

Line 9 shows the code chunk for the next execution of the hook in the format

```
->  $\langle next-code \rangle$ 
```

This code will be used and disappear at the next `\UseHook{example-hook}`, in contrast to the chunks mentioned earlier, which can only be removed from that hook by doing `\RemoveFromHook{label}[example-hook]`.

Lines 11 and 12 show the rules declared that affect this hook in the format

```
 $\langle label-1 \rangle | \langle label-2 \rangle$  with  $\langle default? \rangle$  relation  $\langle relation \rangle$ 
```

which means that the $\langle relation \rangle$ applies to $\langle label-1 \rangle$ and $\langle label-2 \rangle$, in that order, as detailed in `\DeclareHookRule`. If the relation is `default` it means that this rule applies to $\langle label-1 \rangle$ and $\langle label-2 \rangle$ in *all* hooks, (unless overridden by a non-default relation).

Finally, line 14 lists the labels in the hook after sorting; that is, in the order they will be executed when the hook is used.

2.1.10 Debugging hook code

```
\DebugHooksOn \DebugHooksOn ... \DebugHooksOff
\DebugHooksOff
```

Turn the debugging of hook code on or off. This displays most changes made to the hook data structures. The output is rather coarse and not really intended for normal use, but it can be helpful in case hooks do not work as expected. See also [2.1.9](#) for commands to inspect individual hooks.

2.2 L3 programming layer (expl3) interfaces

This is a quick summary of the L^AT_EX₃ programming interfaces for use with packages written in `expl3`. In contrast to the L^AT_EX_{2 ϵ} interfaces they always use mandatory arguments only, e.g., you always have to specify the $\langle label \rangle$ for a code chunk. We therefore suggest to use the declarations discussed in the previous section even in `expl3` packages, but the choice is yours.

```
\hook_new:n \hook_new:n {hook}
\hook_new_reversed:n \hook_new_reversed:n {hook}
\hook_new_pair:nn \hook_new_pair:nn {hook-1} {hook-2}
```

Creates a new $\langle hook \rangle$ with normal or reverse ordering of code chunks. `\hook_new_pair:nn` creates a pair of such hooks with $\{hook-2\}$ being a reversed hook. If a hook name is already taken, an error is raised and the hook is not created.

The $\langle hook \rangle$ can be specified using the dot-syntax to denote the current package name. See section [2.1.5](#).

<code>\hook_new_with_args:nn</code>	<code>\hook_new_with_args:nn <hook> <number></code>
<code>\hook_new_reversed_with_args:nn</code>	<code>\hook_new_reversed_with_args:nn <hook> <number></code>
<code>\hook_new_pair_with_args:nnn</code>	<code>\hook_new_pair_with_args:nnn <hook-1> <hook-2> <number></code>

New: 2023-06-01

Creates a new `<hook>` with normal or reverse ordering of code chunks, that takes `<number>` arguments from the input stream when it is used. `\hook_new_pair_with_args:nn` creates a pair of such hooks with `<hook-2>` being a reversed hook. If a hook name is already taken, an error is raised and the hook is not created.

The `<hook>` can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

<code>\hook_disable_generic:n</code>	<code>\hook_disable_generic:n <hook></code>
--------------------------------------	---

Marks `<hook>` as disabled. Any further attempt to add code to it or declare it, will result in an error and any call to `\hook_use:n` will simply do nothing.

This declaration is intended for use with generic hooks that are known not to work (see `ltxcmdhooks-doc`) if they receive code.

The `<hook>` can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

<code>\hook_activate_generic:n</code>	<code>\hook_activate_generic:n <hook></code>
---------------------------------------	--

This is like `\hook_new:n` but it does nothing if the hook was previously declared with `\hook_new:n`. This declaration should be used only in special situations, e.g., when a command from another package needs to be altered and it is not clear whether a generic `cmd` hook (for that command) has been previously explicitly declared.

Normally `\hook_new:n` should be used instead of this.

<code>\hook_use:n</code>	<code>\hook_use:n <hook></code>
--------------------------	---------------------------------------

<code>\hook_use:nnw</code>	<code>\hook_use:nnw <hook> <number> <arg1> ... <argn></code>
----------------------------	--

New: 2023-06-01

Executes the `<hook>` code followed (if set up) by the code for next invocation only, then empties that next invocation code. `\hook_use:nnw` should be used for hooks declared with arguments, and should be followed by as many brace groups as the declared number of arguments. The `<number>` should be the number of arguments declared for the hook. If the hook is not declared, this command does nothing and it will remove `<number>` items from the input.

The `<hook>` *cannot* be specified using the dot-syntax. A leading `.` is treated literally.

<code>\hook_use_once:n</code>	<code>\hook_use_once:n <hook></code>
-------------------------------	--

<code>\hook_use_once:nnw</code>	<code>\hook_use_once:nnw <hook> <number> <arg1> ... <argn></code>
---------------------------------	---

New: 2023-06-01

Changes the `<hook>` status so that from now on any addition to the hook code is executed immediately. Then execute any `<hook>` code already set up. `\hook_use_once:nnw` should be used for hooks declared with arguments, and should be followed by as many brace groups as the declared number of arguments. The `<number>` should be the number of arguments declared for the hook. If the hook is not declared, this command does nothing and it will remove `<number>` items from the input.

The `<hook>` *cannot* be specified using the dot-syntax. A leading `.` is treated literally.

`\hook_gput_code:nnn` `\hook_gput_code:nnn` $\langle hook \rangle$ $\langle label \rangle$ $\langle code \rangle$

`\hook_gput_code_with_args:nnn` `\hook_gput_code_with_args:nnn` $\langle hook \rangle$ $\langle label \rangle$ $\langle code \rangle$

New: 2023-06-01

Adds a chunk of $\langle code \rangle$ to the $\langle hook \rangle$ labeled $\langle label \rangle$. If the label already exists the $\langle code \rangle$ is appended to the already existing code.

If `\hook_gput_code_with_args:nnn` is used, the $\langle code \rangle$ can access the arguments passed to `\hook_use:nnw` (or `\hook_use_once:nnw`) with #1, #2, ..., #n (up to the number of arguments declared for the hook). In that case, if an actual parameter token should be added to the code, it should be doubled.

If code is added to an external $\langle hook \rangle$ (of the kernel or another package) then the convention is to use the package name as the $\langle label \rangle$ not some internal module name or some other arbitrary string.

The $\langle hook \rangle$ and $\langle label \rangle$ can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

`\hook_gput_next_code:nn` `\hook_gput_next_code:nn` $\langle hook \rangle$ $\langle code \rangle$

`\hook_gput_next_code_with_args:nn` `\hook_gput_next_code_with_args:nn` $\langle hook \rangle$ $\langle code \rangle$

New: 2023-06-01

Adds a chunk of $\langle code \rangle$ for use only in the next invocation of the $\langle hook \rangle$. Once used it is gone.

If `\hook_gput_next_code_with_args:nn` is used, the $\langle code \rangle$ can access the arguments passed to `\hook_use:nnw` (or `\hook_use_once:nnw`) with #1, #2, ..., #n (up to the number of arguments declared for the hook). In that case, if an actual parameter token should be added to the code, it should be doubled.

This is simpler than `\hook_gput_code:nnn`, the code is simply appended to the hook in the order of declaration at the very end, i.e., after all standard code for the hook got executed. Thus if one needs to undo what the standard does one has to do that as part of $\langle code \rangle$.

The $\langle hook \rangle$ can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

`\hook_gclear_next_code:n` `\hook_gclear_next_code:n` $\langle hook \rangle$

Undo any earlier `\hook_gput_next_code:nn`.

`\hook_gremove_code:nn` `\hook_gremove_code:nn` $\langle hook \rangle$ $\langle label \rangle$

Removes any code for $\langle hook \rangle$ labeled $\langle label \rangle$.

If there is no code under the $\langle label \rangle$ in the $\langle hook \rangle$, or if the $\langle hook \rangle$ does not exist, a warning is issued when you attempt to use `\hook_gremove_code:nn`, and the command is ignored.

If the second argument is `*`, then all code chunks are removed. This is rather dangerous as it drops code from other packages one may not know about, so think twice before using that!

The $\langle hook \rangle$ and $\langle label \rangle$ can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

`\hook_gset_rule:nmmn` `\hook_gset_rule:nmmn` $\langle\mathit{hook}\rangle$ $\langle\mathit{label1}\rangle$ $\langle\mathit{relation}\rangle$ $\langle\mathit{label2}\rangle$

Relate $\langle\mathit{label1}\rangle$ with $\langle\mathit{label2}\rangle$ when used in $\langle\mathit{hook}\rangle$. See `\DeclareHookRule` for the allowed $\langle\mathit{relation}\rangle$ s. If $\langle\mathit{hook}\rangle$ is ?? a default rule is specified.

The $\langle\mathit{hook}\rangle$ and $\langle\mathit{label}\rangle$ can be specified using the dot-syntax to denote the current package name. See section 2.1.5. The dot-syntax is parsed in both $\langle\mathit{label}\rangle$ arguments, but it usually makes sense to be used in only one of them.

`\hook_if_empty_p:n` * `\hook_if_empty:nTF` $\langle\mathit{hook}\rangle$ $\langle\mathit{true\ code}\rangle$ $\langle\mathit{false\ code}\rangle$

`\hook_if_empty:nTF` * Tests if the $\langle\mathit{hook}\rangle$ is empty (*i.e.*, no code was added to it using either `\AddToHook` or `\AddToHookNext`), and branches to either $\langle\mathit{true\ code}\rangle$ or $\langle\mathit{false\ code}\rangle$ depending on the result.

The $\langle\mathit{hook}\rangle$ *cannot* be specified using the dot-syntax. A leading . is treated literally.

`\hook_show:n` `\hook_show:n` $\langle\mathit{hook}\rangle$

`\hook_log:n` `\hook_log:n` $\langle\mathit{hook}\rangle$

Displays information about the $\langle\mathit{hook}\rangle$ such as

- the code chunks (and their labels) added to it,
- any rules set up to order them,
- the computed order in which the chunks are executed,
- any code executed on the next invocation only.

`\hook_log:n` prints the information to the .log file, and `\hook_show:n` prints them to the terminal/command window and starts TeX's prompt (only if `\errorstopmode`) to wait for user action.

The $\langle\mathit{hook}\rangle$ can be specified using the dot-syntax to denote the current package name. See section 2.1.5.

`\hook_debug_on:` `\hook_debug_on:`

`\hook_debug_off:` Turns the debugging of hook code on or off. This displays changes to the hook data.

2.3 On the order of hook code execution

Chunks of code for a $\langle\mathit{hook}\rangle$ under different labels are supposed to be independent if there are no special rules set up that define a relation between the chunks. This means that you can't make assumptions about the order of execution!

Suppose you have the following declarations:

```
\NewHook{myhook}
\AddToHook{myhook}[packageA]{\typeout{A}}
\AddToHook{myhook}[packageB]{\typeout{B}}
\AddToHook{myhook}[packageC]{\typeout{C}}
```

then executing the hook with `\UseHook` will produce the typeout A B C in that order. In other words, the execution order is computed to be `packageA`, `packageB`, `packageC` which you can verify with `\ShowHook{myhook}`:


```

-> The hook 'myhook':
> Code chunks:
>   packageA -> \typeout {A}
>   packageB -> \typeout {B}
>   packageC -> \typeout {C}
> Document-level (top-level) code (executed last):
>   ---
> Extra code for next invocation:
>   ---
> Rules:
>   ---
> Execution order:
>   packageA, packageB, packageC.

```

The reason is that the code chunks are internally saved in a property list and the initial order of such a property list is the order in which key-value pairs got added. However, that is only true if nothing other than adding happens!

Suppose, for example, you want to replace the code chunk for `packageA`, e.g.,

```

\RemoveFromHook{myhook}[packageA]
\AddToHook{myhook}[packageA]{\typeout{A alt}}

```

then your order becomes `packageB`, `packageC`, `packageA` because the label got removed from the property list and then re-added (at its end).

While that may not be too surprising, the execution order is also sometimes altered if you add a redundant rule, e.g. if you specify

```

\DeclareHookRule{myhook}{packageA}{before}{packageB}

```

instead of the previous lines we get

```

-> The hook 'myhook':
> Code chunks:
>   packageA -> \typeout {A}
>   packageB -> \typeout {B}
>   packageC -> \typeout {C}
> Document-level (top-level) code (executed last):
>   ---
> Extra code for next invocation:
>   ---
> Rules:
>   packageB|packageA with relation >
> Execution order (after applying rules):
>   packageA, packageC, packageB.

```

As you can see the code chunks are still in the same order, but in the execution order for the labels `packageB` and `packageC` have swapped places. The reason is that, with the rule there are two orders that satisfy it, and the algorithm for sorting happened to pick a different one compared to the case without rules (where it doesn't run at all as there is nothing to resolve). Incidentally, if we had instead specified the redundant rule

```

\DeclareHookRule{myhook}{packageB}{before}{packageC}

```

the execution order would not have changed.

In summary: it is not possible to rely on the order of execution unless there are rules that partially or fully define the order (in which you can rely on them being fulfilled).

2.4 The use of “reversed” hooks

You may have wondered why you can declare a “reversed” hook with `\NewReversedHook` and what that does exactly.

In short: the execution order of a reversed hook (without any rules!) is exactly reversed to the order you would have gotten for a hook declared with `\NewHook`.

This is helpful if you have a pair of hooks where you expect to see code added that involves grouping, e.g., starting an environment in the first and closing that environment in the second hook. To give a somewhat contrived example⁶, suppose there is a package adding the following:

```
\AddToHook{env/quote/before}[package-1]{\begin{itshape}}
\AddToHook{env/quote/after} [package-1]{\end{itshape}}
```

As a result, all quotes will be in italics. Now suppose further that another `package-too` makes the quotes also in blue and therefore adds:

```
\usepackage{color}
\AddToHook{env/quote/before}[package-too]{\begin{color}{blue}}
\AddToHook{env/quote/after} [package-too]{\end{color}}
```

Now if the `env/quote/after` hook would be a normal hook we would get the same execution order in both hooks, namely:

```
package-1, package-too
```

(or vice versa) and as a result, would get:

```
\begin{itshape}\begin{color}{blue} ...
\end{itshape}\end{color}
```

and an error message saying that `\begin{color}` was ended by `\end{itshape}`. With `env/quote/after` declared as a reversed hook the execution order is reversed and so all environments are closed in the correct sequence and `\ShowHook` would give us the following output:

```
-> The hook 'env/quote/after':
> Code chunks:
>   package-1 -> \end {itshape}
>   package-too -> \end {color}
> Document-level (top-level) code (executed first):
>   ---
> Extra code for next invocation:
>   ---
> Rules:
>   ---
> Execution order (after reversal):
>   package-too, package-1.
```

If there is a matching default rule (done with `\DeclareDefaultHookRule` or with `??` for the hook name) then this default rule is applied before the reversal so that the order in the reversed hook mirrors the one in the normal hook. However, all rules specific to a hook happen always after the reversal of the execution order, so if you alter the order you will probably have to alter it in both hooks, not just in one, but that depends on the use case.

⁶There are simpler ways to achieve the same effect.

2.5 Difference between “normal” and “one-time” hooks

When executing a hook a developer has the choice of using either `\UseHook` or `\UseOneTimeHook` (or their `expl3` equivalents `\hook_use:n` and `\hook_use_once:n`). This choice affects how `\AddToHook` is handled after the hook has been executed for the first time.

With normal hooks adding code via `\AddToHook` means that the code chunk is added to the hook data structure and then used each time `\UseHook` is called.

With one-time hooks it this is handled slightly differently: After `\UseOneTimeHook` has been called, any further attempts to add code to the hook via `\AddToHook` will simply execute the `<code>` immediately.

This has some consequences one needs to be aware of:

- If `<code>` is added to a normal hook after the hook was executed and it is never executed again for one or the other reason, then this new `<code>` will never be executed.
- In contrast if that happens with a one-time hook the `<code>` is executed immediately.

In particular this means that construct such as

```
\AddToHook{myhook}
  { <code-1> \AddToHook{myhook}{<code-2>} <code-3> }
```

works for one-time hooks⁷ (all three code chunks are executed one after another), but it makes little sense with a normal hook, because with a normal hook the first time `\UseHook{myhook}` is executed it would

- execute `<code-1>`,
- then execute `\AddToHook{myhook}{code-2}` which adds the code chunk `<code-2>` to the hook for use on the next invocation,
- and finally execute `<code-3>`.

The second time `\UseHook` is called it would execute the above and in addition `<code-2>` as that was added as a code chunk to the hook in the meantime. So each time the hook is used another copy of `<code-2>` is added and so that code chunk is executed `<# of invocations> - 1` times.

2.6 Generic hooks provided by packages

The hook management system also implements a category of hooks that are called “Generic Hooks”. Normally a hook has to be explicitly declared before it can be used in code. This ensures that different packages are not using the same hook name for unrelated purposes—something that would result in absolute chaos. However, there are a number of “standard” hooks where it is unreasonable to declare them beforehand, e.g. each and every command has (in theory) an associated **before** and **after** hook. In such cases, i.e., for command, environment or file hooks, they can be used simply by adding code to them with `\AddToHook`. For more specialized generic hooks, e.g., those provided

⁷This is sometimes used with `\AtBeginDocument` which is why it is supported.

by `babel`, you have to additionally enable them with `\ActivateGenericHook` as explained below.

The generic hooks provided by \LaTeX are those for `cmd`, `env`, `file`, `include`, `package`, and `class`, and all these are available out of the box: you only have to use `\AddToHook` to add code to them, but you don't have to add `\UseHook` or `\UseOneTimeHook` to your code, because this is already done for you (or, in the case of `cmd` hooks, the command's code is patched at `\begin{document}`, if necessary).

However, if you want to provide further generic hooks in your own code, the situation is slightly different. To do this you should use `\UseHook` or `\UseOneTimeHook`, but *without declaring the hook* with `\NewHook`. As mentioned earlier, a call to `\UseHook` with an undeclared hook name does nothing. So as an additional setup step, you need to explicitly activate your generic hook. Note that a generic hook produced in this way is always a normal hook.

For a truly generic hook, with a variable part in the hook name, such upfront activation would be difficult or impossible, because you typically do not know what kind of variable parts may come up in real documents.

For example, `babel` provides hooks such as `babel/⟨language⟩/afterextras`. However, language support in `babel` is often done through external language packages. Thus doing the activation for all languages inside the core `babel` code is not a viable approach. Instead it needs to be done by each language package (or by the user who wants to use a particular hook).

Because the hooks are not declared with `\NewHook` their names should be carefully chosen to ensure that they are (likely to be) unique. Best practice is to include the package or command name, as was done in the `babel` example above.

Generic hooks defined in this way are always normal hooks (i.e., you can't implement reversed hooks this way). This is a deliberate limitation, because it speeds up the processing considerably.

2.7 Hooks with arguments

Sometimes it is necessary to pass contextual information to a hook, and, for one reason or another, it is not feasible to store such information in macros. To serve this purpose, hooks can be declared with arguments, so that the programmer can pass along the data necessary for the code in the hook to function properly.

A hook with arguments works mostly like a regular hook, and most commands that work for regular hooks, also work for hooks that take arguments. The differences are when the hook is declared (`\NewHookWithArguments` is used instead of `\NewHook`), then code can be added with both `\AddToHook` and `\AddToHookWithArguments`, and when the hook is used (`\UseHookWithArguments` instead of `\UseHook`).

A hook with arguments must be declared as such (before it is first used, as all regular hooks) using `\NewHookWithArguments{⟨hook⟩}{⟨number⟩}`. All code added to that hook can then use `#1` to access the first argument, `#2` to access the second, and so forth up to the number of arguments declared. However, it is still possible to add code with references to the arguments of a hook that was not yet declared (we will discuss that later). At their core, hooks are macros, so \TeX 's limit of 9 arguments applies, and a low-level \TeX error is raised if you try to reference an argument number that doesn't exist.

To use a hook with arguments, just write `\UseHookWithArguments{<hook>}{<number>}` followed by a braced list of the arguments. For example, if the hook `test` takes three arguments, write:

```
\UseHookWithArguments{test}{3}{arg-1}{arg-2}{arg-3}
```

then, in the `<code>` of the hook, all instances of `#1` will be replaced by `arg-1`, `#2` by `arg-2` and so on. If, at the point of usage, the programmer provides more arguments than the hook is declared to take, the excess arguments are simply ignored by the hook. Behavior is unpredictable⁸ if too few arguments are provided. If the hook isn't declared, `<number>` arguments are removed from the input stream.

Adding code to a hook with arguments can be done with `\AddToHookWithArguments` as well as with the regular `\AddToHook`, to achieve different outcomes. The main difference when it comes to adding code to a hook, in this case, is firstly the possibility of accessing a hook's arguments, of course, and second, how parameter tokens (`#6`) are treated.

Using `\AddToHook` in a hook that takes arguments will work as it does for all other hooks. This allows a package developer to add arguments to a hook that otherwise had none without having to worry about compatibility. This means that, for example:

```
\AddToHook{test}{\def\foo#1{Hello, #1!}}
```

will define the same macro `\foo` regardless if the hook `test` takes arguments or not.

Using `\AddToHookWithArguments` allows the `<code>` added to access the arguments of the hook with `#1`, `#2`, and so forth, up to the number of the arguments declared in the hook. This means that if one wants to add a `#6` to the `<code>` that token must be doubled in the input. The same definition from above, using `\AddToHookWithArguments`, needs to be rewritten:

```
\AddToHookWithArguments{test}{\def\foo##1{Hello, ##1!}}
```

Extending the above example to use the hook arguments, we could rewrite something like (now from declaration to usage, to get the whole picture):

```
\NewHookWithArguments{test}{1}
\AddToHookWithArguments{test}{%
  \typeout{Defining foo with "#1"}
  \def\foo##1{Hello, ##1! Some text after: #1}%
}
\UseHook{test}{Howdy!}
\ShowCommand\foo
```

Running the code above prints in the terminal:

```
Defining foo with "Howdy!"
> \foo=macro:
#1->Hello, #1! Some text after: Howdy!.
```

⁸The hook *will* take the declared number of arguments, and what will happen depends on what was grabbed, and what the hook code does with its arguments.

Note how `##1` in the call to `\AddToHookWithArguments` became `#1`, and the `#1` was replaced by the argument passed to the hook. Should the hook be used again, with a different argument, the definition would naturally change.

It is possible to add code referencing a hook’s arguments before such hook is declared and the number of hooks is fixed. However, if some code is added to the hook, that references more arguments than will be declared for the hook, there will be a low-level `TeX` error about an “Illegal parameter number” at the time the hook is declared, which will be hard to track down because at that point `TeX` can’t know whence the offending code came from. Thus it is important that package writers explicitly document how many arguments (if any) each hook can take, so users of those packages know how many arguments can be referenced, and equally important, what each argument means.

2.8 Private `LaTeX` kernel hooks

There are a few places where it is absolutely essential for `LaTeX` to function correctly that code is executed in a precisely defined order. Even that could have been implemented with the hook management (by adding various rules to ensure the appropriate ordering with respect to other code added by packages). However, this makes every document unnecessary slow, because there has to be sorting even though the result is predetermined. Furthermore it forces package writers to unnecessarily add such rules if they add further code to the hook (or break `LaTeX`).

For that reason such code is not using the hook management, but instead private kernel commands directly before or after a public hook with the following naming convention: `\@kernel@before@hook` or `\@kernel@after@hook`. For example, in `\enddocument` you find

```
\UseHook{enddocument}%
\@kernel@after@enddocument
```

which means first the user/package-accessible `enddocument` hook is executed and then the internal kernel hook. As their name indicates these kernel commands should not be altered by third-party packages, so please refrain from that in the interest of stability and instead use the public hook next to it.⁹

2.9 Legacy `LaTeX 2ε` interfaces

`LaTeX 2ε` offered a small number of hooks together with commands to add to them. They are listed here and are retained for backwards compatibility.

With the new hook management, several additional hooks have been added to `LaTeX` and more will follow. See the next section for what is already available.

⁹As with everything in `TeX` there is not enforcement of this rule, and by looking at the code it is easy to find out how the kernel adds to them. The main reason of this section is therefore to say “please don’t do that, this is unconfigurable code!”

`\AtBeginDocument` `\AtBeginDocument` [*label*] {*code*}

If used without the optional argument `<label>`, it works essentially like before, i.e., it is adding `<code>` to the hook `begindocument` (which is executed inside `\begin{document}`). However, all code added this way is labeled with the label `top-level` (see section 2.1.6) if done outside of a package or class or with the package/class name if called inside such a file (see section 2.1.5).

This way one can add code to the hook using `\AddToHook` or `\AtBeginDocument` using a different label and explicitly order the code chunks as necessary, e.g., run some code before or after another package's code. When using the optional argument the call is equivalent to running `\AddToHook {begindocument} [label] {code}`.

`\AtBeginDocument` is a wrapper around the `begindocument` hook (see section 3.2), which is a one-time hook. As such, after the `begindocument` hook is executed at `\begin{document}` any attempt to add `<code>` to this hook with `\AtBeginDocument` or with `\AddToHook` will cause that `<code>` to execute immediately instead. See section 2.5 for more on one-time hooks.

For important packages with known order requirement we may over time add rules to the kernel (or to those packages) so that they work regardless of the loading-order in the document.

`\AtEndDocument` `\AtEndDocument` [*label*] {*code*}

Like `\AtBeginDocument` but for the `enddocument` hook.

The few hooks that existed previously in $\text{\LaTeX} 2_{\epsilon}$ used internally commands such as `@begindocumenthook` and packages sometimes augmented them directly rather than working through `\AtBeginDocument`. For that reason there is currently support for this, that is, if the system detects that such an internal legacy hook command contains code it adds it to the new hook system under the label `legacy` so that it doesn't get lost.

However, over time the remaining cases of direct usage need updating because in one of the future release of \LaTeX we will turn this legacy support off, as it does unnecessary slow down the processing.

3 $\text{\LaTeX} 2_{\epsilon}$ commands and environments augmented by hooks

In this section we describe the standard hooks that are now offered by \LaTeX , or give pointers to other documents in which they are described. This section will grow over time (and perhaps eventually move to `usrguide3`).

3.1 Generic hooks

As stated earlier, with the exception of generic hooks, all hooks must be declared with `\NewHook` before they can be used. All generic hooks have names of the form “`<type>/<name>/<position>`”, where `<type>` is from the predefined list shown below, and `<name>` is the variable part whose meaning will depend on the `<type>`. The last component, `<position>`, has more complex possibilities: it can always be `before` or `after`; for `env` hooks, it can also be `begin` or `end`; and for `include` hooks it can also be `end`. Each specific hook is documented below, or in `lthooks-doc.pdf` or `lthook-doc.pdf`.

The generic hooks provided by \LaTeX belong to one of the six types:

- env** Hooks executed before and after environments – $\langle name \rangle$ is the name of the environment, and available values for $\langle position \rangle$ are `before`, `begin`, `end`, and `after`;
- cmd** Hooks added to and executed before and after commands – $\langle name \rangle$ is the name of the command, and available values for $\langle position \rangle$ are `before` and `after`;
- file** Hooks executed before and after reading a file – $\langle name \rangle$ is the name of the file (with extension), and available values for $\langle position \rangle$ are `before` and `after`;
- package** Hooks executed before and after loading packages – $\langle name \rangle$ is the name of the package, and available values for $\langle position \rangle$ are `before` and `after`;
- class** Hooks executed before and after loading classes – $\langle name \rangle$ is the name of the class, and available values for $\langle position \rangle$ are `before` and `after`;
- include** Hooks executed before and after `\included` files – $\langle name \rangle$ is the name of the included file (without the `.tex` extension), and available values for $\langle position \rangle$ are `before`, `end`, and `after`.

Each of the hooks above are detailed in the following sections and in linked documentation.

3.1.1 Generic hooks for all environments

Every environment $\langle env \rangle$ has now four associated hooks coming with it:

- env/ $\langle env \rangle$ /before** This hook is executed as part of `\begin` as the very first action, in particular prior to starting the environment group. Its scope is therefore not restricted by the environment.
- env/ $\langle env \rangle$ /begin** This hook is executed as part of `\begin` directly in front of the code specific to the environment start (e.g., the third argument of `\NewDocumentEnvironment` and the second argument of `\newenvironment`). Its scope is the environment body.
- env/ $\langle env \rangle$ /end** This hook is executed as part of `\end` directly in front of the code specific to the end of the environment (e.g., the fourth argument of `\NewDocumentEnvironment` and the third argument of `\newenvironment`).
- env/ $\langle env \rangle$ /after** This hook is executed as part of `\end` after the code specific to the environment end and after the environment group has ended. Its scope is therefore not restricted by the environment.

The hook is implemented as a reversed hook so if two packages add code to `env/ $\langle env \rangle$ /before` and to `env/ $\langle env \rangle$ /after` they can add surrounding environments and the order of closing them happens in the right sequence.

Given that these generic hook names involve `/` as part of their name they would not work if one tries to define an environment using a name that involves a `/`.¹⁰

Generic environment hooks are never one-time hooks even with environments that are supposed to appear only once in a document.¹¹ In contrast to other hooks there is also no need to declare them using `\NewHook`.

¹⁰Officially, L^AT_EX names for environments should only consist of a sequence of letters, numbers, and the character `*`, i.e., this is not a new restriction.

¹¹Thus if one adds code to such hooks after the environment has been processed, it will only be executed if the environment appears again and if that doesn't happen the code will never get executed.

The hooks are only executed if `\begin{env}` and `\end{env}` is used. If the environment code is executed via low-level calls to `\{env}` and `\end{env}` (e.g., to avoid the environment grouping) they are not available. If you want them available in code using this method, you would need to add them yourself, i.e., write something like

```
\UseHook{env/quote/before}\quote
...
\endquote\UseHook{env/quote/after}
```

to add the outer hooks, etc.

Largely for compatibility with existing packages, the following four commands are also available to set the environment hooks; but for new packages we recommend directly using the hook names and `\AddToHook`.

`\BeforeBeginEnvironment` `\BeforeBeginEnvironment` [*label*] {*env*} {*code*}

This declaration adds to the `env/{env}/before` hook using the *label*. If *label* is not given, the *default label* is used (see section 2.1.5).

`\AtBeginEnvironment` `\AtBeginEnvironment` [*label*] {*env*} {*code*}

This is like `\BeforeBeginEnvironment` but it adds to the `env/{env}/begin` hook.

`\AtEndEnvironment` `\AtEndEnvironment` [*label*] {*env*} {*code*}

This is like `\BeforeBeginEnvironment` but it adds to the `env/{env}/end` hook.

`\AfterEndEnvironment` `\AfterEndEnvironment` [*label*] {*env*} {*code*}

This is like `\BeforeBeginEnvironment` but it adds to the `env/{env}/after` hook.

3.1.2 Generic hooks for commands

Similar to environments there are now (at least in theory) two generic hooks available for any \LaTeX command. These are

`cmd/{name}/before` This hook is executed at the very start of the command execution.

`cmd/{name}/after` This hook is executed at the very end of the command body. It is implemented as a reversed hook.

In practice there are restrictions and especially the `after` hook works only with a subset of commands. Details about these restrictions are documented in `ltxcmdhooks-doc.pdf` or with code in `ltxcmdhooks-code.pdf`.

3.1.3 Generic hooks provided by file loading operations

There are several hooks added to \LaTeX 's process of loading file via its high-level interfaces such as `\input`, `\include`, `\usepackage`, `\RequirePackage`, etc. These are documented in `ltxfilehook-doc.pdf` or with code in `ltxfilehook-code.pdf`.

3.2 Hooks provided by `\begin{document}`

Until 2020 `\begin{document}` offered exactly one hook that one could add to using `\AtBeginDocument`. Experiences over the years have shown that this single hook in one place was not enough and as part of adding the general hook management system a number of additional hooks have been added at this point. The places for these hooks have been chosen to provide the same support as offered by external packages, such as `etoolbox` and others that augmented `\document` to gain better control.

Supported are now the following hooks (all of them one-time hooks):

`begindocument/before` This hook is executed at the very start of `\document`, one can think of it as a hook for code at the end of the preamble section and this is how it is used by `etoolbox`'s `\AtEndPreamble`.

This is a one-time hook, so after it is executed, all further attempts to add code to it will execute such code immediately (see section 2.5).

`begindocument` This hook is added to by using `\AddToHook{begindocument}` or by using `\AtBeginDocument` and it is executed after the `.aux` file has been read and most initialization are done, so they can be altered and inspected by the hook code. It is followed by a small number of further initializations that shouldn't be altered and are therefore coming later.

The hook should not be used to add material for typesetting as we are still in `LATEX`'s initialization phase and not in the document body. If such material needs to be added to the document body use the next hook instead.

This is a one-time hook, so after it is executed, all further attempts to add code to it will execute such code immediately (see section 2.5).

`begindocument/end` This hook is executed at the end of the `\document` code in other words at the beginning of the document body. The only command that follows it is `\ignorespaces`.

This is a one-time hook, so after it is executed, all further attempts to add code to it will execute such code immediately (see section 2.5).

The generic hooks executed by `\begin` also exist, i.e., `env/document/before` and `env/document/begin`, but with this special environment it is better use the dedicated one-time hooks above.

3.3 Hooks provided by `\end{document}`

`LATEX 2ε` has always provided `\AtEndDocument` to add code to the `\end{document}`, just in front of the code that is normally executed there. While this was a big improvement over the situation in `LATEX 2.09`, it was not flexible enough for a number of use cases and so packages, such as `etoolbox`, `atveryend` and others patched `\enddocument` to add additional points where code could be hooked into.

Patching using packages is always problematical as leads to conflicts (code availability, ordering of patches, incompatible patches, etc.). For this reason a number of additional hooks have been added to the `\enddocument` code to allow packages to add code in various places in a controlled way without the need for overwriting or patching the core code.

Supported are now the following hooks (all of them one-time hooks):

enddocument The hook associated with `\AtEndDocument`. It is immediately called at the beginning of `\enddocument`.

When this hook is executed there may be still unprocessed material (e.g., floats on the deferlist) and the hook may add further material to be typeset. After it, `\clearpage` is called to ensure that all such material gets typeset. If there is nothing waiting the `\clearpage` has no effect.

This is a one-time hook, so after it is executed, all further attempts to add code to it will execute such code immediately (see section 2.5).

enddocument/afterlastpage As the name indicates this hook should not receive code that generates material for further pages. It is the right place to do some final housekeeping and possibly write out some information to the `.aux` file (which is still open at this point to receive data, but since there will be no more pages you need to write to it using `\immediate\write`). It is also the correct place to set up any testing code to be run when the `.aux` file is re-read in the next step.

After this hook has been executed the `.aux` file is closed for writing and then read back in to do some tests (e.g., looking for missing references or duplicated labels, etc.).

This is a one-time hook, so after it is executed, all further attempts to add code to it will execute such code immediately (see section 2.5).

enddocument/afteraux At this point, the `.aux` file has been reprocessed and so this is a possible place for final checks and display of information to the user. However, for the latter you might prefer the next hook, so that your information is displayed after the (possibly longish) list of files if that got requested via `\listfiles`.

This is a one-time hook, so after it is executed, all further attempts to add code to it will execute such code immediately (see section 2.5).

enddocument/info This hook is meant to receive code that write final information messages to the terminal. It follows immediately after the previous hook (so both could have been combined, but then packages adding further code would always need to also supply an explicit rule to specify where it should go.

This hook already contains some code added by the kernel (under the labels `kernel/filelist` and `kernel/warnings`), namely the list of files when `\listfiles` has been used and the warnings for duplicate labels, missing references, font substitutions etc.

This is a one-time hook, so after it is executed, all further attempts to add code to it will execute such code immediately (see section 2.5).

enddocument/end Finally, this hook is executed just in front of the final call to `\@@end`.

This is a one-time hook, so after it is executed, all further attempts to add code to it will execute such code immediately (see section 2.5).is it even possible to add code after this one?

There is also the hook `shipout/lastpage`. This hook is executed as part of the last `\shipout` in the document to allow package to add final `\special`'s to that page. Where this hook is executed in relation to those from the above list can vary from document to document. Furthermore to determine correctly which of the `\shipouts` is the last one,

L^AT_EX needs to be run several times, so initially it might get executed on the wrong page. See section 3.4 for where to find the details.

It is also possible to use the generic `env/document/end` hook which is executed by `\end`, i.e., just in front of the first hook above. Note however that the other generic `\end` environment hook, i.e., `env/document/after` will never get executed, because by that time L^AT_EX has finished the document processing.

3.4 Hooks provided by `\shipout` operations

There are several hooks and mechanisms added to L^AT_EX's process of generating pages. These are documented in `ltshipout-doc.pdf` or with code in `ltshipout-code.pdf`.

3.5 Hooks provided for paragraphs

The paragraph processing has been augmented to include a number of internal and public hooks. These are documented in `ltpara-doc.pdf` or with code in `ltpara-code.pdf`.

3.6 Hooks provided in NFSS commands

In languages that need to support for more than one script in parallel (and thus several sets of fonts, e.g., supporting both Latin and Japanese fonts), NFSS font commands such as `\sffamily` need to switch both the Latin family to “Sans Serif” and in addition alter a second set of fonts.

To support this, several NFSS commands have hooks to which such support can be added.

rmfamily After `\rmfamily` has done its initial checks and prepared a font series update, this hook is executed before `\selectfont`.

sffamily This is like the `rmfamily` hook, but for the `\sffamily` command.

ttfamily This is like the `rmfamily` hook, but for the `\ttfamily` command.

normalfont The `\normalfont` command resets the font encoding, family, series and shape to their document defaults. It then executes this hook and finally calls `\selectfont`.

expand@font@defaults The internal `\expand@font@defaults` command expands and saves the current defaults for the meta families (rm/sf/tt) and the meta series (bf/md). If the NFSS machinery has been augmented, e.g., for Chinese or Japanese fonts, then further defaults may need to be set at this point. This can be done in this hook which is executed at the end of this macro.

bfseries/defaults, bfseries If the `\bfdefault` was explicitly changed by the user, its new value is used to set the bf series defaults for the meta families (rm/sf/tt) when `\bfseries` is called. The `bfseries/defaults` hook allows further adjustments to be made in this case. This hook is only executed if such a change is detected. In contrast, the `bfseries` hook is always executed just before `\selectfont` is called to change to the new series.

mdseries/defaults, mdseries These two hooks are like the previous ones but they are in the `\mdseries` command.

selectfont This hook is executed inside `\selectfont`, after the current values for *encoding*, *family*, *series*, *shape*, and *size* are evaluated and the new font is selected (and if necessary loaded). After the hook has executed, NFSS will still do any updates necessary for a new *size* (such as changing the size of `\strut`) and any updates necessary to a change in *encoding*.

This hook is intended for use cases where, in parallel to a change in the main font, some other fonts need to be altered (e.g., in CJK processing where you may need to deal with several different alphabets).

3.7 Hook provided by the mark mechanism

See `ltmarks-doc.pdf` for details.

insertmark This hook allows for a special setup while `\InsertMark` inserts a mark. It is executed in group so local changes only apply to the mark being inserted.

4 The Implementation

```

1 <@@=hook>
2 <{*2ekernel | latexrelease>
3 \ExplSyntaxOn
4 <latexrelease> \NewModuleRelease{2020/10/01}{lthooks}
5 <latexrelease> {The~hook~management~system}

```

4.1 Debugging

```

\g__hook_debug_bool Holds the current debugging state.
6 \bool_new:N \g__hook_debug_bool
(End of definition for \g__hook_debug_bool.)

\hook_debug_on: Turns debugging on and off by redefining \__hook_debug:n.
\hook_debug_off:
\__hook_debug:n
\__hook_debug_gset:
7 \cs_new_eq:NN \__hook_debug:n \use_none:n
8 \cs_new_protected:Npn \hook_debug_on:
9 {
10 \bool_gset_true:N \g__hook_debug_bool
11 \__hook_debug_gset:
12 }
13 \cs_new_protected:Npn \hook_debug_off:
14 {
15 \bool_gset_false:N \g__hook_debug_bool
16 \__hook_debug_gset:
17 }
18 \cs_new_protected:Npn \__hook_debug_gset:
19 {
20 \cs_gset_protected:Npx \__hook_debug:n ##1
21 { \bool_if:NT \g__hook_debug_bool {##1} }
22 }

```

(End of definition for `\hook_debug_on:` and others. These functions are documented on page 214.)

4.2 Borrowing from internals of other kernel modules

`__hook_str_compare:nn` Private copy of `__str_if_eq:nn`
23 `\cs_new_eq:NN __hook_str_compare:nn __str_if_eq:nn`
(End of definition for `__hook_str_compare:nn`.)

4.3 Declarations

`\l__hook_tmpa_bool` Scratch boolean used throughout the package.
24 `\bool_new:N \l__hook_tmpa_bool`
(End of definition for `\l__hook_tmpa_bool`.)

`\l__hook_return_tl` Scratch variables used throughout the package.
`\l__hook_tmpa_tl` 25 `\tl_new:N \l__hook_return_tl`
`\l__hook_tmpb_tl` 26 `\tl_new:N \l__hook_tmpa_tl`
27 `\tl_new:N \l__hook_tmpb_tl`
(End of definition for `\l__hook_return_tl`, `\l__hook_tmpa_tl`, and `\l__hook_tmpb_tl`.)

`\g__hook_all_seq` In a few places we need a list of all hook names ever defined so we keep track of them in this sequence.
28 `\seq_new:N \g__hook_all_seq`
(End of definition for `\g__hook_all_seq`.)

`\l__hook_cur_hook_tl` Stores the name of the hook currently being sorted.
29 `\tl_new:N \l__hook_cur_hook_tl`
(End of definition for `\l__hook_cur_hook_tl`.)

`\l__hook_work_prop` A property list holding a copy of the `\g__hook_{hook}_code_prop` of the hook being sorted to work on, so that changes don't act destructively on the hook data structure.
30 `\prop_new:N \l__hook_work_prop`
(End of definition for `\l__hook_work_prop`.)

`\g__hook_used_prop` All hooks that receive code (for use in debugging display).
31 `\prop_new:N \g__hook_used_prop`
(End of definition for `\g__hook_used_prop`.)

`\g__hook_hook_curr_name_tl` Default label used for hook commands, and a stack to keep track of packages within
`\g__hook_name_stack_seq` packages.
32 `\tl_new:N \g__hook_hook_curr_name_tl`
33 `\seq_new:N \g__hook_name_stack_seq`
(End of definition for `\g__hook_hook_curr_name_tl` and `\g__hook_name_stack_seq`.)

`__hook_tmp:w` Temporary macro for generic usage.
34 `\cs_new_eq:NN __hook_tmp:w ?`
(End of definition for `__hook_tmp:w`.)

`\c__hook_empty_tl` An empty token list, and one containing nine parameters.

`\c__hook_nine_parameters_tl`

```

35 \tl_const:Nn \c__hook_empty_tl { }
36 \tl_const:Nn \c__hook_nine_parameters_tl { #1#2#3#4#5#6#7#8#9 }

```

(End of definition for `\c__hook_empty_tl` and `\c__hook_nine_parameters_tl`.)

`\tl_gremove_once:Nx` Some variants of `expl3` functions.

`\tl_show:x`

`\tl_log:x` *FMi: should probably be moved to `expl3`*

`\tl_set:Ne`

```

37 \cs_generate_variant:Nn \tl_gremove_once:Nn { Nx }
\cs_replacement_spec:c 38 \cs_generate_variant:Nn \tl_show:n { x }
\prop_put:Nne 39 \cs_generate_variant:Nn \tl_log:n { x }
\str_count:e 40 \cs_generate_variant:Nn \tl_set:Nn { Ne }
41 \cs_generate_variant:Nn \cs_replacement_spec:N { c }
42 \cs_generate_variant:Nn \prop_put:Nnn { Nne }
43 \cs_generate_variant:Nn \str_count:n { e }

```

(End of definition for `\tl_gremove_once:Nx` and others.)

`\s__hook_mark` Scan mark used for delimited arguments.

```

44 \scan_new:N \s__hook_mark

```

(End of definition for `\s__hook_mark`.)

`__hook_use_none_delimit_by_s_mark:w` Removes tokens until the next `\s__hook_mark`.

`__hook_use_i_delimit_by_s_mark:nw`

```

45 \cs_new:Npn \__hook_use_none_delimit_by_s_mark:w #1 \s__hook_mark { }
46 \cs_new:Npn \__hook_use_i_delimit_by_s_mark:nw #1 #2 \s__hook_mark {#1}

```

(End of definition for `__hook_use_none_delimit_by_s_mark:w` and `__hook_use_i_delimit_by_s_mark:nw`.)

`__hook_tl_set:cn` Private copies of a few `expl3` functions. `l3debug` will only add debugging to the public names, not to these copies, so we don't have to use `\debug_suspend:` and `\debug_resume:` everywhere.

Functions like `__hook_tl_set:Nn` have to be redefined, rather than copied because in `expl3` they use `__kernel_tl_(g)set:Nx`, which is also patched by `l3debug`.

```

47 \cs_new_protected:Npn \__hook_tl_set:cn #1#2
48 { \cs_set_nopar:cpx {#1} { \__kernel_exp_not:w {#2} } }

```

(End of definition for `__hook_tl_set:cn`.)

`__hook_tl_gset:Nn` Same as above.

`__hook_tl_gset:Nx`

`__hook_tl_gset:cn`

`__hook_tl_gset:co`

`__hook_tl_gset:cx`

```

49 \cs_new_protected:Npn \__hook_tl_gset:Nn #1#2
50 { \cs_gset_nopar:Npx #1 { \__kernel_exp_not:w {#2} } }
\cs_new_protected:Npn \__hook_tl_gset:Nx #1#2
51 { \cs_gset_nopar:Npx #1 {#2} }
52 { \cs_gset_nopar:Npx #1 {#2} }
53 \cs_generate_variant:Nn \__hook_tl_gset:Nn { c, co }
54 \cs_generate_variant:Nn \__hook_tl_gset:Nx { c }

```

(End of definition for `__hook_tl_gset:Nn`.)

`__hook_tl_gput_right:Nn` Same as above.

`__hook_tl_gput_right:Ne`

`__hook_tl_gput_right:cn`

```

55 \cs_new_protected:Npn \__hook_tl_gput_right:Nn #1#2
56 { \__hook_tl_gset:Nx #1 { \__kernel_exp_not:w \exp_after:wN { #1 #2 } } }
57 \cs_generate_variant:Nn \__hook_tl_gput_right:Nn { Ne, cn }

```

(End of definition for `__hook_tl_gput_right:Nn`.)

`__hook_tl_gput_left:Nn` Same as above.

```
58 \cs_new_protected:Npn \__hook_tl_gput_left:Nn #1#2
59   {
60     \__hook_tl_gset:Nx #1
61     { \__kernel_exp_not:w {#2} \__kernel_exp_not:w \exp_after:wN {#1} }
62   }
```

(End of definition for `__hook_tl_gput_left:Nn`.)

`__hook_tl_gset_eq:NN` Same as above.

```
63 \cs_new_eq:NN \__hook_tl_gset_eq:NN \tl_gset_eq:NN
```

(End of definition for `__hook_tl_gset_eq:NN`.)

`__hook_tl_gclear:N` Same as above.

```
\__hook_tl_gclear:c
64 \cs_new_protected:Npn \__hook_tl_gclear:N #1
65   { \__hook_tl_gset_eq:NN #1 \c_empty_tl }
66 \cs_generate_variant:Nn \__hook_tl_gclear:N { c }
```

(End of definition for `__hook_tl_gclear:N`.)

4.4 Providing new hooks

4.4.1 The data structures of a hook

`\g__hook_⟨hook⟩_code_prop` Hooks have a name (called `⟨hook⟩` in the description below) and for each hook we have
`__hook_⟨hook⟩` to provide a number of data structures. These are

`\g__hook_⟨hook⟩_reversed_tl` `\g__hook_⟨hook⟩_code_prop` A property list holding the code for the hook in separate
`\g__hook_⟨hook⟩_declared_tl` chunks. The keys are by default the package names that add code to the hook, but
`\g__hook_⟨hook⟩_parameter_tl` it is possible for packages to define other keys.
`__hook_next_⟨hook⟩`

`__hook_toplevel_⟨hook⟩` `\g__hook_⟨hook⟩_rule_⟨label1⟩|⟨label2⟩_tl` A token list holding the relation between `⟨label1⟩` and `⟨label2⟩` in the `⟨hook⟩`. The `⟨labels⟩` are lexically (reverse) sorted to ensure that two labels always point to the same token list. For global rules, the `⟨hook⟩` name is `??`.

`__hook_⟨hook⟩` The code that is actually executed when the hook is called in the document is stored in this token list. It is constructed from the code chunks applying the information. This token list is named like that so that in case of an error inside the hook, the reported token list in the error is shorter, and to make it simpler to normalize hook names in `__hook_make_name:n`.

`\g__hook_⟨hook⟩_reversed_tl` Some hooks are “reversed”. This token list stores a `-` for such hook so that it can be identified. The `-` character is used because `⟨reversed⟩`1 is `+1` for normal hooks and `-1` for reversed ones.

`\g__hook_⟨hook⟩_declared_tl` This token list serves as a marker for the hook being officially declared. Its existence is tested to raise an error in case another declaration is attempted.

`\c__hook_⟨hook⟩_parameter_tl` This token list stores the parameter text for a declared hook (its existence almost completely intersects the token list above), which is used for managing hooks with arguments.

`__hook_toplevel_⟨hook⟩` This token list stores the code inserted in the hook from the user's document, in the `top-level` label. This label is special, and doesn't participate in sorting. Instead, all code is appended to it and executed after (or before, if the hook is reversed) the normal hook code, but before the `next` code chunk.

`__hook_next_⟨hook⟩` Finally there is extra code (normally empty) that is used on the next invocation of the hook (and then deleted). This can be used to define some special behavior for a single occasion from within the document. This token list follows the same naming scheme than the main `__hook_⟨hook⟩` token list. It is called `__hook_next_⟨hook⟩` rather than `__hook_next_⟨hook⟩` because otherwise a hook whose name is `next_⟨hook⟩` would clash with the next code-token list of the hook called `⟨hook⟩`.

4.4.2 On the existence of hooks

A hook may be in different states of existence. Here we give an overview of the internal commands to set up hooks and explain how the different states are distinguished. The actual implementation then follows in subsequent sections.

One problem we have to solve is that we need to be able to add code to hooks (e.g., with `\AddToHook`) even if that code has not yet been declared. For example, one package needs to write into a hook of another package, but that package may not get loaded, or is loaded only later. Another problem is that most hooks, but not the generic hooks, require a declaration.

We therefore distinguish the following states for a hook, which are managed by four different tests: structure existence (`__hook_if_structure_exist:nTF`), creation (`__hook_if_usable:nTF`), declaration (`__hook_if_declared:nTF`) and disabled or not (`__hook_if_disabled:nTF`)

not existing Nothing is known about the hook so far. This state can be detected with `__hook_if_structure_exist:nTF` (which uses the false branch).

In this state the hook can be declared, disabled, rules can be defined or code could be added to it, but it is not possible to use the hook (with `\UseHook`).

basic data structure set up A hook is in this state when its basic data structure has been set up (using `__hook_init_structure:n`). The data structure setup happens automatically when commands such as `\AddToHook` are used and the hook is at that point in state “not existing”.

In this state the four tests give the following results:

```
\__hook_if_structure_exist:nTF returns true.  
    \__hook_if_usable:nTF returns false.  
    \__hook_if_declared:nTF returns false.  
    \__hook_if_disabled:nTF returns false.
```

The allowed actions are the same as in the “not existing” state.

declared A hook is in this state it is not disabled and was explicitly declared (e.g., with `\NewHook`). In this case the four tests give the following results:

```

\__hook_if_structure_exist:nTF returns true.
    \__hook_if_usable:nTF returns true.
    \__hook_if_declared:nTF returns true.
    \__hook_if_disabled:nTF returns false.

```

usable A hook is in this state if it is not disabled, was not explicitly declared but nevertheless is allowed to be used (with `\UseHook` or `\hook_use:n`). This state is only possible for generic hooks as they do not need to be declared. Therefore such hooks move directly from state “not existing” to “usable” the moment a declaration such as `\AddToHook` wants to add to the hook data structure. In this state the tests give the following results:

```

\__hook_if_structure_exist:nTF returns true.
    \__hook_if_usable:nTF returns true.
    \__hook_if_declared:nTF returns false.
    \__hook_if_disabled:nTF returns false.

```

disabled A generic hook in any state is moved to this state when `\DisableGenericHook` is used. This changes the tests to give the following results:

```

\__hook_if_structure_exist:nTF unchanged.
    \__hook_if_usable:nTF returns false.
    \__hook_if_declared:nTF returns true.
    \__hook_if_disabled:nTF returns true.

```

The structure test is unchanged (if the hook was unknown before it is `false`, otherwise `true`). The usable test returns `false` so that any `\UseHook` will bypass the hook from now on. The declared test returns `true` so that any further `\NewHook` generates an error and the disabled test returns `true` so that `\AddToHook` can return an error.

FMi: maybe it should do this only after begin document?

4.4.3 Setting hooks up

`\hook_new:n` The `\hook_new:n` declaration declares a new hook and expects the hook `<name>` as its argument, e.g., `begindocument`.

```

\hook_new_with_args:nn
\__hook_new:nn
67 <latexrelease> \IncludeInRelease{2023/06/01}{\hook_new_with_args:nn}
68 <latexrelease> {Hooks-with-args}
69 \cs_new_protected:Npn \hook_new:n #1
70 { \__hook_normalize_hook_args:Nn \__hook_new:nn {#1} { 0 } }
71 \cs_new_protected:Npn \hook_new_with_args:nn #1 #2
72 { \__hook_normalize_hook_args:Nn \__hook_new:nn {#1} {#2} }
73 \cs_new_protected:Npn \__hook_new:nn #1 #2
74 {

```

We check if the hook was already *explicitly* declared with `\hook_new:n`, and if it already exists we complain, otherwise set the “created” flag for the hook so that it errors next time `\hook_new:n` is used.

```

75   \__hook_if_declared:nTF {#1}
76     { \msg_error:nnn { hooks } { exists } {#1} }
77     {
78       \tl_new:c { g__hook_#1_declared_tl }
79       \cs_undefine:c { __hook~#1 }
80       \cs_undefine:c { c__hook_#1_parameter_tl }
81       \__hook_make_usable:nn {#1} {#2}

```

In case there is already code in a hook, but it’s undeclared, run `__hook_update_hook_code:n` to make it ready to be executed (see test `lthooks-034`).

```

82     \__hook_update_hook_code:n {#1}
83   }
84 }
85 <latexrelease>\EndIncludeInRelease
86 <latexrelease>\IncludeInRelease{2020/10/01}{\hook_new_with_args:nn}
87 <latexrelease>      {Hooks-with-args}
88 <latexrelease>\cs_gset_protected:Npn \hook_new:n #1
89 <latexrelease>  { \__hook_normalize_hook_args:Nn \__hook_new:n {#1} }
90 <latexrelease>\cs_undefine:N \__hook_new:nn
91 <latexrelease>\cs_gset_protected:Npn \__hook_new:n #1
92 <latexrelease>  {
93 <latexrelease>    \__hook_if_declared:nTF {#1}
94 <latexrelease>    { \msg_error:nnn { hooks } { exists } {#1} }
95 <latexrelease>    {
96 <latexrelease>      \tl_new:c { g__hook_#1_declared_tl }
97 <latexrelease>      \__hook_make_usable:n {#1}
98 <latexrelease>    }
99 <latexrelease>  }
100 <latexrelease>\cs_gset_protected:Npn \hook_new_with_args:nn #1 { }
101 <latexrelease>\EndIncludeInRelease

```

(End of definition for `\hook_new:n`, `\hook_new_with_args:nn`, and `__hook_new:nn`. These functions are documented on page 211.)

`__hook_make_usable:nn` This initializes all hook data structures for the hook but if used on its own doesn’t mark the hook as declared (as `\hook_new:n` does, so a later `\hook_new:n` on that hook will not result in an error. This command is internally used by `\hook_gput_code:nnn` when adding code to a generic hook.

```

102 <latexrelease>\IncludeInRelease{2023/06/01}{\__hook_make_usable:nn}
103 <latexrelease>      {Hooks-with-args}
104 \cs_new_protected:Npn \__hook_make_usable:nn #1 #2
105   {

```

Now we check if the hook’s data structure can be safely created without `expl3` raising errors, then we add the hook name to the list of all hooks and allocate the necessary data structures for the new hook, otherwise just do nothing.

```

106   \__hook_if_usable:nF {#1}
107   {
108     \seq_gput_right:Nn \g__hook_all_seq {#1}

```

Here we'll define the `\c__hook_⟨hook⟩_parameter_tl` to hold a run of parameters up to the number of arguments of the hook (`#2`).

```

109     \__kernel_cs_parm_from_arg_count:nnF
110     { \tl_const:cn { c__hook_#1_parameter_tl } } {#2}
111     {
112     \msg_error:nnnn { hooks } { too-many-args } {#1} {#2}
113     \tl_const:cx { c__hook_#1_parameter_tl }
114     { \exp_not:V \c__hook_nine_parameters_tl }
115     }

```

After that, use `__hook_normalise_cs_args:nn` to correct the number of parameters of the macros `__hook_toplevel_⟨hook⟩` and `__hook_next_⟨hook⟩`. We need to be able to add code with arguments to a hook without prior knowledge of the number of arguments of that hook, so `lthooks` assumes 9 until the hook is properly declared and the number of arguments is known. `__hook_normalise_cs_args:nn` does the normalization by using the `\c__hook_⟨hook⟩_parameter_tl` defined just above.

```

116     \__hook_normalise_cs_args:nn { _toplevel } {#1}
117     \__hook_normalise_cs_args:nn { _next } {#1}

```

This is only used by the actual code of the current hook, so declare it normally:

```

118     \__hook_code_gset:nn {#1} { }

```

Now ensure that the base data structure for the hook exists:

```

119     \__hook_init_structure:n {#1}

```

The call to `__hook_normalise_code_pool:n` will correct any improper reference to arguments that don't exist in the hook, raising a low-level `TEX` error and doubling the offending parameter tokens. It has to be done after `__hook_init_structure:n` because it operates on `\g__hook_⟨hook⟩_code_prop`.

```

120     \__hook_normalise_code_pool:n {#1}

```

The `\g__hook_⟨hook⟩_labels_clist` holds the sorted list of labels (once it got sorted). This is used only for debugging. These are defined conditionally, in case `__hook_make_usable:nn` is being used to redefine a hook.

```

121     \clist_if_exist:cF { g__hook_#1_labels_clist }
122     {
123     \clist_new:c { g__hook_#1_labels_clist }

```

Some hooks should reverse the default order of code chunks. To signal this we have a token list which is empty for normal hooks and contains a `-` for reversed hooks.

```

124     \tl_new:c { g__hook_#1_reversed_tl }
125     }

```

The above is all in L3 convention, but we also provide an interface to legacy `LATEX 2ε` hooks of the form `\@...hook`, e.g., `\@begindocumenthook`. There have been a few of them and they have been added to using `\g@addto@macro`. If there exists such a macro matching the name of the new hook, i.e., `\@⟨hook-name⟩hook` and it is not empty then we add its contents as a code chunk under the label `legacy`.

Warning: this support will vanish in future releases!

```

126     \__hook_include_legacy_code_chunk:n {#1}
127     }
128 }
129 \<latexrelease>\EndIncludeInRelease

```

```

130 <latexrelease> \IncludeInRelease{2020/10/01}{\__hook_make_usable:nn}
131 <latexrelease> {Hooks-with-args}
132 <latexrelease> \cs_undefine:N \__hook_make_usable:nn
133 <latexrelease> \cs_gset_protected:Npn \__hook_make_usable:n #1
134 <latexrelease> {
135 <latexrelease> \tl_if_exist:cF { __hook~#1 }
136 <latexrelease> {
137 <latexrelease> \seq_gput_right:Nn \g__hook_all_seq {#1}
138 <latexrelease> \tl_new:c { __hook~#1 }
139 <latexrelease> \__hook_init_structure:n {#1}
140 <latexrelease> \clist_new:c { g__hook_#1_labels_clist }
141 <latexrelease> \tl_new:c { g__hook_#1_reversed_tl }
142 <latexrelease> \__hook_include_legacy_code_chunk:n {#1}
143 <latexrelease> }
144 <latexrelease> }
145 <latexrelease> \EndIncludeInRelease

```

(End of definition for __hook_make_usable:nn.)

`__hook_init_structure:n` This function declares the basic data structures for a hook without explicit declaring the hook itself. This is needed to allow adding to undeclared hooks. Here it is unnecessary to check whether all variables exist, since all three are declared at the same time (either all of them exist, or none).

It creates the hook code pool (`\g__hook_<hook>_code_prop`) and the top-level and next token lists. A hook is initialized with `__hook_init_structure:n` the first time anything is added to it. Initializing a hook just with `__hook_init_structure:n` will not make it usable with `\hook_use:n`.

```

146 <latexrelease> \IncludeInRelease{2023/06/01}{\__hook_init_structure:n}
147 <latexrelease> {Hooks-with-args}
148 <latexrelease> \cs_new_protected:Npn \__hook_init_structure:n #1
149 <latexrelease> {
150 <latexrelease> \__hook_if_structure_exist:nF {#1}
151 <latexrelease> {
152 <latexrelease> \prop_new:c { g__hook_#1_code_prop }
153 <latexrelease> \__hook_toplevel_gset:nn {#1} { }
154 <latexrelease> \__hook_next_gset:nn {#1} { }
155 <latexrelease> }
156 <latexrelease> }
157 <latexrelease> \EndIncludeInRelease

```

```

158 <latexrelease> \IncludeInRelease{2020/10/01}{\__hook_init_structure:n}
159 <latexrelease> {Hooks-with-args}
160 <latexrelease> \cs_gset_protected:Npn \__hook_init_structure:n #1
161 <latexrelease> {
162 <latexrelease> \__hook_if_structure_exist:nF {#1}
163 <latexrelease> {
164 <latexrelease> \prop_new:c { g__hook_#1_code_prop }
165 <latexrelease> \tl_new:c { __hook_toplevel~#1 }
166 <latexrelease> \tl_new:c { __hook_next~#1 }
167 <latexrelease> }
168 <latexrelease> }
169 <latexrelease> \EndIncludeInRelease

```

(End of definition for __hook_init_structure:n.)

`\hook_new_reversed:n` Declare a new hook. The default ordering of code chunks is reversed, signaled by setting the token list to a minus sign.

`\hook_new_reversed_with_args:nn`

`__hook_new_reversed:nn`

```

170 <latexrelease> \IncludeInRelease{2023/06/01}{\hook_new_reversed_with_args:nn}
171 <latexrelease> {Hooks-with-args}
172 \cs_new_protected:Npn \hook_new_reversed:n #1
173 { \__hook_normalize_hook_args:Nn \__hook_new_reversed:nn {#1} { 0 } }
174 \cs_new_protected:Npn \hook_new_reversed_with_args:nn #1 #2
175 { \__hook_normalize_hook_args:Nn \__hook_new_reversed:nn {#1} {#2} }
176 \cs_new_protected:Npn \__hook_new_reversed:nn #1 #2
177 {
178   \__hook_if_declared:nTF {#1}
179   { \msg_error:nnn { hooks } { exists } {#1} }
180   {
181     \__hook_new:nn {#1} {#2}
182     \tl_gset:cn { g__hook_#1_reversed_tl } { - }
183   }
184 }
185 <latexrelease> \EndIncludeInRelease

186 <latexrelease> \IncludeInRelease{2020/10/01}{\hook_new_reversed_with_args:nn}
187 <latexrelease> {Hooks-with-args}
188 <latexrelease> \cs_gset_protected:Npn \hook_new_reversed:n #1
189 <latexrelease> { \__hook_normalize_hook_args:Nn \__hook_new_reversed:n {#1} }
190 <latexrelease> \cs_undefine:N \__hook_new_reversed:nn
191 <latexrelease> \cs_gset_protected:Npn \__hook_new_reversed:n #1
192 <latexrelease> {
193 <latexrelease>   \__hook_new:n {#1}
194 <latexrelease>   \tl_gset:cn { g__hook_#1_reversed_tl } { - }
195 <latexrelease> }
196 <latexrelease> \cs_undefine:N \__hook_new_reversed:nn
197 <latexrelease> \cs_gset_protected:Npn \hook_new_reversed_with_args:nn #1 #2 { }
198 <latexrelease> \EndIncludeInRelease

```

(End of definition for `\hook_new_reversed:n`, `\hook_new_reversed_with_args:nn`, and `__hook_new_reversed:nn`. These functions are documented on page 211.)

`\hook_new_pair:nn` A shorthand for declaring a normal and a (matching) reversed hook in one go.

`\hook_new_pair_with_args:nnn`

```

199 <latexrelease> \IncludeInRelease{2023/06/01}{\hook_new_pair_with_args:nnn}
200 <latexrelease> {Hooks-with-args}
201 \cs_new_protected:Npn \hook_new_pair:nn #1#2
202 { \__hook_normalize_hook_args:Nnn \__hook_new_pair:nnn {#1} {#2} { 0 } }
203 \cs_new_protected:Npn \hook_new_pair_with_args:nnn #1#2#3
204 { \__hook_normalize_hook_args:Nnn \__hook_new_pair:nnn {#1} {#2} {#3} }
205 \cs_new_protected:Npn \__hook_new_pair:nnn #1 #2 #3
206 {
207   \__hook_if_declared:nTF {#1}
208   { \msg_error:nnn { hooks } { exists } {#1} }
209   {
210     \__hook_if_declared:nTF {#2}
211     { \msg_error:nnn { hooks } { exists } {#2} }
212     {
213       \__hook_new:nn {#1} {#3}
214       \__hook_new_reversed:nn {#2} {#3}
215     }
216   }

```

```

217 }
218 <latexrelease>\EndIncludeInRelease
219 <latexrelease>\IncludeInRelease{2020/10/01}{\hook_new_pair_with_args:nnn}
220 <latexrelease> {Hooks-with-args}
221 <latexrelease>\cs_gset_protected:Npn \hook_new_pair:nn #1#2
222 <latexrelease> {
223 <latexrelease> \hook_new:n {#1}
224 <latexrelease> \hook_new_reversed:n {#2}
225 <latexrelease> }
226 <latexrelease>\cs_gset_protected:Npn \hook_new_pair_with_args:nnn #1#2#3
227 <latexrelease> { }
228 <latexrelease>\EndIncludeInRelease

```

(End of definition for `\hook_new_pair:nn` and `\hook_new_pair_with_args:nnn`. These functions are documented on page 211.)

`__hook_include_legacy_code_chunk:n`

The L^AT_EX legacy concept for hooks uses with hooks the following naming scheme in the code: `\@...hook`.

If this macro is not empty we add it under the label `legacy` to the current hook and then empty it globally. This way packages or classes directly manipulating commands such as `\@begindocumenthook` still get their hook data added.

Warning: this support will vanish in future releases!

```

229 <latexrelease>\IncludeInRelease{2023/06/01}{\__hook_include_legacy_code_chunk:n}
230 <latexrelease> {Hooks-with-args}
231 \cs_new_protected:Npn \__hook_include_legacy_code_chunk:n #1
232 {

```

If the macro doesn't exist (which is the usual case) then nothing needs to be done.

```

233 \tl_if_exist:cT { @#1hook }
234 {

```

Of course if the legacy hook exists but is empty, there is no need to add anything under `legacy` the legacy label.

```

235 \tl_if_empty:cF { @#1hook }
236 {

```

Here we set `__hook_replacing_args_false:` because no legacy code will reference hook arguments.

```

237 \__hook_replacing_args_false:
238 \use:e
239 {
240 \__hook_hook_gput_code_do:nnn {#1} { legacy }
241 { \exp_not:v { @#1hook } }
242 }
243 \__hook_replacing_args_reset:

```

Once added to the hook, we need to clear it otherwise it might get added again later if the hook data gets updated.

```

244 \__hook_tl_gclear:c { @#1hook }
245 }
246 }
247 }

```

```

248 <latexrelease>\EndIncludeInRelease
249 <latexrelease>\IncludeInRelease{2020/10/01}{\__hook_include_legacy_code_chunk:n}

```

```

250 <latexrelease> {Hooks-with-args}
251 <latexrelease> \cs_gset_protected:Npn \__hook_include_legacy_code_chunk:n #1
252 <latexrelease> {
253 <latexrelease> \tl_if_exist:cT { @#1hook }
254 <latexrelease> {
255 <latexrelease> \tl_if_empty:cF { @#1hook }
256 <latexrelease> {
257 <latexrelease> \exp_args:Nnnv \__hook_hook_gput_code_do:nnn
258 <latexrelease> {#1} { legacy } { @#1hook }
259 <latexrelease> \__hook_tl_gclear:c { @#1hook }
260 <latexrelease> }
261 <latexrelease> }
262 <latexrelease> }
263 <latexrelease> \EndIncludeInRelease

```

(End of definition for __hook_include_legacy_code_chunk:n.)

4.4.4 Disabling and providing hooks

`\hook_disable_generic:n` Disables a hook by creating its `\g__hook_⟨hook⟩_declared_tl` so that the hook errors when used with `\hook_new:n`, then it undefines `__hook_⟨hook⟩` so that it may not be executed.

`__hook_disable:n`
`__hook_if_disabled:p:n`
`__hook_if_disabled:nTF`

This does not clear any code that may be already stored in the hook's structure, but doesn't allow adding more code. `__hook_if_disabled:nTF` uses that specific combination to check if the hook is disabled.

```

264 <latexrelease> \IncludeInRelease{2021/06/01}{\hook_disable_generic:n}
265 <latexrelease> {Disable-hooks}

266 \cs_new_protected:Npn \hook_disable_generic:n #1
267 { \__hook_normalize_hook_args:Nn \__hook_disable:n {#1} }
268 \cs_new_protected:Npn \__hook_disable:n #1
269 {
270 \tl_gclear_new:c { g__hook_#1_declared_tl }
271 \cs_undefine:c { __hook~#1 }
272 }
273 \prg_new_conditional:Npnn \__hook_if_disabled:n #1 { p, T, F, TF }
274 {
275 \bool_lazy_and:nnTF
276 { \tl_if_exist_p:c { g__hook_#1_declared_tl } }
277 { ! \cs_if_exist_p:c { __hook~#1 } }
278 { \prg_return_true: }
279 { \prg_return_false: }
280 }
281 <latexrelease> \EndIncludeInRelease

282 <latexrelease> \IncludeInRelease{2020/10/01}{\hook_disable_generic:n}
283 <latexrelease> {Disable-hooks}
284 <latexrelease>
285 <latexrelease> \cs_new_protected:Npn \hook_disable_generic:n #1 {}
286 <latexrelease>
287 <latexrelease> \EndIncludeInRelease

```

(End of definition for `\hook_disable_generic:n`, `__hook_disable:n`, and `__hook_if_disabled:nTF`. This function is documented on page 212.)

`\hook_activate_generic:n` The `\hook_activate_generic:n` declaration declares a new hook if it wasn't declared already, in which case it only checks that the already existing hook is not a reversed hook.

```

288 <latexrelease>\IncludeInRelease{2023/06/01}{\hook_activate_generic:n}
289 <latexrelease>          {Providing-hooks}

290 \cs_new_protected:Npn \hook_activate_generic:n #1
291   { \__hook_normalize_hook_args:Nn \__hook_activate_generic:nn {#1} { } }

292 \cs_new_protected:Npn \__hook_activate_generic:nn #1 #2
293   {

```

If the hook to be activated was disabled we warn (for now — this may change).

```

294   \__hook_if_disabled:nTF {#1}
295     { \msg_warning:nnn { hooks } { activate-disabled } {#1} }

```

Otherwise we check if the hook is not declared, and if it isn't, figure out if it's reversed or not, then declare it accordingly.

```

296     {
297       \__hook_if_declared:nF {#1}
298       {
299         \tl_new:c { g__hook_#1_declared_tl }
300         \__hook_make_usable:nn {#1} { 0 }
301         \tl_gset:cx { g__hook_#1_reversed_tl }
302         { \__hook_if_generic_reversed:nT {#1} { - } }

```

Reflect that we have activated the generic hook and set its execution code.

```

303         \__hook_update_hook_code:n {#1}
304       }
305     }
306   }

```

```

307 <latexrelease>\EndIncludeInRelease

```

```

308 <latexrelease>\IncludeInRelease{2021/06/01}{\hook_activate_generic:n}
309 <latexrelease>          {Providing-hooks}
310 <latexrelease>\cs_gset_protected:Npn \__hook_activate_generic:nn #1 #2
311 <latexrelease>  {
312 <latexrelease>    \__hook_if_disabled:nTF {#1}
313 <latexrelease>      { \msg_warning:nnn { hooks } { activate-disabled } {#1} }
314 <latexrelease>      {
315 <latexrelease>        \__hook_if_declared:nF {#1}
316 <latexrelease>        {
317 <latexrelease>          \tl_new:c { g__hook_#1_declared_tl }
318 <latexrelease>          \__hook_make_usable:n {#1}
319 <latexrelease>          \tl_gset:cx { g__hook_#1_reversed_tl }
320 <latexrelease>          { \__hook_if_generic_reversed:nT {#1} { - } }
321 <latexrelease>          \__hook_update_hook_code:n {#1}
322 <latexrelease>        }
323 <latexrelease>      }
324 <latexrelease>    }
325 <latexrelease>\EndIncludeInRelease

```

```

326 <latexrelease>\IncludeInRelease{2020/10/01}{\hook_activate_generic:n}
327 <latexrelease>          {Providing-hooks}
328 <latexrelease>\cs_gset_protected:Npn \hook_activate_generic:n #1 { }
329 <latexrelease>\EndIncludeInRelease

```

(End of definition for `\hook_activate_generic:n` and `_hook_activate_generic:n`. This function is documented on page 212.)

4.5 Parsing a label

`_hook_parse_label_default:nN` This macro checks if a label was given (not `\c_novalue_tl`), and if so, tries to parse the label looking for a leading `.` to replace by `_hook_currname_or_default:.` #2 is a boolean representing if #1 is a label name.

```

330 \cs_new:Npn \_hook_parse_label_default:nN #1#2
331   {
332     \tl_if_novalue:nTF {#1}
333       { \_hook_currname_or_default: }
334       { \tl_trim_spaces_apply:nN {#1} \_hook_parse_dot_label:nN #2 }
335   }

```

(End of definition for `_hook_parse_label_default:nN`.)

`_hook_parse_dot_label:nN` Start by checking if the label is empty, which raises an error, and uses the fallback value.
`_hook_parse_dot_label:w` If not, split the label at a `.`, if any, and check if no tokens are before the `.`, or if the
`_hook_parse_dot_label_cleanup:w` only character is a `..`. If these requirements are fulfilled, the leading `.` is replaced with
`_hook_parse_dot_label_aux:w` `_hook_currname_or_default:.` Otherwise the label is returned unchanged. #2 is a boolean representing if #1 is a label name.

```

336 \cs_new:Npn \_hook_parse_dot_label:nN #1#2
337   {
338     \tl_if_empty:nTF {#1}
339       {
340         \bool_if:NTF #2
341           { \msg_expandable_error:nn { hooks } { empty-label } }
342           { \msg_expandable_error:nn { hooks } { empty-hook } }
343         \_hook_currname_or_default:
344       }
345       {
346         \str_if_eq:nnTF {#1} { . }
347           { \_hook_currname_or_default: }
348           { \_hook_parse_dot_label:w #1 ./ \s_hook_mark }
349       }
350   }
351 \cs_new:Npn \_hook_parse_dot_label:w #1 ./ #2 \s_hook_mark
352   {
353     \tl_if_empty:nTF {#1}
354       { \_hook_parse_dot_label_aux:w #2 \s_hook_mark }
355       {
356         \tl_if_empty:nTF {#2}
357           { \_hook_make_name:n {#1} }
358           { \_hook_parse_dot_label_cleanup:w #1 ./ #2 \s_hook_mark }
359       }
360   }
361 \cs_new:Npn \_hook_parse_dot_label_cleanup:w #1 ./ \s_hook_mark {#1}
362 \cs_new:Npn \_hook_parse_dot_label_aux:w #1 ./ \s_hook_mark
363   { \_hook_currname_or_default: / \_hook_make_name:n {#1} }

```

(End of definition for `_hook_parse_dot_label:nN` and others.)

`__hook_currname_or_default:` This uses `\g__hook_hook_curr_name_tl` if it is set, otherwise it tries `\@currname`. If neither is set, it raises an error and uses the fallback value `label-missing`.

```

364 \cs_new:Npn \__hook_currname_or_default:
365   {
366     \tl_if_empty:NTF \g__hook_hook_curr_name_tl
367     {
368       \tl_if_empty:NTF \@currname
369       {
370         \msg_expandable_error:nnn { latex2e } { should-not-happen }
371         { Empty~default~label. }
372         \__hook_make_name:n { label-missing }
373       }
374       { \@currname }
375     }
376     { \g__hook_hook_curr_name_tl }
377   }

```

(End of definition for `__hook_currname_or_default:.`)

`__hook_make_name:n` This provides a standard sanitization of a hook’s name. It uses `\cs:w` to build a control sequence out of the hook name, then uses `\cs_to_str:N` to get the string representation of that, without the escape character. `\cs:w`-based expansion is used instead of `e`-based because Unicode characters don’t behave well inside `\expanded`. The macro adds the `__hook_` prefix to the hook name to reuse the hook’s code token list to build the csname and avoid leaving “public” control sequences defined (as `\relax`) in TeX’s memory.

`__hook_make_name:w`

```

378 \cs_new:Npn \__hook_make_name:n #1
379   {
380     \exp_after:wN \exp_after:wN \exp_after:wN \__hook_make_name:w
381     \exp_after:wN \token_to_str:N \cs:w __hook~ #1 \cs_end:
382   }
383 \exp_last_unbraced:NNNNo
384 \cs_new:Npn \__hook_make_name:w #1 \tl_to_str:n { __hook~ } { }

```

(End of definition for `__hook_make_name:n` and `__hook_make_name:w`.)

`__hook_normalize_hook_args:Nn` This is the standard route for normalizing hook and label arguments. The main macro does the entire operation within a group so that csnames made by `__hook_make_name:n` are wiped off before continuing. This means that this function cannot be used for `\hook_use:n!`

`__hook_normalize_hook_args:Nnn`

`__hook_normalize_hook_rule_args:Nnnnn`

`__hook_normalize_hook_args_aux:Nn`

```

385 \cs_new_protected:Npn \__hook_normalize_hook_args_aux:Nn #1 #2
386   {
387     \group_begin:
388     \use:e
389     {
390       \group_end:
391       \exp_not:N #1 #2
392     }
393   }
394 \cs_new_protected:Npn \__hook_normalize_hook_args:Nn #1 #2
395   {
396     \__hook_normalize_hook_args_aux:Nn #1
397     { { \__hook_parse_label_default:nN {#2} \c_false_bool } }
398   }

```

```

399 \cs_new_protected:Npn \__hook_normalize_hook_args:Nnn #1 #2 #3
400 {
401   \__hook_normalize_hook_args_aux:Nn #1
402   {
403     { \__hook_parse_label_default:nN {#2} \c_false_bool }
404     { \__hook_parse_label_default:nN {#3} \c_true_bool }
405   }
406 }
407 \cs_new_protected:Npn \__hook_normalize_hook_rule_args:Nnnnn #1 #2 #3 #4 #5
408 {
409   \__hook_normalize_hook_args_aux:Nn #1
410   {
411     { \__hook_parse_label_default:nN {#2} \c_false_bool }
412     { \__hook_parse_label_default:nN {#3} \c_true_bool }
413     { \tl_trim_spaces:n {#4} }
414     { \__hook_parse_label_default:nN {#5} \c_true_bool }
415   }
416 }

```

(End of definition for `__hook_normalize_hook_args:Nn` and others.)

`__hook_curr_name_push:n` The token list `\g__hook_hook_curr_name_tl` stores the name of the current package/file to be used as the default label in hooks. Providing a consistent interface is tricky because packages can be loaded within packages, and some packages may not use `\SetDefaultHookLabel` to change the default label (in which case `\@currname` is used).

`__hook_curr_name_push_aux:n`
`__hook_curr_name_pop:`
`__hook_end_document_label_check:`

To pull that one off, we keep a stack that contains the default label for each level of input. The bottom of the stack contains the default label for the top-level (this stack should never go empty). If we're building the format, set the default label to be top-level:

```
417 \tl_gset:Nn \g__hook_hook_curr_name_tl { top-level }
```

Then, in case we're in `latexrelease` we push something on the stack to support roll forward. But in some rare cases, `latexrelease` may be loaded inside another package (notably `platexrelease`), so we'll first push the top-level entry:

```
418 <latexrelease> \seq_if_empty:NT \g__hook_name_stack_seq
419 <latexrelease> { \seq_gput_right:Nn \g__hook_name_stack_seq { top-level } }
```

then we dissect the `\@currnamestack`, adding `\@currname` to the stack:

```
420 <latexrelease> \cs_set_protected:Npn \__hook_tmp:w #1 #2 #3
421 <latexrelease> {
422 <latexrelease>   \quark_if_recursion_tail_stop:n {#1}
423 <latexrelease>   \seq_gput_right:Nn \g__hook_name_stack_seq {#1}
424 <latexrelease>   \__hook_tmp:w
425 <latexrelease> }
426 <latexrelease> \exp_after:wN \__hook_tmp:w \@currnamestack
427 <latexrelease> \q_recursion_tail \q_recursion_tail
428 <latexrelease> \q_recursion_tail \q_recursion_stop
```

and finally set the default label to be the `\@currname`:

```
429 <latexrelease> \tl_gset:Nx \g__hook_hook_curr_name_tl { \@currname }
430 <latexrelease> \seq_gpop_right:NN \g__hook_name_stack_seq \l__hook_tmpa_tl
```

Two commands keep track of the stack: when a file is input, `__hook_curr_name_push:n` pushes the current default label onto the stack and sets the new default label (all in one go):

```

431 \cs_new_protected:Npn \__hook_curr_name_push:n #1
432 { \exp_args:Nx \__hook_curr_name_push_aux:n { \__hook_make_name:n {#1} } }
433 \cs_new_protected:Npn \__hook_curr_name_push_aux:n #1
434 {
435   \tl_if_blank:nTF {#1}
436     { \msg_error:nn { hooks } { no-default-label } }
437     {
438       \str_if_eq:nnTF {#1} { top-level }
439       {
440         \msg_error:nnnnn { hooks } { set-top-level }
441         { to } { PushDefaultHookLabel } {#1}
442       }
443       {
444         \seq_gpush:NV \g__hook_name_stack_seq \g__hook_hook_curr_name_tl
445         \tl_gset:Nn \g__hook_hook_curr_name_tl {#1}
446       }
447     }
448 }

```

and when an input is over, the topmost item of the stack is popped, since that label will not be used again, and `\g__hook_hook_curr_name_tl` is updated to equal the now topmost item of the stack:

```

449 \cs_new_protected:Npn \__hook_curr_name_pop:
450 {
451   \seq_gpop:NNTF \g__hook_name_stack_seq \l__hook_return_tl
452   { \tl_gset_eq:NN \g__hook_hook_curr_name_tl \l__hook_return_tl }
453   { \msg_error:nn { hooks } { extra-pop-label } }
454 }

```

At the end of the document we want to check if there was no `__hook_curr_name_push:n` without a matching `__hook_curr_name_pop:` (not a critical error, but it might indicate that something else is not quite right):

```

455 \tl_gput_right:Nn \@kernel@after@enddocument@afterlastpage
456 { \__hook_end_document_label_check: }
457 \cs_new_protected:Npn \__hook_end_document_label_check:
458 {
459   \seq_gpop:NNT \g__hook_name_stack_seq \l__hook_return_tl
460   {
461     \msg_error:nxx { hooks } { missing-pop-label }
462     { \g__hook_hook_curr_name_tl }
463     \tl_gset_eq:NN \g__hook_hook_curr_name_tl \l__hook_return_tl
464     \__hook_end_document_label_check:
465   }
466 }

```

The token list `\g__hook_hook_curr_name_tl` is but a mirror of the top of the stack.

Now define a wrapper that replaces the top of the stack with the argument, and updates `\g__hook_hook_curr_name_tl` accordingly.

```

467 \cs_new_protected:Npn \__hook_set_default_hook_label:n #1
468 {
469   \seq_if_empty:NNTF \g__hook_name_stack_seq
470   {
471     \msg_error:nnnnn { hooks } { set-top-level }
472     { for } { SetDefaultHookLabel } {#1}

```

`__hook_set_default_hook_label:n`
`__hook_set_default_hook_label:n`

```

473     }
474     { \exp_args:Nx
475       \__hook_set_default_label:n { \__hook_make_name:n {#1} } }
476   }
477 \cs_new_protected:Npn \__hook_set_default_label:n #1
478 {
479   \str_if_eq:nnTF {#1} { top-level }
480   {
481     \msg_error:nnnn { hooks } { set-top-level }
482     { to } { SetDefaultHookLabel } {#1}
483   }
484   { \tl_gset:Nn \g__hook_hook_curr_name_tl {#1} }
485 }

```

(End of definition for `__hook_curr_name_push:n` and others.)

4.6 Adding or removing hook code

`\hook_gput_code:nnn` With `\hook_gput_code:nnn{<hook>}{<label>}{<code>}` a chunk of `<code>` is added to an existing `<hook>` labeled with `<label>`.

`\hook_gput_code_with_args:nnn`

`__hook_gput_code:nnn`

`__hook_gput_code_store:nnn`

`__hook_hook_gput_code_do:nnn`

`__hook_prop_gput_labeled_cleanup:nnn`

`__hook_prop_gput_labeled_do:Nnnn`

```

486 <latexrelease> \IncludeInRelease{2023/06/01}{\hook_gput_code:nnn}
487 <latexrelease>           {Hooks-with-args}
488 \cs_new_protected:Npn \hook_gput_code:nnn #1 #2 #3
489 {
490   \__hook_replacing_args_false:
491   \__hook_normalize_hook_args:Nnn \__hook_gput_code:nnn {#1} {#2} {#3}
492   \__hook_replacing_args_reset:
493 }
494 \cs_new_protected:Npn \hook_gput_code_with_args:nnn #1 #2 #3
495 {
496   \__hook_replacing_args_true:
497   \__hook_normalize_hook_args:Nnn \__hook_gput_code:nnn {#1} {#2} {#3}
498   \__hook_replacing_args_reset:
499 }

```

If `\AddToHookWithArguments` was used, do some sanity checking, and if it's not possible to use arguments at this point, fall back to regular `\AddToHook` by using `__hook_replacing_args_false:`.

```

500 \cs_new_protected:Npn \__hook_gput_code:nnn #1 #2 #3
501 {
502   \__hook_chk_args_allowed:nn {#1} { AddToHook }

```

Then check if the code should be executed immediately, rather than stored:

```

503   \__hook_if_execute_immediately:nTF {#1}
504   {

```

`\AddToHookWithArguments` can't be used on one-time hooks (that were already used).

```

505     \__hook_if_replacing_args:TF
506     {
507       \msg_error:nnnn { hooks } { one-time-args }
508       {#1} { AddToHook }
509     }
510     { }
511   \use:n
512 }

```

```

513     { \_hook_gput_code_store:nnn {#1} {#2} }
514       {#3}
515   }

```

```

516 \cs_new_protected:Npn \_hook_gput_code_store:nnn #1 #2 #3
517   {

```

Then check if the hook is usable.

```

518   \_hook_if_usable:nTF {#1}

```

If so we simply add (or append) the new code to the property list holding different chunks for the hook. At `\begin{document}` this is then sorted into a token list for fast execution.

```

519     {
520       \_hook_hook_gput_code_do:nnn {#1} {#2} {#3}

```

However, if there is an update within the document we need to alter this execution code which is done by `_hook_update_hook_code:n`. In the preamble this does nothing.

```

521       \_hook_update_hook_code:n {#1}
522     }

```

If the hook is not usable, before giving up, check if it's not disabled and otherwise try to declare it as a generic hook, if its name matches one of the valid patterns.

```

523     {
524       \_hook_if_disabled:nTF {#1}
525       { \msg_error:nnn { hooks } { hook-disabled } {#1} }
526       { \_hook_try_declaring_generic_hook:nnn {#1} {#2} {#3} }
527     }
528   }

```

This macro will unconditionally add a chunk of code to the given hook.

```

529 \cs_new_protected:Npn \_hook_hook_gput_code_do:nnn #1 #2 #3
530   {

```

However, first some debugging info if debugging is enabled:

```

531   \_hook_debug:n{\iow_term:x{[lthooks]~ Add~ to~
532     \_hook_if_usable:nF {#1} { undeclared~ }
533     hook~ '#1'~ (#2) \on@line
534     ^^J [lthooks] \@spaces
535     <~ \tl_to_str:n{#3}} }

```

Then try to get the code chunk labeled #2 from the hook. If there's code already there, then append #3 to that, otherwise just put #3. If the current label is `top-level`, the code is added to a dedicated token list `_hook_toplevel_<hook>` that goes at the end of the hook (or at the beginning, for a reversed hook), just before `_hook_next_<hook>`.

```

536   \str_if_eq:nnTF {#2} { top-level }
537   {
538     \str_if_eq:eeTF { top-level } { \_hook_currname_or_default: }
539   {

```

If the hook's basic structure does not exist, we need to declare it with `_hook_init_structure:n`.

```

540       \_hook_init_structure:n {#1}

```

Then append to the `_toplevel` container for the hook.

```

541     \__hook_cs_gput_right:nnn { _toplevel } {#1} {#3}
542   }
543   { \msg_error:nnn { hooks } { misused-top-level } {#1} }
544 }
545 {

```

When adding to the code pool, we have to double hashes if `\AddToHook` was used (`replacing_args` is false), so that later it is turned into a single parameter token, rather than a parameter to the hook macro.

```

546     \exp_args:Nx \__hook_prop_gput_labeled_cleanup:nnn
547   {
548     \__hook_if_replacing_args:TF
549     { \exp_not:n }
550     { \__hook_double_hashes:n }
551     {#3}
552   }
553   {#1} {#2}
554 }
555 }

```

Adds code to a hook's code pool.

```

556 \cs_new_protected:Npn \__hook_prop_gput_labeled_cleanup:nnn #1 #2 #3
557 {
558   \tl_set:Nn \l__hook_return_tl {#1}
559   \__hook_if_replacing_args:TF
560   {
561     \__hook_if_usable:nT {#2}
562     {
563       \__hook_set_normalise_fn:nn {#2}
564       { Invalid~code~added~\msg_line_context: }
565       \__hook_normalise_fn:nn {#3} {#1}
566       \prop_get:NnN \l__hook_work_prop {#3} \l__hook_return_tl
567     }
568   }
569   { }
570   \exp_args:NcV \__hook_prop_gput_labeled_do:Nnn
571   { g__hook_#2_code_prop } \l__hook_return_tl {#3}
572 }
573 \cs_new_protected:Npn \__hook_prop_gput_labeled_do:Nnn #1 #2 #3
574 {
575   \prop_get:NnNTF #1 {#3} \l__hook_return_tl
576   { \prop_gput:Nno #1 {#3} { \l__hook_return_tl #2 } }
577   { \prop_gput:Nnn #1 {#3} {#2} }
578 }
579 <latexrelease> \EndIncludeInRelease
580 <latexrelease> \IncludeInRelease{2020/10/01}{\hook_gput_code:nnn}
581 <latexrelease> {Providing-hooks}
582 <latexrelease> \cs_gset_protected:Npn \hook_gput_code:nnn #1 #2
583 <latexrelease> { \__hook_normalize_hook_args:Nnn
584 <latexrelease> \__hook_gput_code:nnn {#1} {#2} }
585 <latexrelease> \cs_gset_protected:Npn \__hook_gput_code:nnn #1 #2 #3
586 <latexrelease> {
587 <latexrelease> \__hook_if_execute_immediately:nTF {#1}

```



```

588 <latexrelease>     {#3}
589 <latexrelease>     {
590 <latexrelease>         \__hook_if_usable:nTF {#1}
591 <latexrelease>         {
592 <latexrelease>             \__hook_hook_gput_code_do:nnn {#1} {#2} {#3}
593 <latexrelease>             \__hook_update_hook_code:n {#1}
594 <latexrelease>         }
595 <latexrelease>         {
596 <latexrelease>             \__hook_if_disabled:nTF {#1}
597 <latexrelease>             { \msg_error:nnn { hooks } { hook-disabled } {#1} }
598 <latexrelease>             { \__hook_try_declaring_generic_hook:nnn
599 <latexrelease>                 {#1} {#2} {#3} }
600 <latexrelease>         }
601 <latexrelease>     }
602 <latexrelease> }
603 <latexrelease> \cs_gset_protected:Npn \__hook_hook_gput_code_do:nnn #1 #2 #3
604 <latexrelease> {
605 <latexrelease>     \__hook_debug:n{\iow_term:x{****~ Add~ to~
606 <latexrelease>         \__hook_if_usable:nF {#1} { undeclared~
607 <latexrelease>         hook~ #1~ (#2)
608 <latexrelease>         \on@line\space <-- \tl_to_str:n{#3}} }
609 <latexrelease>     \str_if_eq:nnTF {#2} { top-level }
610 <latexrelease>     {
611 <latexrelease>         \str_if_eq:eeTF { top-level }
612 <latexrelease>         { \__hook_currname_or_default: }
613 <latexrelease>     {
614 <latexrelease>         \__hook_init_structure:n {#1}
615 <latexrelease>         \__hook_tl_gput_right:cn { \__hook_toplevel~#1 } {#3}
616 <latexrelease>     }
617 <latexrelease>     { \msg_error:nnn { hooks } { misused-top-level } {#1} }
618 <latexrelease> }
619 <latexrelease> {
620 <latexrelease>     \prop_get:cnNTF
621 <latexrelease>     { g__hook_#1_code_prop } {#2} \l__hook_return_tl
622 <latexrelease>     {
623 <latexrelease>         \prop_gput:cno { g__hook_#1_code_prop } {#2}
624 <latexrelease>         { \l__hook_return_tl #3 }
625 <latexrelease>     }
626 <latexrelease>     { \prop_gput:cnn { g__hook_#1_code_prop } {#2} {#3} }
627 <latexrelease> }
628 <latexrelease> }
629 <latexrelease> \cs_gset_protected:Npn \hook_gput_code_with_args:nnn #1#2#3 { }
630 <latexrelease> \EndIncludeInRelease

```

(End of definition for `\hook_gput_code:nnn` and others. These functions are documented on page 213.)

`__hook_chk_args_allowed:nn` This macro checks if it is possible to add code with references to a hook's arguments for hook #1. It only does something if the function being run is `replacing_args`. This macro will error if the hook is declared and takes no arguments, then it will set `__hook_replacing_args_false:` so that the macro which called it will add the code normally.

```

631 <latexrelease> \IncludeInRelease{2023/06/01}{\__hook_chk_args_allowed:nn}
632 <latexrelease>     {Hooks-with-args}
633 \cs_new_protected:Npn \__hook_chk_args_allowed:nn #1 #2

```

```

634 {
635   \__hook_if_replacing_args:TF
636   {
637     \__hook_if_declared:nT {#1}
638     { \tl_if_empty:cT { c__hook_#1_parameter_tl } { \use_ii:nn } }
639     \use_none:n
640     {
641       \msg_error:nnnn { hooks } { without-args } {#1} {#2}
642       \__hook_replacing_args_false:
643     }
644   }
645   { }
646 }
647 <latexrelease>\EndIncludeInRelease
648 <latexrelease>\IncludeInRelease{2020/10/01}{\__hook_chk_args_allowed:nn}
649 <latexrelease>           {Hooks-with-args}
650 <latexrelease>\cs_undefine:N \__hook_chk_args_allowed:nn
651 <latexrelease>\EndIncludeInRelease

```

(End of definition for __hook_chk_args_allowed:nn.)

__hook_gput_undeclared_hook:nnn Often it may happen that a package *A* defines a hook *foo*, but package *B*, that adds code to that hook, is loaded before *A*. In such case we need to add code to the hook before it is declared. An implicitly declared hook doesn't have arguments (in principle), so use `\c_false_bool` here.

```

652 \cs_new_protected:Npn \__hook_gput_undeclared_hook:nnn #1 #2 #3
653 {
654   \__hook_init_structure:n {#1}
655   \__hook_hook_gput_code_do:nnn {#1} {#2} {#3}
656 }

```

(End of definition for __hook_gput_undeclared_hook:nnn.)

__hook_try_declaring_generic_hook:nnn These entry-level macros just pass the arguments along to the common `__hook_try_declaring_generic_hook:nNNnn` with the right functions to execute when some action is to be taken.

The wrapper `__hook_try_declaring_generic_hook:nnn` then defers `\hook_gput_code:nnn` if the generic hook was declared, or to `__hook_gput_undeclared_hook:nnn` otherwise (the hook was tested for existence before, so at this point if it isn't generic, it doesn't exist).

The wrapper `__hook_try_declaring_generic_next_hook:nn` for next-execution hooks does the same: it defers the code to `\hook_gput_next_code:nn` if the generic hook was declared, or to `__hook_gput_next_do:nn` otherwise.

```

657 <latexrelease>\IncludeInRelease{2023/06/01}
658 <latexrelease>           {\__hook_try_declaring_generic_hook:nnn}
659 <latexrelease>           {Hooks-with-args}
660 \cs_new_protected:Npn \__hook_try_declaring_generic_hook:nnn #1
661 {
662   \__hook_try_declaring_generic_hook:wnTF #1 / / / \scan_stop: {#1}
663   \__hook_gput_code:nnn
664   \__hook_gput_undeclared_hook:nnn
665   {#1}
666 }

```

```

667 \cs_new_protected:Npn \__hook_try_declaring_generic_next_hook:nn #1
668 {
669   \__hook_try_declaring_generic_hook:wTF #1 / / \scan_stop: {#1}
670   \__hook_gput_next_code:nn
671   \__hook_gput_next_do:nn
672   {#1}
673 }
674 \<latexrelease>\EndIncludeInRelease
675 \<latexrelease>\IncludeInRelease{2021/11/15}
676 \<latexrelease>      {\__hook_try_declaring_generic_hook:nnn}
677 \<latexrelease>      {Standardize-generic-hook-names}
678 \<latexrelease>\cs_gset_protected:Npn \__hook_try_declaring_generic_hook:nnn #1
679 \<latexrelease> {
680 \<latexrelease>   \__hook_try_declaring_generic_hook:wTF #1 / / \scan_stop:
681 \<latexrelease>     {#1}
682 \<latexrelease>   \hook_gput_code:nnn
683 \<latexrelease>   \__hook_gput_undeclared_hook:nnn
684 \<latexrelease>     {#1}
685 \<latexrelease> }
686 \<latexrelease>\cs_gset_protected:Npn
687 \<latexrelease> \__hook_try_declaring_generic_next_hook:nn #1
688 \<latexrelease> {
689 \<latexrelease>   \__hook_try_declaring_generic_hook:wTF #1 / / \scan_stop:
690 \<latexrelease>     {#1}
691 \<latexrelease>   \hook_gput_next_code:nn
692 \<latexrelease>   \__hook_gput_next_do:nn
693 \<latexrelease>     {#1}
694 \<latexrelease> }
695 \<latexrelease>\EndIncludeInRelease
696 \<latexrelease>\IncludeInRelease{2020/10/01}
697 \<latexrelease>      {\__hook_try_declaring_generic_hook:nnn}
698 \<latexrelease>      {Standardize-generic-hook-names}
699 \<latexrelease>\cs_new_protected:Npn
700 \<latexrelease> \__hook_try_declaring_generic_hook:nnn #1
701 \<latexrelease> {
702 \<latexrelease>   \__hook_try_declaring_generic_hook:nNNnn {#1}
703 \<latexrelease>   \hook_gput_code:nnn \__hook_gput_undeclared_hook:nnn
704 \<latexrelease> }
705 \<latexrelease>\cs_new_protected:Npn
706 \<latexrelease> \__hook_try_declaring_generic_next_hook:nn #1
707 \<latexrelease> {
708 \<latexrelease>   \__hook_try_declaring_generic_hook:nNNnn {#1}
709 \<latexrelease>   \hook_gput_next_code:nn \__hook_gput_next_do:nn
710 \<latexrelease> }

```

(End of definition for __hook_try_declaring_generic_hook:nnn and
__hook_try_declaring_generic_next_hook:nn.)

__hook_try_declaring_generic_hook:nNNnn __hook_try_declaring_generic_hook:nNNnn now splits the hook name at the first / (if any) and first checks if it is a file-specific hook (they require some normalization) using __hook_if_file_hook:wTF. If not then check it is one of a predefined set for generic names. We also split off the second component to see if we have to make a reversed hook. In either case the function returns <true> for a generic hook and <false> in other cases.

```

711 \<latexrelease>\cs_new_protected:Npn \__hook_try_declaring_generic_hook:nNNnn #1

```

```

712 <latexrelease> {
713 <latexrelease>   \_hook_if_file_hook:wTF #1 / / \s__hook_mark
714 <latexrelease>     {
715 <latexrelease>       \exp_args:Ne
716 <latexrelease>       \_hook_try_declaring_generic_hook_split:nNNnn
717 <latexrelease>       { \exp_args:Ne \_hook_file_hook_normalize:n {#1} }
718 <latexrelease>     }
719 <latexrelease>     { \_hook_try_declaring_generic_hook_split:nNNnn {#1} }
720 <latexrelease>   }

721 <latexrelease> \cs_new_protected:Npn
722 <latexrelease>   \_hook_try_declaring_generic_hook_split:nNNnn #1 #2 #3
723 <latexrelease> {
724 <latexrelease>   \_hook_try_declaring_generic_hook:wTF #1 / / / \scan_stop:
725 <latexrelease>     {#1}
726 <latexrelease>     { #2 }
727 <latexrelease>     { #3 } {#1}
728 <latexrelease> }
729 <latexrelease> \EndIncludeInRelease

(End of definition for \_hook_try_declaring_generic_hook:nNNnn and
\_hook_try_declaring_generic_hook_split:nNNnn.)

```

_hook_try_declaring_generic_hook:wTF

```

730 <latexrelease> \IncludeInRelease{2023/06/01}
731 <latexrelease>   {\_hook_try_declaring_generic_hook:wn}
732 <latexrelease>   {Hooks-with-args}
733 \prg_new_protected_conditional:Npnn
734   \_hook_try_declaring_generic_hook:wn
735   #1 / #2 / #3 / #4 \scan_stop: #5 { TF }
736 {
737   \_hook_if_generic:nTF {#5}
738   {
739     \_hook_if_usable:nF {#5}
740     {

```

If the hook doesn't exist yet we check if it is a cmd hook and if so we attempt patching the command in addition to declaring the hook.

For some commands this will not be possible, in which case _hook_patch_cmd_or_delay:Nnn (defined in ltcmdhooks) will generate an appropriate error message.

```

741   \str_if_eq:nnT {#1} { cmd }
742   {
743     \_hook_try_put_cmd_hook:n {#5}
744     \_hook_make_usable:nn {#5} { 9 }
745     \use_none:nnn
746   }

```

Declare the hook always even if it can't really be used (error message generated elsewhere).

Here we use _hook_make_usable:nn, so that a \hook_new:n is still possible later. Generic hooks (except cmd hooks) take no arguments, so use zero as the second argument.

```

747     \_hook_make_usable:nn {#5} { 0 }
748   }
749   \_hook_if_generic_reversed:nT {#5}
750   { \tl_gset:cn { g__hook_#5_reversed_tl } { - } }

```

```

751     \prg_return_true:
752     }
753     {

```

Generic hooks are all named $\langle type \rangle / \langle name \rangle / \langle place \rangle$, where $\langle type \rangle$ and $\langle place \rangle$ are predefined ($\backslash c_hook_generic_ \langle type \rangle / . / \langle place \rangle_t1$), and $\langle name \rangle$ is the variable component. Older releases had some hooks with the $\langle name \rangle$ in the third part, so the code below supports that syntax for a while, with a warning.

The $\backslash exp_after:wN \dots \backslash exp:w$ trick is there to remove the conditional structure inserted by $\backslash_hook_try_declaring_generic_hook:wNTF$ and thus allow access to the tokens that follow it, as is needed to keep things going.

When the deprecation cycle ends, the lines below should all be replaced by $\backslash prg_return_false:$.

```

754     \_hook_if_deprecated_generic:nTF {#5}
755     {
756         \_hook_deprecated_generic_warn:n {#5}
757         \exp_after:wN \_hook_declare_deprecated_generic:NNn
758         \exp:w % \exp_end:
759     }
760     { \prg_return_false: }
761 }
762 }

```

$\backslash_hook_deprecated_generic_warn:n$ will issue a deprecation warning for a given hook, and mark that hook such that the warning will not be issued again (multiple warnings can be issued, but only once per hook).

```

\_hook_deprecated_generic_warn:Nn
\_hook_deprecated_generic_warn:Nw

```

```

763 \cs_new_protected:Npn \_hook_deprecated_generic_warn:n #1
764 { \_hook_deprecated_generic_warn:w #1 \s__hook_mark }
765 \cs_new_protected:Npn \_hook_deprecated_generic_warn:w
766 #1 / #2 / #3 \s__hook_mark
767 {
768     \if_cs_exist:w __hook~#1/#2/#3 \cs_end: \else:
769         \msg_warning:nnnnn { hooks } { generic-deprecated } {#1} {#2} {#3}
770     \fi:
771     \cs_gset_eq:cN { __hook~#1/#2/#3 } \scan_stop:
772 }

```

Now that the user has been told about the deprecation, we proceed by swapping $\langle name \rangle$ and $\langle place \rangle$ and adding the code to the correct hook.

```

\_hook_do_deprecated_generic:Nn
\_hook_do_deprecated_generic:Nw
\_hook_declare_deprecated_generic:NNw
\_hook_declare_deprecated_generic:NNw

```

```

773 \cs_new_protected:Npn \_hook_do_deprecated_generic:Nn #1 #2
774 { \_hook_do_deprecated_generic:Nw #1 #2 \s__hook_mark }
775 \cs_new_protected:Npn \_hook_do_deprecated_generic:Nw #1
776 #2 / #3 / #4 \s__hook_mark
777 { #1 { #2 / #4 / #3 } }
778 \cs_new_protected:Npn \_hook_declare_deprecated_generic:NNn #1 #2 #3
779 { \_hook_declare_deprecated_generic:NNw #1 #2 #3 \s__hook_mark }
780 \cs_new_protected:Npn \_hook_declare_deprecated_generic:NNw #1 #2
781 #3 / #4 / #5 \s__hook_mark
782 {
783     \_hook_try_declaring_generic_hook:wNTF #3 / #5 / #4 / \scan_stop:
784     { #3 / #5 / #4 }
785     #1 #2 { #3 / #5 / #4 }
786 }
787 <latexrelease>\EndIncludeInRelease

```

```

788 <latexrelease> \IncludeInRelease{2021/11/15}
789 <latexrelease>           {\_hook_try_declaring_generic_hook:wn}
790 <latexrelease>           {Standardize-generic-hook-names}
791 <latexrelease> \prg_new_protected_conditional:Npnn
792 <latexrelease>   \_hook_try_declaring_generic_hook:wn
793 <latexrelease>   #1 / #2 / #3 / #4 \scan_stop: #5 { TF }
794 <latexrelease>   {
795 <latexrelease>     \_hook_if_generic:nTF {#5}
796 <latexrelease>     {
797 <latexrelease>       \_hook_if_usable:nF {#5}
798 <latexrelease>       {
799 <latexrelease>         \str_if_eq:nnT {#1} { cmd }
800 <latexrelease>         { \_hook_try_put_cmd_hook:n {#5} }
801 <latexrelease>         \_hook_make_usable:n {#5}
802 <latexrelease>       }
803 <latexrelease>     \_hook_if_generic_reversed:nT {#5}
804 <latexrelease>     { \tl_gset:cn { g_hook_#5_reversed_tl } { - } }
805 <latexrelease>     \prg_return_true:
806 <latexrelease>   }
807 <latexrelease>   {
808 <latexrelease>     \_hook_if_deprecated_generic:nTF {#5}
809 <latexrelease>     {
810 <latexrelease>       \_hook_deprecated_generic_warn:n {#5}
811 <latexrelease>       \exp_after:wN \_hook_declare_deprecated_generic:NNn
812 <latexrelease>       \exp:w % \exp_end:
813 <latexrelease>     }
814 <latexrelease>     { \prg_return_false: }
815 <latexrelease>   }
816 <latexrelease> }
817 <latexrelease> \EndIncludeInRelease

818 <latexrelease> \IncludeInRelease{2021/06/01}
819 <latexrelease>           {\_hook_try_declaring_generic_hook:wn}
820 <latexrelease>           {Support~cmd-hooks}
821 <latexrelease> \prg_new_protected_conditional:Npnn
822 <latexrelease>   \_hook_try_declaring_generic_hook:wn
823 <latexrelease>   #1 / #2 / #3 / #4 \scan_stop: #5 { TF }
824 <latexrelease>   {
825 <latexrelease>     \tl_if_empty:nTF {#2}
826 <latexrelease>     { \prg_return_false: }
827 <latexrelease>     {
828 <latexrelease>       \prop_if_in:NnTF \c__hook_generics_prop {#1}
829 <latexrelease>       {
830 <latexrelease>         \_hook_if_usable:nF {#5}
831 <latexrelease>         {
832 <latexrelease>           \str_if_eq:nnT {#1} { cmd }
833 <latexrelease>           { \_hook_try_put_cmd_hook:n {#5} }
834 <latexrelease>           \_hook_make_usable:n {#5}
835 <latexrelease>         }
836 <latexrelease>       \prop_if_in:NnTF
837 <latexrelease>       \c__hook_generics_reversed_ii_prop {#2}
838 <latexrelease>       { \tl_gset:cn { g__hook_#5_reversed_tl } { - } }
839 <latexrelease>       {
840 <latexrelease>         \prop_if_in:NnT
841 <latexrelease>         \c__hook_generics_reversed_iii_prop {#3}

```

```

842 <latexrelease>          { \tl_gset:cn { g__hook_#5_reversed_tl } { - } }
843 <latexrelease>          }
844 <latexrelease>          \prg_return_true:
845 <latexrelease>          }
846 <latexrelease>          { \prg_return_false: }
847 <latexrelease>        }
848 <latexrelease>      }
849 <latexrelease> \EndIncludeInRelease

850 <latexrelease> \IncludeInRelease{2020/10/01}
851 <latexrelease>          { \__hook_try_declaring_generic_hook:wn }
852 <latexrelease>          { Support~cmd~hooks }
853 <latexrelease> \prg_new_protected_conditional:Npnn
854 <latexrelease>   \__hook_try_declaring_generic_hook:wn
855 <latexrelease>   #1 / #2 / #3 / #4 \scan_stop: #5 { TF }
856 <latexrelease> {
857 <latexrelease>   \tl_if_empty:nTF {#2}
858 <latexrelease>   { \prg_return_false: }
859 <latexrelease>   {
860 <latexrelease>     \prop_if_in:NnTF \c__hook_generics_prop {#1}
861 <latexrelease>     {
862 <latexrelease>       \__hook_if_declared:nF {#5} { \hook_new:n {#5} }
863 <latexrelease>       \prop_if_in:NnTF
864 <latexrelease>       \c__hook_generics_reversed_ii_prop {#2}
865 <latexrelease>       { \tl_gset:cn { g__hook_#5_reversed_tl } { - } }
866 <latexrelease>       {
867 <latexrelease>         \prop_if_in:NnT
868 <latexrelease>         \c__hook_generics_reversed_iii_prop {#3}
869 <latexrelease>         { \tl_gset:cn { g__hook_#5_reversed_tl } { - } }
870 <latexrelease>       }
871 <latexrelease>       \prg_return_true:
872 <latexrelease>     }
873 <latexrelease>     { \prg_return_false: }
874 <latexrelease>   }
875 <latexrelease> }
876 <latexrelease> \EndIncludeInRelease

```

(End of definition for __hook_try_declaring_generic_hook:wnTF and others.)

__hook_if_file_hook_p:w __hook_if_file_hook:wTF checks if the argument is a valid file-specific hook (not, for example, file/before, but file/foo.tex/before). If it is a file-specific hook, then it executes the `<true>` branch, otherwise `<false>`.

```

877 <latexrelease> \IncludeInRelease{2021/11/15}{\__hook_if_file_hook:w}
878 <latexrelease>          { Standardize~generic~hook~names }
879 <latexrelease> \EndIncludeInRelease
880 <latexrelease> \IncludeInRelease{2020/10/01}{\__hook_if_file_hook:w}
881 <latexrelease>          { Standardize~generic~hook~names }
882 <latexrelease> \prg_new_conditional:Npnn \__hook_if_file_hook:w
883 <latexrelease>   #1 / #2 / #3 \s__hook_mark { TF }
884 <latexrelease> {
885 <latexrelease>   \str_if_eq:nnTF {#1} { file }
886 <latexrelease>   {
887 <latexrelease>     \bool_lazy_or:nnTF
888 <latexrelease>     { \tl_if_empty_p:n {#3} }
889 <latexrelease>     { \str_if_eq_p:nn {#3} { / } }

```

```

890 <latexrelease>          { \prg_return_false: }
891 <latexrelease>          {
892 <latexrelease>          \prop_if_in:NnTF \c__hook_generics_file_prop {#2}
893 <latexrelease>          { \prg_return_true: }
894 <latexrelease>          { \prg_return_false: }
895 <latexrelease>          }
896 <latexrelease>          }
897 <latexrelease>          { \prg_return_false: }
898 <latexrelease>          }
899 <latexrelease> \EndIncludeInRelease

```

(End of definition for __hook_if_file_hook:wTF.)

```

\__hook_file_hook_normalize:n
\__hook_strip_double_slash:n 900 <latexrelease> \IncludeInRelease{2021/11/15}{\__hook_file_hook_normalize:n}
\__hook_strip_double_slash:w 901 <latexrelease>          {Standardize-generic-hook-names}
902 <latexrelease> \EndIncludeInRelease

```

When a file-specific hook is found, before being declared it is lightly normalized by __hook_file_hook_normalize:n. The current implementation just replaces two consecutive slashes (//) by a single one, to cope with simple cases where the user did something like \def\input@path{./mypath/}, in which case a hook would have to be \AddToHook{file/./mypath//file.tex/after}.

```

903 <latexrelease> \IncludeInRelease{2020/10/01}{\__hook_file_hook_normalize:n}
904 <latexrelease>          {Standardize-generic-hook-names}
905 <latexrelease> \cs_new:Npn \__hook_file_hook_normalize:n #1
906 <latexrelease> { \__hook_strip_double_slash:n {#1} }
907 <latexrelease> \cs_new:Npn \__hook_strip_double_slash:n #1
908 <latexrelease> { \__hook_strip_double_slash:w #1 // \s__hook_mark }

```

This function is always called after testing if the argument is a file hook with __hook_if_file_hook:wTF, so we can assume it has three parts (it is either file/.../before or file/.../after), so we use #1/#2/#3 // instead of just #1 // to prevent losing a slash if the file name is empty.

```

909 <latexrelease> \cs_new:Npn \__hook_strip_double_slash:w #1/#2/#3//#4\s__hook_mark
910 <latexrelease> {
911 <latexrelease>   \tl_if_empty:nTF {#4}
912 <latexrelease>   { #1/#2/#3 }
913 <latexrelease>   { \__hook_strip_double_slash:w #1/#2/#3 /#4\s__hook_mark }
914 <latexrelease> }
915 <latexrelease> \EndIncludeInRelease

```

(End of definition for __hook_file_hook_normalize:n, __hook_strip_double_slash:n, and __hook_strip_double_slash:w.)

<pre> \c__hook_generic_cmd/.before_tl \c__hook_generic_cmd/.after_tl \c__hook_generic_env/.before_tl \c__hook_generic_env/.after_tl \c__hook_generic_file/.before_tl \c__hook_generic_file/.after_tl \c__hook_generic_package/.before_tl \c__hook_generic_package/.after_tl \c__hook_generic_class/.before_tl \c__hook_generic_class/.after_tl \c__hook_generic_include/.before_tl \c__hook_generic_include/.after_tl \c__hook_generic_env/.begin_tl \c__hook_generic_env/.end_tl \c__hook_generic_include/.end_tl </pre>	<p>Token lists defining the possible generic hooks. We don't provide any user interface to this as this is meant to be static.</p> <p>cmd The generic hooks used for commands.</p> <p>env The generic hooks used in \begin and \end.</p> <p>file, package, class, include The generic hooks used when loading a file</p> <p>File 08: lthooks.dtx Date: 2025/02/05 Version v1.11</p>
---	--


```

916 <latexrelease>\IncludeInRelease{2021/11/15}{\c__hook_generics_prop}
917 <latexrelease>           {Standardize-generic-hook-names}
918 \clist_map_inline:nn { cmd , env , file , package , class , include }
919   {
920     \tl_const:cn { c__hook_generic_#1/./before_tl } { + }
921     \tl_const:cn { c__hook_generic_#1/./after_tl } { - }
922   }
923 \tl_const:cn { c__hook_generic_env/./begin_tl } { + }
924 \tl_const:cn { c__hook_generic_env/./end_tl } { + }
925 \tl_const:cn { c__hook_generic_include/./end_tl } { - }
926 \tl_const:cn { c__hook_generic_include/./excluded_tl } { + }

```

Deprecated generic hooks:

```

927 \clist_map_inline:nn { file , package , class , include }
928   {
929     \tl_const:cn { c__hook_deprecated_#1/./before_tl } { }
930     \tl_const:cn { c__hook_deprecated_#1/./after_tl } { }
931   }
932 \tl_const:cn { c__hook_deprecated_include/./end_tl } { }
933 <latexrelease>\EndIncludeInRelease
934 <latexrelease>\IncludeInRelease{2020/10/01}{\c__hook_generics_prop}
935 <latexrelease>           {Standardize-generic-hook-names}
936 <latexrelease>\prop_const_from_keyval:Nn \c__hook_generics_prop
937 <latexrelease>   {cmd=,env=,file=,package=,class=,include=}
938 <latexrelease>\EndIncludeInRelease

```

(End of definition for \c__hook_generic_cmd/./before_tl and others.)

```

\c__hook_generics_reversed_ii_prop
\c__hook_generics_reversed_iii_prop
\c__hook_generics_file_prop

```

The following generic hooks are supposed to use reverse ordering (the ii and iii names are kept for the deprecation cycle):

```

939 <latexrelease>\IncludeInRelease{2021/11/15}{\c__hook_generics_reversed_ii_prop}
940 <latexrelease>           {Standardize-generic-hook-names}
941 <latexrelease>\EndIncludeInRelease
942 <latexrelease>\IncludeInRelease{2020/10/01}{\c__hook_generics_reversed_ii_prop}
943 <latexrelease>           {Standardize-generic-hook-names}
944 <latexrelease>\prop_const_from_keyval:Nn
945 <latexrelease>   \c__hook_generics_reversed_ii_prop {after=,end=}
946 <latexrelease>\prop_const_from_keyval:Nn
947 <latexrelease>   \c__hook_generics_reversed_iii_prop {after=}
948 <latexrelease>\prop_const_from_keyval:Nn
949 <latexrelease>   \c__hook_generics_file_prop {before=,after=}
950 <latexrelease>\EndIncludeInRelease

```

(End of definition for \c__hook_generics_reversed_ii_prop, \c__hook_generics_reversed_iii_prop, and \c__hook_generics_file_prop.)

```

\c__hook_parameter_cmd/./before_tl
\c__hook_parameter_cmd/./after_tl

```

Token lists defining the number of arguments for a given type of generic hook.

```

951 <latexrelease>\IncludeInRelease{2023/06/01}{\c__hook_parameter_cmd/./before_tl}
952 <latexrelease>           {Hooks-with-args}

```

cmd hooks are declared with 9 arguments because they have a variable number of arguments (depending on the command they are attached to), so we use the maximum here.

```

953 \tl_const:cn { c__hook_parameter_cmd/./before_tl } { #1#2#3#4#5#6#7#8#9 }
954 \tl_const:cn { c__hook_parameter_cmd/./after_tl } { #1#2#3#4#5#6#7#8#9 }

```

```

955 <latexrelease>\EndIncludeInRelease
956 <latexrelease>\IncludeInRelease{2020/10/01}{\c__hook_parameter_cmd/./before_tl}
957 <latexrelease>           {Hooks~with~args}
958 <latexrelease>\EndIncludeInRelease

```

(End of definition for \c__hook_parameter_cmd/./before_tl and
\c__hook_parameter_cmd/./after_tl.)

`\hook_gremove_code:nn` With `\hook_gremove_code:nn{<hook>}{<label>}` any code for `<hook>` stored under `<label>` is removed.

```

959 <latexrelease>\IncludeInRelease{2023/06/01}{\hook_gremove_code:nn}
960 <latexrelease>           {Hooks~with~args}
961 \cs_new_protected:Npn \hook_gremove_code:nn #1 #2
962   { \__hook_normalize_hook_args:Nnn \__hook_gremove_code:nn {#1} {#2} }
963 \cs_new_protected:Npn \__hook_gremove_code:nn #1 #2
964   {

```

First check that the hook code pool exists. `__hook_if_usable:nTF` isn't used here because it should be possible to remove code from a hook before its defined (see section 2.1.8).

```

965   \__hook_if_structure_exist:nTF {#1}
966     {

```

Then remove the chunk and run `__hook_update_hook_code:n` so that the execution token list reflects the change if we are after `\begin{document}`.

If all code is to be removed, clear the code pool `\g__hook_<hook>_code_prop`, the top-level code `__hook_toplevel_<hook>`, and the next-execution code `__hook_next_<hook>`.

```

967     \str_if_eq:nnTF {#2} {*}
968     {
969         \prop_gclear:c { g__hook_#1_code_prop }
970         \__hook_toplevel_gset:nn {#1} { }
971         \__hook_next_gset:nn {#1} { }
972     }
973     {

```

If the label is top-level then clear the token list, as all code there is under the same label.

```

974         \str_if_eq:nnTF {#2} { top-level }
975         { \__hook_toplevel_gset:nn {#1} { } }
976         {
977             \prop_gpop:cnNF { g__hook_#1_code_prop }
978             {#2} \l__hook_return_tl
979             { \msg_warning:nnnn { hooks } { cannot-remove } {#1} {#2} }
980         }
981     }

```

Finally update the code, if the hook exists.

```

982     \__hook_if_usable:nT {#1}
983     { \__hook_update_hook_code:n {#1} }
984 }

```

If the code pool for this hook doesn't exist, show a warning:

```

985 {
986     \__hook_if_deprecated_generic:nTF {#1}
987     {

```

```

988         \_hook_deprecated_generic_warn:n {#1}
989         \_hook_do_deprecated_generic:Nn
990         \_hook_gremove_code:nn {#1} {#2}
991     }
992     { \msg_warning:nmmm { hooks } { cannot-remove } {#1} {#2} }
993 }
994 }
995 \<latexrelease>\EndIncludeInRelease

996 \<latexrelease>\IncludeInRelease{2020/10/01}{\hook_gremove_code:nn}
997 \<latexrelease>
998 \<latexrelease>\cs_new_protected:Npn \_hook_gremove_code:nn #1 #2
999 \<latexrelease> {
1000 \<latexrelease> \_hook_if_structure_exist:nTF {#1}
1001 \<latexrelease> {
1002 \<latexrelease> \str_if_eq:nnTF {#2} {*}
1003 \<latexrelease> {
1004 \<latexrelease> \prop_gclear:c { g__hook_#1_code_prop }
1005 \<latexrelease> \_hook_tl_gclear:c { \_hook_toplevel~#1 }
1006 \<latexrelease> \_hook_tl_gclear:c { \_hook_next~#1 }
1007 \<latexrelease> }
1008 \<latexrelease> {
1009 \<latexrelease> \str_if_eq:nnTF {#2} { top-level }
1010 \<latexrelease> { \_hook_tl_gclear:c { \_hook_toplevel~#1 } }
1011 \<latexrelease> {
1012 \<latexrelease> \prop_gpop:cnNF { g__hook_#1_code_prop }
1013 \<latexrelease> {#2} \l__hook_return_tl
1014 \<latexrelease> { \msg_warning:nmmm { hooks } { cannot-remove }
1015 \<latexrelease> {#1} {#2} }
1016 \<latexrelease> }
1017 \<latexrelease> }
1018 \<latexrelease> \_hook_if_usable:nT {#1}
1019 \<latexrelease> { \_hook_update_hook_code:n {#1} }
1020 \<latexrelease> }
1021 \<latexrelease> {
1022 \<latexrelease> \_hook_if_deprecated_generic:nTF {#1}
1023 \<latexrelease> {
1024 \<latexrelease> \_hook_deprecated_generic_warn:n {#1}
1025 \<latexrelease> \_hook_do_deprecated_generic:Nn
1026 \<latexrelease> \_hook_gremove_code:nn {#1} {#2}
1027 \<latexrelease> }
1028 \<latexrelease> { \msg_warning:nmmm { hooks } { cannot-remove }
1029 \<latexrelease> {#1} {#2} }
1030 \<latexrelease> }
1031 \<latexrelease> }
1032 \<latexrelease>\EndIncludeInRelease

```

(End of definition for `\hook_gremove_code:nn` and `_hook_gremove_code:nn`. This function is documented on page 213.)

```

\_hook_cs_gput_right:nnn This macro is used to append code to the toplevel and next token lists, treating them
\_hook_cs_gput_right_fast:nnn correctly depending on their number of arguments, and depending on whether the code
\_hook_cs_gput_right_slow:nnn being added should have parameter tokens understood as parameters, or doubled to be
\_hook_code_gset_auxi:nnnn stored as parameter tokens.
\_hook_code_gset_auxi:eeen
1033 \<latexrelease>\IncludeInRelease{2023/06/01}{\_hook_cs_gput_right:nnn}

```

1034 <latexrelease> *{Hooks-with-args}*

Check if the current hook is declared and takes no arguments. In this case, we short-circuit and use the simpler and much faster approach that doesn't require hash-doubling.

```

1035 \cs_new_protected:Npn \__hook_cs_gput_right:nnn #1 #2
1036 {
1037   \if:w T
1038     \__hook_if_declared:nF {#2} { F }
1039     \tl_if_empty:cF { c__hook_#2_parameter_tl } { F }
1040     T
1041     \exp_after:wN \__hook_cs_gput_right_fast:nnn
1042   \else:
1043     \exp_after:wN \__hook_cs_gput_right_slow:nnn
1044   \fi:
1045   {#1} {#2}
1046 }
1047 \cs_new_protected:Npn \__hook_cs_gput_right_fast:nnn #1 #2 #3
1048 { \cs_gset:cpx { __hook#1~#2 }
1049   { \exp_not:v { __hook#1~#2 } \exp_not:n {#3} } }
1050 \cs_new_protected:Npn \__hook_cs_gput_right_slow:nnn #1 #2 #3
1051 {

```

The auxiliary `__hook_code_gset_auxi:eeen` just does the assignment at the end. Its first argument is the parameter text of the macro, which is chosen here depending if `\c__hook_(hook)_parameter_tl` exists, if the hook is declared, and if it's a generic hook.

```

1052 \cs_if_exist:cF { __hook#1~#2 }
1053 { \__hook_code_gset_aux:nnn {#1} {#2} { } }
1054 \__hook_code_gset_auxi:eeen
1055 {
1056   \__hook_if_declared:nTF {#2}
1057   { \tl_use:c { c__hook_#2_parameter_tl } }
1058   {
1059     \__hook_if_generic:nTF {#2}
1060     { \__hook_generic_parameter:n {#2} }
1061     { \c__hook_nine_parameters_tl }
1062   }
1063 }

```

Here we take the existing code in the macro, expand it with as many arguments as it takes, then double the hashes so the code can be reused.

PhO: Maybe can be improved. The case of adding to an empty cs can be optimized by quickly checking `\cs_replacement_spec`.

```

1064 {
1065   \exp_args:NNo \exp_args:No \__hook_double_hashes:n
1066   {
1067     \cs:w __hook#1~#2 \exp_last_unbraced:Ne \cs_end:
1068     { \__hook_braced_cs_parameter:n { __hook#1~#2 } }
1069   }
1070 }

```

Now the new code: if we are replacing arguments, then hashes are left untouched, otherwise they are doubled.

```

1071 {
1072   \__hook_if_replacing_args:TF
1073   { \exp_not:n }
1074   { \__hook_double_hashes:n }
1075   {#3}
1076 }

```

And finally, the csname which we'll define with all the above.

```
1077     { __hook#1~#2 }
1078 }
```

And as promised, the auxiliary that does the definition.

```
1079 \cs_new_protected:Npn \__hook_code_gset_auxi:nnnn #1 #2 #3 #4
1080   { \cs_gset:cpn {#4} #1 { #2 #3 } }
1081 \cs_generate_variant:Nn \__hook_code_gset_auxi:nnnn { een }
1082 \<latexrelease>\EndIncludeInRelease
1083 \<latexrelease>\IncludeInRelease{2020/10/01}{\__hook_cs_gput_right:nnn}
1084 \<latexrelease>
1085   {Hooks-with-args}
1086 \<latexrelease>\cs_undefine:N \__hook_cs_gput_right:nnn
1087 \<latexrelease>\cs_undefine:N \__hook_cs_gput_right_fast:nnn
1088 \<latexrelease>\cs_undefine:N \__hook_cs_gput_right_slow:nnn
1089 \<latexrelease>\cs_undefine:N \__hook_code_gset_auxi:nnnn
1090 \<latexrelease>\EndIncludeInRelease
```

(End of definition for __hook_cs_gput_right:nnn and others.)

These macros define `__hook<type>_<hook>` (with `<type>` being `_next`, `_toplevel`, or empty) with the given code and the parameters stored in `\c__hook_<hook>_parameter_tl` (or none, if that doesn't exist).

```
\__hook_code_gset:nn
\__hook_code_gset:ne
\__hook_toplevel_gset:nn
\__hook_next_gset:nn
\__hook_code_gset_aux:nnn
1090 \<latexrelease>\IncludeInRelease{2023/06/01}{\__hook_code_gset:nn}
1091 \<latexrelease>
1092   {Hooks-with-args}
1093 \cs_new_protected:Npn \__hook_code_gset:nn
1094   { \__hook_code_gset_aux:nnn { } }
1095 \cs_new_protected:Npn \__hook_toplevel_gset:nn
1096   { \__hook_code_gset_aux:nnn { _toplevel } }
1097 \cs_new_protected:Npn \__hook_next_gset:nn
1098   { \__hook_code_gset_aux:nnn { _next } }
1099 \cs_new_protected:Npn \__hook_code_gset_aux:nnn #1 #2 #3
1100   {
1101     \cs_gset:cpn { __hook#1~#2 \exp_last_unbraced:Ne }
1102     { \__hook_parameter:n {#2} }
1103     {#3}
1104   }
1105 \cs_generate_variant:Nn \__hook_code_gset:nn { ne }
1106 \<latexrelease>\EndIncludeInRelease
1107 \<latexrelease>\IncludeInRelease{2020/10/01}{\__hook_code_gset:nn}
1108 \<latexrelease>
1109   {Hooks-with-args}
1110 \<latexrelease>\cs_undefine:N \__hook_code_gset:nn
1111 \<latexrelease>\cs_undefine:N \__hook_toplevel_gset:nn
1112 \<latexrelease>\cs_undefine:N \__hook_next_gset:nn
1113 \<latexrelease>\cs_undefine:N \__hook_code_gset_aux:nnn
1114 \<latexrelease>\EndIncludeInRelease
```

(End of definition for __hook_code_gset:nn and others.)

This macro normalizes the parameters of the macros `__hook<type>_<hook>` to take the right number of arguments after a hook is declared. At this point we know `\c__hook_<hook>_parameter_tl` exists, so use that to count the arguments and use that as `<parameter text>` for the newly (re)defined macro.

```
1113 \<latexrelease>\IncludeInRelease{2023/06/01}{\__hook_normalise_cs_args:nn}
```

```

1114 <latexrelease> {Hooks-with-args}
1115 \cs_new_protected:Npn \__hook_normalise_cs_args:nn #1 #2
1116 {
1117   \cs_if_exist:cT { __hook#1~#2 }
1118   {
1119     \__hook_code_gset_auxi:eeen
1120     { \tl_use:c { c__hook_#2_parameter_tl } }
1121     {
1122       \exp_args:NNo \exp_args:No \__hook_double_hashes:n
1123       {
1124         \cs:w __hook#1~#2 \exp_last_unbraced:Ne \cs_end:
1125         { \__hook_braced_cs_parameter:n { __hook#1~#2 } }
1126       }
1127     }
1128     { }
1129     { __hook#1~#2 }
1130   }
1131 }
1132 <latexrelease> \EndIncludeInRelease
1133 <latexrelease> \IncludeInRelease{2020/10/01}{\__hook_normalise_cs_args:nn}
1134 <latexrelease> {Hooks-with-args}
1135 <latexrelease> \cs_undefine:N \__hook_normalise_cs_args:nn
1136 <latexrelease> \EndIncludeInRelease

```

(End of definition for __hook_normalise_cs_args:nn.)

```

\__hook_normalise_code_pool:n
\__hook_set_normalise_fn:nn

```

This one's a bit of a hack. It takes a hook, and iterates over its code pool (`__hook_<hook>_code_prop`), redefining each code label to use only valid arguments. This is used when, for example, a code is added referencing arguments #1 and #2, but the hook has only #1. In this example, every reference to #2 is changed to ##2. This is done because otherwise T_EX will throw a low-level error every time some change happens to the hook (code is added, a rule is set, etc), which can get quite repetitive for no good reason.

```

1137 <latexrelease> \IncludeInRelease{2023/06/01}{\__hook_normalise_code_pool:n}
1138 <latexrelease> {Hooks-with-args}
1139 \cs_new_protected:Npn \__hook_normalise_code_pool:n #1
1140 {

```

First, call `__hook_set_normalise_fn:nn` with the hook name to set everything up, then we'll loop over the hook's code pool applying the normalization above. After that's done, copy the temporary property list back to the hook's.

```

1141   \__hook_set_normalise_fn:nn {#1} { Offending-label:~'##1' }
1142   \prop_clear:N \l__hook_work_prop
1143   \prop_map_function:cN { g__hook_#1_code_prop } \__hook_normalise_fn:nn
1144   \prop_gset_eq:cN { g__hook_#1_code_prop } \l__hook_work_prop
1145 }

```

The sole purpose of this function is to define `__hook_normalise_fn:nn`, which will then do the correcting of the code being added to the hook.

```

1146 \cs_new_protected:Npn \__hook_set_normalise_fn:nn #1 #2
1147 {

```

To start, we define two auxiliary token lists. `\l__hook_tmpb_tl` contains:

```

{\c__hook_hashes_tl 1}
{\c__hook_hashes_tl 2}
...
{\c__hook_hashes_tl 9}
1148   \cs_set:Npn \__hook_tmp:w ##1##2##3##4##5##6##7##8##9 { }
1149   \tl_set:Ne \l__hook_tmpb_tl
1150     { \__hook_braced_cs_parameter:n { __hook_tmp:w } }
1151   \group_begin:
1152     \__hook_tl_set:cn { c__hook_hash_tl } { \exp_not:N \c__hook_hashes_tl }
1153     \use:e
1154     {
1155   \group_end:
1156   \tl_set:Nn \exp_not:N \l__hook_tmpb_tl { \l__hook_tmpb_tl }
1157   }

```

And `\l__hook_tmpa_tl` contains:

```

{\c__hook_hash_tl 1}
{\c__hook_hash_tl 2}
...
{\c__hook_hash_tl <n>}

```

with `<n>` being the number of arguments declared for the hook.

```

1158   \exp_last_unbraced:NNf
1159   \cs_set:Npn \__hook_tmp:w { \__hook_parameter:n {#1} } { }
1160   \tl_set:Ne \l__hook_tmpa_tl
1161     { \__hook_braced_cs_parameter:n { __hook_tmp:w } }

```

Now this function does the fun part. It is meant to be used with `\prop_map_function:NN`, taking a label name in `##1` and the code stored in that label in `##2`.

```

1162   \cs_gset_protected:Npx \__hook_normalise_fn:nn ##1 ##2
1163   {

```

Here we'll define two auxiliary macros: the first one throws an error when it detects an invalid argument reference. It piggybacks on TeX's low-level "Illegal parameter number" error, but it defines a weirdly-named control sequence so that the error comes out nicely formatted. For example, if the label "badpkg" adds some code that references argument #3 in the hook "foo", which takes only two arguments, the error will be:

```

! Illegal parameter number in definition of hook 'foo'.
(hooks)          Offending label: 'badpkg'.
<to be read again>

```

3

At the point of this definition, the error is raised if the code happens to reference an invalid argument. If it was possible to detect that this definition raised no error, the next step would be unnecessary. We'll do all this in a group so this weird definition doesn't leak out, and set `\tex_escapechar:D` to `-1` so this hack shows up extra nice in the case of an error.

```

1164   \group_begin:
1165     \int_set:Nn \tex_escapechar:D { -1 }
1166     \cs_set:cpn
1167     {
1168       hook-'#1'. ^^J

```

```

1169         (hooks) \prg_replicate:nn { 13 } { ~ }
1170         #2 % more message text
1171     }
1172     \exp_not:v { c__hook_#1_parameter_tl }
1173     {##2}
1174     \group_end:

```

This next macro, with a much less fabulous name, takes always nine arguments, and it just transfers the code ##2 under the label ##1 to the temporary property list. The first $\langle n \rangle$ arguments are taken from `\l__hook_tmpa_tl`, and the other $9 - \langle n \rangle$ taken from `\l__hook_tmpb_tl` (which contains twice as many # tokens as the former). Then, `__hook_double_hashes:n` is used to double non-argument hashes, and expand the `\c__hook_hash_tl` and `\c__hook_hashes_tl` to the actual parameter tokens.

```

1175     \cs_set:Npn \exp_not:N \__hook_tmp:w
1176     \exp_not:V \c__hook_nine_parameters_tl
1177     {
1178     \prop_put:Nne \exp_not:N \l__hook_work_prop
1179     {##1} { \exp_not:N \__hook_double_hashes:n {##2} }
1180     }

```

This next macro, with a much less fabulous name, takes always nine arguments, and it just transfers the code ##2 under the label ##1 to the temporary property list. The first $\langle n \rangle$ arguments are taken from `\l__hook_tmpa_tl`, and the other $9 - \langle n \rangle$ taken from `\l__hook_tmpb_tl` (which contains twice as many # tokens as the former). Then, `__hook_double_hashes:n` is used to double non-argument hashes, and expand the `\c__hook_hash_tl` and `\c__hook_hashes_tl` to the actual parameter tokens.

```

1181     \exp_not:N \__hook_tmp:w
1182     \exp_not:V \l__hook_tmpa_tl
1183     \exp_args:No \exp_not:o
1184     { \exp_after:wN \__hook_tmp:w \l__hook_tmpb_tl }
1185     }
1186 }
1187 \cs_new_eq:NN \__hook_normalise_fn:nn ?
1188 \langle latexrelease \rangle \EndIncludeInRelease

```

```

1189 \langle latexrelease \rangle \IncludeInRelease{2020/10/01}{\__hook_normalise_code_pool:n}
1190 \langle latexrelease \rangle {Hooks-with-args}
1191 \langle latexrelease \rangle \cs_undefine:N \__hook_normalise_code_pool:n
1192 \langle latexrelease \rangle \EndIncludeInRelease

```

Check if the expansion of a control sequence is empty by looking at its replacement text.

```

\__hook_cs_if_empty_p:c
\__hook_cs_if_empty:cTF
1193 \langle latexrelease \rangle \IncludeInRelease{2023/06/01}{\__hook_cs_if_empty:c}
1194 \langle latexrelease \rangle {Hooks-with-args}
1195 \prg_new_conditional:Npnn \__hook_cs_if_empty:c #1 { p, T, F, TF }
1196 {
1197     \if:w \scan_stop: \__hook_replacement_spec:c {#1} \scan_stop:
1198     \prg_return_true:
1199     \else:
1200     \prg_return_false:
1201     \fi:
1202 }
1203 \cs_new:Npn \__hook_replacement_spec:c #1
1204 {

```



```

1205 \exp_args:Nc \token_if_macro:NT {#1}
1206 { \cs_replacement_spec:c {#1} }
1207 }
1208 <latexrelease>\EndIncludeInRelease
1209 <latexrelease>\IncludeInRelease{2020/10/01}{\__hook_cs_if_empty:c}
1210 <latexrelease> {Hooks-with-args}
1211 <latexrelease>\cs_undefine:N \__hook_cs_if_empty:c
1212 <latexrelease>\EndIncludeInRelease

(End of definition for \__hook_normalise_code_pool:n, \__hook_set_normalise_fn:nn, and
 \__hook_cs_if_empty:cTF.)

```

`__hook_braced_cs_parameter:n` Looks at the *<parameter text>* of a control sequence, and returns a run of “hidden” zero to nine arguments. This works as long as the macros take a simple run of zero to nine arguments. The parameters are “hidden” because the parameter tokens are returned inside `\c__hook_hash_tl` instead of explicitly, so that `__hook_double_hashes:n` won’t touch these.

```

1213 <latexrelease>\IncludeInRelease{2023/06/01}{\__hook_braced_cs_parameter:n}
1214 <latexrelease> {Hooks-with-args}
1215 \cs_new:Npn \__hook_braced_cs_parameter:n #1
1216 {
1217 \exp_last_unbraced:Ne \__hook_braced_hidden_loop:w
1218 { \exp_args:Nc \__hook_cs_parameter_count:N {#1} } ? \s__hook_mark
1219 }
1220 \cs_new:Npn \__hook_braced_hidden_loop:w #1
1221 {
1222 \if:w ? #1
1223 \__hook_use_i_delimit_by_s_mark:nw
1224 \fi:
1225 { \exp_not:N \c__hook_hash_tl #1 }
1226 \__hook_braced_hidden_loop:w
1227 }
1228 \cs_new:Npn \__hook_cs_parameter_count:N #1
1229 {
1230 \exp_last_unbraced:Nf \__hook_cs_parameter_count:w
1231 { \token_if_macro:NT #1 { \cs_parameter_spec:N #1 } }
1232 ? \__hook_cs_end:w ? \__hook_cs_end:w ? \__hook_cs_end:w
1233 ? \__hook_cs_end:w ? \__hook_cs_end:w ? \__hook_cs_end:w
1234 ? \__hook_cs_end:w ? \__hook_cs_end:w ? \__hook_cs_end:w
1235 \s__hook_mark
1236 }
1237 \cs_new:Npn \__hook_cs_parameter_count:w #1#2 #3#4 #5#6 #7#8
1238 { #2 #4 #6 #8 \__hook_cs_parameter_count:w }
1239 \cs_new:Npn \__hook_cs_end:w #1 \s__hook_mark { }
1240 <latexrelease>\EndIncludeInRelease

```

This function can’t be undefined when rolling back because it’s used at the end of this module to adequate the hook data structures to previous versions.

```

1241 <latexrelease>\IncludeInRelease{2020/10/01}{\__hook_braced_cs_parameter:n}
1242 <latexrelease> {Hooks-with-args}
1243 <latexrelease>\EndIncludeInRelease

```

(End of definition for `__hook_braced_cs_parameter:n` and others.)

`_hook_braced_parameter:n` This one is used in simpler cases, where no special handling of hashes is required. This is used only inside `_hook_initialize_hook_code:n`, so it assumes `\c_hook_⟨hook⟩_parameter_tl` is defined, but should work otherwise.

```

1244 <latexrelease>\IncludeInRelease{2023/06/01}{\_hook_braced_parameter:n}
1245 <latexrelease>          {Hooks-with-args}
1246 \cs_new:Npn \_hook_braced_parameter:n #1
1247   {
1248     \if_case:w
1249       \int_eval:n
1250         { \exp_args:Nv \str_count:n { c\_hook_#1_parameter_tl } / 3 }
1251       \exp_stop_f:
1252     \or: {##1}
1253     \or: {##1} {##2}
1254     \or: {##1} {##2} {##3}
1255     \or: {##1} {##2} {##3} {##4}
1256     \or: {##1} {##2} {##3} {##4} {##5}
1257     \or: {##1} {##2} {##3} {##4} {##5} {##6}
1258     \or: {##1} {##2} {##3} {##4} {##5} {##6} {##7}
1259     \or: {##1} {##2} {##3} {##4} {##5} {##6} {##7} {##8}
1260     \or: {##1} {##2} {##3} {##4} {##5} {##6} {##7} {##8} {##9}
1261     \else:
1262       \msg_expandable_error:nnn { latex2e } { should-not-happen }
1263       { Invalid~parameter~spec. }
1264     \fi:
1265   }
1266 <latexrelease>\EndIncludeInRelease
1267 <latexrelease>\IncludeInRelease{2020/10/01}{\_hook_braced_parameter:n}
1268 <latexrelease>          {Hooks-with-args}
1269 <latexrelease>\cs_undefine:N \_hook_braced_parameter:n
1270 <latexrelease>\EndIncludeInRelease

```

(End of definition for `_hook_braced_parameter:n` and `_hook_braced_real_loop:w`.)

`_hook_parameter:n` This is just a shortcut to e- or f-expand to the `⟨parameter text⟩` of the hook.

```

1271 <latexrelease>\IncludeInRelease{2023/06/01}{\_hook_parameter:n}
1272 <latexrelease>          {Hooks-with-args}
1273 \cs_new:Npn \_hook_parameter:n #1
1274   {
1275     \cs:w c\_hook_
1276     \tl_if_exist:cTF { c\_hook_#1_parameter_tl }
1277       { #1_parameter } { empty }
1278     _tl \cs_end:
1279   }
1280 \cs_new:Npn \_hook_generic_parameter:n #1
1281   { \_hook_generic_parameter:w #1 / / \s\_hook_mark }
1282 \cs_new:Npn \_hook_generic_parameter:w #1 / #2 / #3 / #4 \s\_hook_mark
1283   {
1284     \cs_if_exist_use:cF { c\_hook_parameter_#1/./#3_tl }
1285     { \c\_hook_empty_tl }
1286   }
1287 <latexrelease>\EndIncludeInRelease
1288 <latexrelease>\IncludeInRelease{2020/10/01}{\_hook_parameter:n}
1289 <latexrelease>          {Hooks-with-args}

```

```

1290 <latexrelease>\cs_undefine:N \__hook_parameter:n
1291 <latexrelease>\cs_undefine:N \__hook_generic_parameter:n
1292 <latexrelease>\EndIncludeInRelease

```

(End of definition for __hook_parameter:n.)

4.7 Setting rules for hooks code

```

\g__hook_??_code_prop
  \__hook~??
\g__hook_??_reversed_tl
\c__hook_??_parameter_tl

```

Initially these variables simply used an empty “label” name (not two question marks). This was a bit unfortunate, because then `l3doc` complains about `__` in the middle of a command name when trying to typeset the documentation. However using a “normal” name such as `default` has the disadvantage of that being not really distinguishable from a real hook name. I now have settled for `??` which needs some gymnastics to get it into the `csname`, but since this is used a lot, the code should be fast, so this is not done with `c` expansion in the code later on.

`__hook_??` isn’t used, but it has to be defined to trick the code into thinking that `??` is actually a hook.

```

1293 \prop_new:c { g__hook_??_code_prop }
1294 \prop_new:c { __hook~?? }

```

Default rules are always given in normal ordering (never in reversed ordering). If such a rule is applied to a reversed hook it behaves as if the rule is reversed (e.g., `after` becomes `before`) because those rules are applied first and then the order is reversed.

```

1295 \tl_new:c { g__hook_??_reversed_tl }

```

The parameter text for the “default” hook is empty.

```

1296 <latexrelease>\IncludeInRelease{2023/06/01}{\c__hook_??_parameter_tl}
1297 <latexrelease>          {Hooks-with-args}
1298 \tl_const:cn { c__hook_??_parameter_tl } { }
1299 <latexrelease>\EndIncludeInRelease
1300 <latexrelease>\IncludeInRelease{2020/10/01}{\c__hook_??_parameter_tl}
1301 <latexrelease>          {Hooks-with-args}
1302 <latexrelease>\cs_undefine:c { c__hook_??_parameter_tl }
1303 <latexrelease>\EndIncludeInRelease

```

(End of definition for `\g__hook_??_code_prop` and others.)

```

\hook_gset_rule:nmnn
\__hook_gset_rule:nmnn

```

With `\hook_gset_rule:nmnn{<hook>}{<label1>}{<relation>}{<label2>}` a relation is defined between the two code labels for the given `<hook>`. The special hook `??` stands for *any* hook, which sets a default rule (to be used if no other relation between the two hooks exist).

```

1304 \cs_new_protected:Npn \hook_gset_rule:nmnn #1#2#3#4
1305   {
1306     \__hook_normalize_hook_rule_args:Nmnnn \__hook_gset_rule:nmnn
1307       {#1} {#2} {#3} {#4}
1308   }
1309 <latexrelease>\IncludeInRelease{2022/06/01}{\__hook_gset_rule:nmnn}
1310 <latexrelease>          {Refuse-setting-rule-for-one-time-hooks}
1311 \cs_new_protected:Npn \__hook_gset_rule:nmnn #1#2#3#4
1312   {
1313     \__hook_if_deprecated_generic:nT {#1}
1314     {
1315       \__hook_deprecated_generic_warn:n {#1}

```

```

1316     \_hook_do_deprecated_generic:Nn \_hook_gset_rule:nnnn {#1}
1317     {#2} {#3} {#4}
1318     \_hook_use_none_delimit_by_s_mark:w
1319   }
1320 \_hook_if_execute_immediately:nT {#1}
1321 {
1322   \msg_error:nnnnnn { hooks } { rule-too-late }
1323   {#1} {#2} {#3} {#4}
1324   \_hook_use_none_delimit_by_s_mark:w
1325 }

```

First we ensure the basic data structure of the hook exists:

```
1326 \_hook_init_structure:n {#1}
```

Then we clear any previous relationship between both labels.

```
1327 \_hook_rule_gclear:nnn {#1} {#2} {#4}
```

Then we call the function to handle the given rule. Throw an error if the rule is invalid.

```

1328 \cs_if_exist_use:cTF { \_hook_rule_#3_gset:nnn }
1329 {
1330   {#1} {#2} {#4}
1331   \_hook_update_hook_code:n {#1}
1332 }
1333 {
1334   \msg_error:nnnnnn { hooks } { unknown-rule }
1335   {#1} {#2} {#3} {#4}
1336 }
1337 \s\_hook_mark
1338 }

```

```

1339 <latexrelease> \EndIncludeInRelease
1340 <latexrelease> \IncludeInRelease{2020/10/01}{\_hook_gset_rule:nnnn}
1341 <latexrelease> {Refuse-setting-rule-for-one-time-hooks}
1342 <latexrelease> \cs_new_protected:Npn \_hook_gset_rule:nnnn #1#2#3#4
1343 <latexrelease> {
1344 <latexrelease>   \_hook_if_deprecated_generic:nT {#1}
1345 <latexrelease>   {
1346 <latexrelease>     \_hook_deprecated_generic_warn:n {#1}
1347 <latexrelease>     \_hook_do_deprecated_generic:Nn \_hook_gset_rule:nnnn
1348 <latexrelease>     {#1} {#2} {#3} {#4}
1349 <latexrelease>     \exp_after:wN \use_none:nnnnnnnn \use_none:n
1350 <latexrelease>   }
1351 <latexrelease>   \_hook_init_structure:n {#1}
1352 <latexrelease>   \_hook_rule_gclear:nnn {#1} {#2} {#4}
1353 <latexrelease>   \cs_if_exist_use:cTF { \_hook_rule_#3_gset:nnn }
1354 <latexrelease>   {
1355 <latexrelease>     {#1} {#2} {#4}
1356 <latexrelease>     \_hook_update_hook_code:n {#1}
1357 <latexrelease>   }
1358 <latexrelease>   {
1359 <latexrelease>     \msg_error:nnnnnn { hooks } { unknown-rule }
1360 <latexrelease>     {#1} {#2} {#3} {#4}
1361 <latexrelease>   }
1362 <latexrelease> }
1363 <latexrelease> \EndIncludeInRelease

```

(End of definition for `\hook_gset_rule:nnnn` and `__hook_gset_rule:nnnn`. This function is documented on page 214.)

`__hook_rule_before_gset:nnn` Then we add the new rule. We need to normalize the rules here to allow for faster processing later. Given a pair of labels l_A and l_B , the rule $l_A > l_B$ is the same as $l_B < l_A$ only presented differently. But by normalizing the forms of the rule to a single representation, say, $l_B < l_A$, reduces the time spent looking for the rules later considerably.

Here we do that normalization by using `\(pdf)strcmp` to lexically sort labels l_A and l_B to a fixed order. This order is then enforced every time these two labels are used together.

Here we use `__hook_label_pair:nn` $\{\langle hook \rangle\} \{\langle l_A \rangle\} \{\langle l_B \rangle\}$ to build a string $l_B || l_A$ with a fixed order, and use `__hook_label_ordered:nnTF` to apply the correct rule to the pair of labels, depending if it was sorted or not.

```

1364 \cs_new_protected:Npn \__hook_rule_before_gset:nnn #1#2#3
1365   {
1366     \__hook_tl_gset:cx
1367     { g__hook_#1_rule_ \__hook_label_pair:nn {#2} {#3} _tl }
1368     { \__hook_label_ordered:nnTF {#2} {#3} { < } { > } }
1369   }
1370 \cs_new_eq:cN { __hook_rule_<_gset:nnn } \__hook_rule_before_gset:nnn
1371 \cs_new_protected:Npn \__hook_rule_after_gset:nnn #1#2#3
1372   {
1373     \__hook_tl_gset:cx
1374     { g__hook_#1_rule_ \__hook_label_pair:nn {#3} {#2} _tl }
1375     { \__hook_label_ordered:nnTF {#3} {#2} { < } { > } }
1376   }
1377 \cs_new_eq:cN { __hook_rule_>_gset:nnn } \__hook_rule_after_gset:nnn

```

(End of definition for `__hook_rule_before_gset:nnn` and others.)

`__hook_rule_voids_gset:nnn` This rule removes (clears, actually) the code from label #3 if label #2 is in the hook #1.

```

1378 \cs_new_protected:Npn \__hook_rule_voids_gset:nnn #1#2#3
1379   {
1380     \__hook_tl_gset:cx
1381     { g__hook_#1_rule_ \__hook_label_pair:nn {#2} {#3} _tl }
1382     { \__hook_label_ordered:nnTF {#2} {#3} { -> } { <- } }
1383   }

```

(End of definition for `__hook_rule_voids_gset:nnn`.)

`__hook_rule_incompatible-error_gset:nnn` These relations make an error/warning if labels #2 and #3 appear together in hook #1.

```

\__hook_rule_incompatible-warning_gset:nnn
1384 \cs_new_protected:cpn { __hook_rule_incompatible-error_gset:nnn } #1#2#3
1385   { \__hook_tl_gset:cn
1386     { g__hook_#1_rule_ \__hook_label_pair:nn {#2} {#3} _tl }
1387     { xE }
1388   }
1389 \cs_new_protected:cpn { __hook_rule_incompatible-warning_gset:nnn } #1#2#3
1390   { \__hook_tl_gset:cn
1391     { g__hook_#1_rule_ \__hook_label_pair:nn {#2} {#3} _tl }
1392     { xW }
1393   }

```

(End of definition for `__hook_rule_incompatible-error_gset:nnn` and `__hook_rule_incompatible-warning_gset:nnn`.)

`_hook_rule_unrelated_gset:nnn` Undo a setting. `_hook_rule_unrelated_gset:nnn` doesn't need to do anything, since we use `_hook_rule_gclear:nnn` before setting any rule.

```

1394 \cs_new_protected:Npn \_hook_rule_unrelated_gset:nnn #1#2#3 { }
1395 \cs_new_protected:Npn \_hook_rule_gclear:nnn #1#2#3
1396   { \cs_undefine:c { g_hook_#1_rule_ \_hook_label_pair:nn {#2} {#3} _t1 } }

```

(End of definition for `_hook_rule_unrelated_gset:nnn` and `_hook_rule_gclear:nnn`.)

`_hook_label_pair:nn` Ensure that the lexically greater label comes first.

```

1397 \cs_new:Npn \_hook_label_pair:nn #1#2
1398   {
1399     \if_case:w \_hook_str_compare:nn {#1} {#2} \exp_stop_f:
1400       #1 | #1 % 0
1401     \or:   #1 | #2 % +1
1402     \else: #2 | #1 % -1
1403     \fi:
1404   }

```

(End of definition for `_hook_label_pair:nn`.)

`_hook_label_ordered_p:nn` Check that labels #1 and #2 are in the correct order (as returned by `_hook_label_pair:nn`) and if so return true, else return false.

```

1405 \prg_new_conditional:Npnn \_hook_label_ordered:nn #1#2 { TF }
1406   {
1407     \if_int_compare:w \_hook_str_compare:nn {#1} {#2} > 0 \exp_stop_f:
1408       \prg_return_true:
1409     \else:
1410       \prg_return_false:
1411     \fi:
1412   }

```

(End of definition for `_hook_label_ordered:nnTF`.)

`_hook_if_label_case:nnnnn` To avoid doing the string comparison twice in `_hook_initialize_single:NNn` (once with `\str_if_eq:nn` and again with `_hook_label_ordered:nn`), we use a three-way branching macro that will compare #1 and #2 and expand to `\use_i:nnn` if they are equal, `\use_ii:nn` if #1 is lexically greater, and `\use_iii:nn` otherwise.

```

1413 \cs_new:Npn \_hook_if_label_case:nnnnn #1#2
1414   {
1415     \cs:w use_
1416     \if_case:w \_hook_str_compare:nn {#1} {#2}
1417       i \or: ii \else: iii \fi: :nnn
1418     \cs_end:
1419   }

```

(End of definition for `_hook_if_label_case:nnnnn`.)

`_hook_update_hook_code:n` Before `\begin{document}` this does nothing, in the body it reinitializes the hook code using the altered data.

```

1420 \cs_new_eq:NN \_hook_update_hook_code:n \use_none:n

```

(End of definition for `_hook_update_hook_code:n`.)

`__hook_initialize_all:` Initialize all known hooks (at `\begin{document}`), i.e., update the fast execution token lists to hold the necessary code in the right order.

```

1421 <latexrelease> \IncludeInRelease{2023/06/01}{\__hook_initialize_all:}
1422 <latexrelease>           {Hooks-with-args}
1423 \cs_new_protected:Npn \__hook_initialize_all:
1424   {

```

First we change `__hook_update_hook_code:n` which so far was a no-op to now initialize one hook. This way any later updates to the hook will run that code and also update the execution token list.

```

1425   \cs_gset_eq:NN \__hook_update_hook_code:n \__hook_initialize_hook_code:n

```

Now we loop over all hooks that have been defined and update each of them. Here we have to determine if the hook has arguments so that auxiliaries know what to do with hashes. We look at `\c__hook_{hook}_parameter_tl`, if it has any parameters, and set `replacing_args` accordingly.

```

1426   \__hook_debug:n { \prop_gclear:N \g__hook_used_prop }
1427   \seq_map_inline:Nn \g__hook_all_seq
1428     {
1429       \tl_if_empty:cTF { c__hook_##1_parameter_tl }
1430         { \__hook_replacing_args_false: }
1431         { \__hook_replacing_args_true: }
1432       \__hook_update_hook_code:n {##1}
1433       \__hook_replacing_args_reset:
1434     }

```

If we are debugging we show results hook by hook for all hooks that have data.

```

1435   \__hook_debug:n
1436     {
1437       \iow_term:x { ^^J[lthooks]~ All-initialized~(non-empty)~hooks: }
1438       \prop_map_inline:Nn \g__hook_used_prop
1439         {
1440           \iow_term:x
1441             { ^^J ~ ##1 ~ -> ~ \cs_replacement_spec:c { __hook~##1 } ~ }
1442         }
1443     }

```

After all hooks are initialized we change the “use” to just call the hook code and not initialize it (as this was already done in the preamble).

```

1444   \__hook_post_initialization_defs:
1445 }

1446 <latexrelease> \EndIncludeInRelease
1447 <latexrelease> \IncludeInRelease{2020/10/01}{\__hook_initialize_all:}
1448 <latexrelease>           {Hooks-with-args}
1449 <latexrelease> \cs_gset_protected:Npn \__hook_initialize_all:
1450 <latexrelease>   {
1451 <latexrelease>     \cs_gset_eq:NN \__hook_update_hook_code:n
1452 <latexrelease>       \__hook_initialize_hook_code:n
1453 <latexrelease>     \__hook_debug:n { \prop_gclear:N \g__hook_used_prop }
1454 <latexrelease>     \seq_map_inline:Nn \g__hook_all_seq
1455 <latexrelease>       { \__hook_update_hook_code:n {##1} }
1456 <latexrelease>     \__hook_debug:n
1457 <latexrelease>     {
1458 <latexrelease>       \iow_term:x{^^JAll~ initialized~ (non-empty)~ hooks:}

```

```

1459 <latexrelease> \prop_map_inline:Nn \g__hook_used_prop
1460 <latexrelease> {
1461 <latexrelease> \iow_term:x
1462 <latexrelease> { ^^J ~ ##1 ~ -> ~
1463 <latexrelease> \cs_replacement_spec:c { __hook~##1 } ~ }
1464 <latexrelease> }
1465 <latexrelease> }
1466 <latexrelease> \cs_gset_eq:NN \hook_use:n \__hook_use_initialized:n
1467 <latexrelease> \cs_gset_eq:NN \__hook_preamble_hook:n \use_none:n
1468 <latexrelease> }
1469 <@@=
1470 <latexrelease> \cs_gset_eq:NN \@expl@@@initialize@all@@
1471 <latexrelease> \__hook_initialize_all:
1472 <@@=hook
1473 <latexrelease> \EndIncludeInRelease

```

(End of definition for __hook_initialize_all:.)

`__hook_initialize_hook_code:n` Initializing or reinitializing the fast execution hook code. In the preamble this is selectively done in case a hook gets used and at `\begin{document}` this is done for all hooks and afterwards only if the hook code changes.

```

1474 <latexrelease> \IncludeInRelease{2023/06/01}{\__hook_initialize_hook_code:n}
1475 <latexrelease> {Hooks-with-args}
1476 \cs_new_protected:Npn \__hook_initialize_hook_code:n #1
1477 {
1478 \__hook_debug:n
1479 { \iow_term:x { ^^J[lthooks]~ Update~code~for~hook~'##1' \on@line :^^J } }

```

This does the sorting and the updates. First thing we do is to check if a legacy hook macro exists and if so we add it to the hook under the label `legacy`. This might make the hook non-empty so we have to do this before the then following test.

```

1480 \__hook_include_legacy_code_chunk:n {#1}

```

If there aren't any code chunks for the current hook, there is no point in even starting the sorting routine so we make a quick test for that and in that case just update `__hook_<hook>` to hold the top-level and next code chunks. If there are code chunks we call `__hook_initialize_single:NNn` and pass to it ready made csnames as they are needed several times inside. This way we save a bit on processing time if we do that up front.

```

1481 \__hook_if_usable:nT {#1}
1482 {
1483 \prop_if_empty:cTF { g__hook_#1_code_prop }
1484 {
1485 \__hook_code_gset:ne {#1}
1486 {

```

The hook may take arguments, so we add a run of braced parameters after the `_next` and `_toplevel` macros, so that the arguments passed to the hook are forwarded to them.

```

1487 \exp_not:c { __hook_toplevel~##1 }
1488 \__hook_braced_parameter:n {#1}
1489 \exp_not:c { __hook_next~##1 }
1490 \__hook_braced_parameter:n {#1}
1491 }
1492 }
1493 {

```


By default the algorithm sorts the code chunks and then saves the result in a token list for fast execution; this is done by adding the code chunks one after another, using `\tl_gput_right:NV`. When we sort code for a reversed hook, all we have to do is to add the code chunks in the opposite order into the token list. So all we have to do in preparation is to change two definitions that are used later on.

```

1494     \__hook_if_reversed:nTF {#1}
1495     { \cs_set_eq:NN \__hook_tl_gput:Nn \__hook_tl_gput_left:Nn
1496       \cs_set_eq:NN \__hook_clist_gput:NV \clist_gput_left:NV }
1497     { \cs_set_eq:NN \__hook_tl_gput:Nn \__hook_tl_gput_right:Nn
1498       \cs_set_eq:NN \__hook_clist_gput:NV \clist_gput_right:NV }

```

When sorting, some relations (namely voids) need to act destructively on the code property lists to remove code that shouldn't appear in the sorted hook token list, so we make a copy of the code property list that we can safely work on without changing the main one.

```

1499     \prop_set_eq:Nc \l__hook_work_prop { g__hook_#1_code_prop }
1500     \__hook_initialize_single:ccn
1501     { __hook~#1 } { g__hook_#1_labels_clist } {#1}

```

For debug display we want to keep track of those hooks that actually got code added to them, so we record that in `plist`. We use a `plist` to ensure that we record each hook name only once, i.e., we are only interested in storing the keys and the value is arbitrary.

```

1502     \__hook_debug:n
1503     { \exp_args:NNx \prop_gput:Nnn \g__hook_used_prop {#1} { } }
1504   }
1505 }
1506 }

```

1507 `\EndIncludeInRelease`

```

1508 \IncludeInRelease{2020/10/01}{\__hook_initialize_hook_code:n}
1509 {Hooks-with-args}
1510 \cs_gset_protected:Npn \__hook_initialize_hook_code:n #1
1511 {
1512   \__hook_debug:n
1513   { \iow_term:x { ^^J Update-code-for-hook~'#1'
1514     \on@line :^^J } }
1515   \__hook_include_legacy_code_chunk:n {#1}
1516   \__hook_if_usable:nT {#1}
1517   {
1518     \prop_if_empty:cTF { g__hook_#1_code_prop }
1519     {
1520       \__hook_tl_gset:co { __hook~#1 }
1521       {
1522         \cs:w __hook_toplevel~#1 \exp_after:wN \cs_end:
1523         \cs:w __hook_next~#1 \cs_end:
1524       }
1525     }
1526     {
1527       \__hook_if_reversed:nTF {#1}
1528       { \cs_set_eq:NN \__hook_tl_gput:Nn
1529         \__hook_tl_gput_left:Nn
1530         \cs_set_eq:NN \__hook_clist_gput:NV
1531         \clist_gput_left:NV }
1532       { \cs_set_eq:NN \__hook_tl_gput:Nn

```

```

1533 <latexrelease> \__hook_tl_gput_right:Nn
1534 <latexrelease> \cs_set_eq:NN \__hook_clist_gput:NV
1535 <latexrelease> \clist_gput_right:NV }
1536 <latexrelease> \prop_set_eq:Nc \l__hook_work_prop
1537 <latexrelease> { g__hook_#1_code_prop }
1538 <latexrelease> \__hook_initialize_single:ccn
1539 <latexrelease> { __hook~#1 } { g__hook_#1_labels_clist } {#1}
1540 <latexrelease> \__hook_debug:n
1541 <latexrelease> { \exp_args:NNx \prop_gput:Nnn \g__hook_used_prop
1542 <latexrelease> {#1} { } }
1543 <latexrelease> }
1544 <latexrelease> }
1545 <latexrelease> }
1546 <latexrelease> \EndIncludeInRelease

```

(End of definition for __hook_initialize_hook_code:n.)

__hook_tl_csname:n It is faster to pass a single token and expand it when necessary than to pass a bunch of
 __hook_seq_csname:n character tokens around.

FMI: note to myself: verify

```

1547 \cs_new:Npn \__hook_tl_csname:n #1 { l__hook_label_#1_tl }
1548 \cs_new:Npn \__hook_seq_csname:n #1 { l__hook_label_#1_seq }

```

(End of definition for __hook_tl_csname:n and __hook_seq_csname:n.)

\l__hook_labels_seq For the sorting I am basically implementing Knuth's algorithm for topological sorting as
 \l__hook_labels_int given in TAOCP volume 1 pages 263–266. For this algorithm we need a number of local
 \l__hook_front_tl variables:

- List of labels used in the current hook to label code chunks:

```

1549 \seq_new:N \l__hook_labels_seq

```

- Number of labels used in the current hook. In Knuth's algorithm this is called N :

```

1550 \int_new:N \l__hook_labels_int

```

- The sorted code list to be build is managed using two pointers one to the front of the queue and one to the rear. We model this using token list pointers. Knuth calls them F and R :

```

1551 \tl_new:N \l__hook_front_tl

```

```

1552 \tl_new:N \l__hook_rear_tl

```

- The data for the start of the queue is kept in this token list, it corresponds to what Don calls `QLINK[0]` but since we aren't manipulating individual words in memory it is slightly differently done:

```

1553 \tl_new:c { \__hook_tl_csname:n { 0 } }

```

(End of definition for \l__hook_labels_seq and others.)

`_hook_initialize_single:NNn` `_hook_initialize_single:ccn` `_hook_initialize_single:NNn` implements the sorting of the code chunks for a hook and saves the result in the token list for fast execution (#4). The arguments are `\hook-code-plist`, `\hook-code-tl`, `\hook-top-level-code-tl`, `\hook-next-code-tl`, `\hook-ordered-labels-clist` and `\hook-name` (the latter is only used for debugging—the `\hook-rule-plist` is accessed using the `\hook-name`).

The additional complexity compared to Don’s algorithm is that we do not use simple positive integers but have arbitrary alphanumeric labels. As usual Don’s data structures are chosen in a way that one can omit a lot of tests and I have mimicked that as far as possible. The result is a restriction I do not test for at the moment: a label can’t be equal to the number 0!

FMi: Needs checking for, just in case ... maybe

```

1554 \latexrelease\IncludeInRelease{2023/06/01}{\_hook_initialize_single:NNn}
1555 \latexrelease{Hooks-with-args}
1556 \cs_new_protected:Npn \_hook_initialize_single:NNn #1#2#3
1557 {

```

Step T1: Initialize the data structure ...

```

1558 \seq_clear:N \l_hook_labels_seq
1559 \int_zero:N \l_hook_labels_int

```

Store the name of the hook:

```

1560 \tl_set:Nn \l_hook_cur_hook_tl {#3}

```

We loop over the property list holding the code and record all the labels listed there. Only the rules for those labels are of interest to us. While we are at it we count them (which gives us the N in Knuth’s algorithm). The prefix `label_` is added to the variables to ensure that labels named `front`, `rear`, `labels`, or `return` don’t interact with our code.

```

1561 \prop_map_inline:Nn \l_hook_work_prop
1562 {
1563   \int_incr:N \l_hook_labels_int
1564   \seq_put_right:Nn \l_hook_labels_seq {##1}
1565   \__hook_tl_set:cn { \__hook_tl_csname:n {##1} } { 0 }
1566   \seq_clear_new:c { \__hook_seq_csname:n {##1} }
1567 }

```

Steps T2 and T3: Here we sort the relevant rules into the data structure...

This loop constitutes a square matrix of the labels in `\l_hook_work_prop` in the vertical and the horizontal directions. However, since the rule $l_A\langle rel\rangle l_B$ is the same as $l_B\langle rel\rangle^{-1}l_A$ we can cut the loop short at the diagonal of the matrix (*i.e.*, when both labels are equal), saving a good amount of time. The way the rules were set up (see the implementation of `_hook_rule_before_gset:nnn` above) ensures that we have no rule in the ignored side of the matrix, and all rules are seen. The rules are applied in `_hook_apply_label_pair:nnn`, which takes the properly-ordered pair of labels as argument.

```

1568 \prop_map_inline:Nn \l_hook_work_prop
1569 {
1570   \prop_map_inline:Nn \l_hook_work_prop
1571   {
1572     \__hook_if_label_case:nnnnn {##1} {####1}
1573     { \prop_map_break: }
1574     { \_hook_apply_label_pair:nnn {##1} {####1} }
1575     { \_hook_apply_label_pair:nnn {####1} {##1} }

```

```

1576             {#3}
1577         }
1578     }

```

Now take a breath, and look at the data structures that have been set up:

```

1579     \__hook_debug:n { \__hook_debug_label_data:N \l__hook_work_prop }

```

Step T4:

```

1580     \tl_set:Nn \l__hook_rear_tl { 0 }
1581     \tl_set:cn { \__hook_tl_csname:n { 0 } } { 0 }
1582     \seq_map_inline:Nn \l__hook_labels_seq
1583     {
1584         \int_compare:nNnT { \cs:w \__hook_tl_csname:n {##1} \cs_end: } = 0
1585         {
1586             \tl_set:cn { \__hook_tl_csname:n { \l__hook_rear_tl } } {##1}
1587             \tl_set:Nn \l__hook_rear_tl {##1}
1588         }
1589     }
1590     \tl_set_eq:Nc \l__hook_front_tl { \__hook_tl_csname:n { 0 } }
1591     \__hook_tl_gc_clear:N #1
1592     \clist_gc_clear:N #2

```

The whole loop gets combined in steps T5–T7:

```

1593     \bool_while_do:nn { ! \str_if_eq_p:Vn \l__hook_front_tl { 0 } }
1594     {

```

This part is step T5:

```

1595         \int_decr:N \l__hook_labels_int
1596         \prop_get:NVN \l__hook_work_prop \l__hook_front_tl \l__hook_return_tl
1597         \exp_args:NNV \__hook_tl_gput:Nn #1 \l__hook_return_tl
1598         \__hook_clist_gput:NV #2 \l__hook_front_tl
1599         \__hook_debug:n{ \iow_term:x{[lthooks]~ Handled~ code~ for~ \l__hook_front_tl} }

```

This is step T6, except that we don't use a pointer *P* to move through the successors, but instead use ##1 of the mapping function.

```

1600         \seq_map_inline:cn { \__hook_seq_csname:n { \l__hook_front_tl } }
1601         {
1602             \tl_set:cx { \__hook_tl_csname:n {##1} }
1603             { \int_eval:n
1604                 { \cs:w \__hook_tl_csname:n {##1} \cs_end: - 1 }
1605             }
1606             \int_compare:nNnT
1607             { \cs:w \__hook_tl_csname:n {##1} \cs_end: } = 0
1608             {
1609                 \tl_set:cn
1610                 { \__hook_tl_csname:n { \l__hook_rear_tl } } {##1}
1611                 \tl_set:Nn \l__hook_rear_tl {##1}
1612             }
1613         }

```

and here is step T7:

```

1614         \tl_set_eq:Nc \l__hook_front_tl
1615         { \__hook_tl_csname:n { \l__hook_front_tl } }

```

This is step T8: If we haven't moved the code for all labels (i.e., if `\l__hook_labels_int` is still greater than zero) we have a loop and our partial order can't be flattened out.

```

1616     }
1617     \int_compare:nNnF \l__hook_labels_int = 0
1618     {
1619         \iow_term:x{=====}
1620         \iow_term:x{Error:~ label~ rules~ are~ incompatible:}

```

This is not really the information one needs in the error case but it will do for now
...

FMI: improve output on a rainy day

```

1621     \__hook_debug_label_data:N \l__hook_work_prop
1622     \iow_term:x{=====}
1623     }

```

After we have added all hook code to #1, we finish it off by adding extra code for the `top-level` (#2) and for one time execution (#3). These should normally be empty. The `top-level` code is added with `__hook_tl_gput:Nn` as that might change for a reversed hook (then `top-level` is the very first code chunk added). The `next` code is always added last (to the right). The hook may take arguments, so we add a run of braced parameters after the `_next` and `_toplevel` macros, so that the arguments passed to the hook are forwarded to them.

```

1624     \exp_args:NNe \__hook_tl_gput:Nn #1
1625     { \exp_not:c { __hook_toplevel~#3 } \__hook_braced_parameter:n {#3} }
1626     \__hook_tl_gput_right:Ne #1
1627     { \exp_not:c { __hook_next~#3 } \__hook_braced_parameter:n {#3} }
1628     \use:e
1629     {
1630         \cs_gset:cpn { __hook~#3 } \use:c { c__hook_#3_parameter_tl }
1631         { \exp_not:V #1 }
1632     }
1633 }

```

```

1634 \cs_generate_variant:Nn \__hook_initialize_single:NNn { cc }
1635 \latexrelease\EndIncludeInRelease
1636 \latexrelease\IncludeInRelease{2020/10/01}{\__hook_initialize_single:NNn}
1637 \latexrelease{Hooks-with-args}
1638 \latexrelease\cs_new_protected:Npn \__hook_initialize_single:NNn #1#2#3
1639 \latexrelease{
1640     \seq_clear:N \l__hook_labels_seq
1641     \int_zero:N \l__hook_labels_int
1642     \tl_set:Nn \l__hook_cur_hook_tl {#3}
1643     \prop_map_inline:Nn \l__hook_work_prop
1644     {
1645         \int_incr:N \l__hook_labels_int
1646         \seq_put_right:Nn \l__hook_labels_seq {##1}
1647         \__hook_tl_set:cn { \__hook_tl_csname:n {##1} } { 0 }
1648         \seq_clear_new:c { \__hook_seq_csname:n {##1} }
1649     }
1650     \prop_map_inline:Nn \l__hook_work_prop
1651     {
1652         \prop_map_inline:Nn \l__hook_work_prop

```

```

1653 <latexrelease>      {
1654 <latexrelease>      \_hook_if_label_case:nnnnn {##1} {####1}
1655 <latexrelease>      { \prop_map_break: }
1656 <latexrelease>      { \_hook_apply_label_pair:nnn {##1} {####1} }
1657 <latexrelease>      { \_hook_apply_label_pair:nnn {####1} {##1} }
1658 <latexrelease>      {#3}
1659 <latexrelease>      }
1660 <latexrelease>      }
1661 <latexrelease> \_hook_debug:n
1662 <latexrelease>   { \_hook_debug_label_data:N \l__hook_work_prop }
1663 <latexrelease> \tl_set:Nn \l__hook_rear_tl { 0 }
1664 <latexrelease> \tl_set:cn { \_hook_tl_csname:n { 0 } } { 0 }
1665 <latexrelease> \seq_map_inline:Nn \l__hook_labels_seq
1666 <latexrelease>   {
1667 <latexrelease>     \int_compare:nNnT
1668 <latexrelease>     { \cs:w \_hook_tl_csname:n {##1} \cs_end: } = 0
1669 <latexrelease>     {
1670 <latexrelease>       \tl_set:cn { \_hook_tl_csname:n
1671 <latexrelease>         { \l__hook_rear_tl } } {##1}
1672 <latexrelease>       \tl_set:Nn \l__hook_rear_tl
1673 <latexrelease>         {##1}
1674 <latexrelease>     }
1675 <latexrelease> \tl_set_eq:Nc \l__hook_front_tl { \_hook_tl_csname:n { 0 } }
1676 <latexrelease> \_hook_tl_gclear:N #1
1677 <latexrelease> \clist_gclear:N #2
1678 <latexrelease> \bool_while_do:nn
1679 <latexrelease>   { ! \str_if_eq_p:Vn \l__hook_front_tl { 0 } }
1680 <latexrelease>   {
1681 <latexrelease>     \int_decr:N \l__hook_labels_int
1682 <latexrelease>     \prop_get:NVN \l__hook_work_prop
1683 <latexrelease>     \l__hook_front_tl \l__hook_return_tl
1684 <latexrelease>     \exp_args:NNV \_hook_tl_gput:Nn #1 \l__hook_return_tl
1685 <latexrelease>     \_hook_clist_gput:NV #2 \l__hook_front_tl
1686 <latexrelease>     \_hook_debug:n{ \iow_term:x
1687 <latexrelease>       {Handled~ code~ for~ \l__hook_front_tl} }
1688 <latexrelease>     \seq_map_inline:cn
1689 <latexrelease>     { \_hook_seq_csname:n { \l__hook_front_tl } }
1690 <latexrelease>     {
1691 <latexrelease>       \tl_set:cx { \_hook_tl_csname:n {##1} }
1692 <latexrelease>       { \int_eval:n
1693 <latexrelease>         { \cs:w \_hook_tl_csname:n {##1} \cs_end: - 1 }
1694 <latexrelease>       }
1695 <latexrelease>       \int_compare:nNnT
1696 <latexrelease>       { \cs:w \_hook_tl_csname:n {##1} \cs_end: } = 0
1697 <latexrelease>       {
1698 <latexrelease>         \tl_set:cn { \_hook_tl_csname:n
1699 <latexrelease>           { \l__hook_rear_tl } } {##1}
1700 <latexrelease>         \tl_set:Nn \l__hook_rear_tl
1701 <latexrelease>           {##1}
1702 <latexrelease>       }
1703 <latexrelease>     }
1704 <latexrelease> \tl_set_eq:Nc \l__hook_front_tl
1705 <latexrelease>   { \_hook_tl_csname:n { \l__hook_front_tl } }
1706 <latexrelease> }
1706 <latexrelease> \int_compare:nNnF \l__hook_labels_int = 0

```

```

1707 <latexrelease>      {
1708 <latexrelease>      \iow_term:x{=====}
1709 <latexrelease>      \iow_term:x{Error:~ label~ rules~ are~ incompatible:}
1710 <latexrelease>      \__hook_debug_label_data:N \l__hook_work_prop
1711 <latexrelease>      \iow_term:x{=====}
1712 <latexrelease>      }
1713 <latexrelease>      \exp_args:NNo \__hook_tl_gput:Nn #1
1714 <latexrelease>      { \cs:w __hook_toplevel~#3 \cs_end: }
1715 <latexrelease>      \__hook_tl_gput_right:No #1 { \cs:w __hook_next~#3 \cs_end: }
1716 <latexrelease>      }
1717 <latexrelease> \cs_generate_variant:Nn \__hook_tl_gput_right:Nn { No }
1718 <latexrelease> \EndIncludeInRelease

```

(End of definition for __hook_initialize_single:NNn.)

__hook_tl_gput:Nn These append either on the right (normal hook) or on the left (reversed hook). This is
 __hook_clist_gput:NV setup up in __hook_initialize_hook_code:n, elsewhere their behavior is undefined.

```

1719 \cs_new:Npn \__hook_tl_gput:Nn { \ERROR }
1720 \cs_new:Npn \__hook_clist_gput:NV { \ERROR }

```

(End of definition for __hook_tl_gput:Nn and __hook_clist_gput:NV.)

__hook_apply_label_pair:nnn
 __hook_label_if_exist_apply:nnnF

This is the payload of steps T2 and T3 executed in the loop described above. This macro assumes #1 and #2 are ordered, which means that any rule pertaining the pair #1 and #2 is \g__hook_<hook>_rule_#1|#2_tl, and not \g__hook_<hook>_rule_#2|#1_tl. This also saves a great deal of time since we only need to check the order of the labels once.

The arguments here are <label1>, <label2>, <hook>, and <hook-code-plist>. We are about to apply the next rule and enter it into the data structure. __hook_apply_label_pair:nnn will just call __hook_label_if_exist_apply:nnnF for the <hook>, and if no rule is found, also try the <hook> name ?? denoting a default hook rule.

__hook_label_if_exist_apply:nnnF will check if the rule exists for the given hook, and if so call __hook_apply_rule:nnn.

```

1721 \cs_new_protected:Npn \__hook_apply_label_pair:nnn #1#2#3
1722 {

```

Extra complication: as we use default rules and local hook specific rules we first have to check if there is a local rule and if that exist use it. Otherwise check if there is a default rule and use that.

```

1723   \__hook_label_if_exist_apply:nnnF {#1} {#2} {#3}
1724   {

```

If there is no hook-specific rule we check for a default one and use that if it exists.

```

1725       \__hook_label_if_exist_apply:nnnF {#1} {#2} { ?? } { }
1726   }
1727 }

```

```

1728 \cs_new_protected:Npn \__hook_label_if_exist_apply:nnnF #1#2#3
1729 {
1730   \if_cs_exist:w g__hook_ #3 _rule_ #1 | #2 _tl \cs_end:

```

What to do precisely depends on the type of rule we have encountered. If it is a before rule it will be handled by the algorithm but other types need to be managed differently. All this is done in __hook_apply_rule:nnnN.

```

1731   \__hook_apply_rule:nnn {#1} {#2} {#3}
1732   \exp_after:wN \use_none:n

```

```

1733 \else:
1734 \use:nn
1735 \fi:
1736 }

```

(End of definition for `__hook_apply_label_pair:nnn` and `__hook_label_if_exist_apply:nnnF`.)

`__hook_apply_rule:nnn` This is the code executed in steps T2 and T3 while looping through the matrix This is part of step T3. We are about to apply the next rule and enter it into the data structure. The arguments are `\langle label1 \rangle`, `\langle label2 \rangle`, `\langle hook-name \rangle`, and `\langle hook-code-plist \rangle`.

```

1737 \cs_new_protected:Npn \__hook_apply_rule:nnn #1#2#3
1738 {
1739 \cs:w __hook_apply_
1740 \cs:w g__hook_#3_reversed_tl \cs_end: rule_
1741 \cs:w g__hook_#3_rule_#1|#2_tl \cs_end: :nnn \cs_end:
1742 {#1} {#2} {#3}
1743 }

```

(End of definition for `__hook_apply_rule:nnn`.)

`__hook_apply_rule_<:nnn` The most common cases are `<` and `>` so we handle that first. They are relations `<` and `>` in TAOCP, and they dictate sorting.

`__hook_apply_rule_>:nnn`

```

1744 \cs_new_protected:cpn { __hook_apply_rule_<:nnn } #1#2#3
1745 {
1746 \__hook_debug:n { \__hook_msg_pair_found:nnn {#1} {#2} {#3} }
1747 \tl_set:cx { \__hook_tl_csname:n {#2} }
1748 { \int_eval:n{ \cs:w \__hook_tl_csname:n {#2} \cs_end: + 1 } }
1749 \seq_put_right:cn{ \__hook_seq_csname:n {#1} }{#2}
1750 }
1751 \cs_new_protected:cpn { __hook_apply_rule_>:nnn } #1#2#3
1752 {
1753 \__hook_debug:n { \__hook_msg_pair_found:nnn {#1} {#2} {#3} }
1754 \tl_set:cx { \__hook_tl_csname:n {#1} }
1755 { \int_eval:n{ \cs:w \__hook_tl_csname:n {#1} \cs_end: + 1 } }
1756 \seq_put_right:cn{ \__hook_seq_csname:n {#2} }{#1}
1757 }

```

(End of definition for `__hook_apply_rule_<:nnn` and `__hook_apply_rule_>:nnn`.)

`__hook_apply_rule_xE:nnn` These relations make two labels incompatible within a hook. `xE` makes raises an error if the labels are found in the same hook, and `xW` makes it a warning.

`__hook_apply_rule_xW:nnn`

```

1758 \cs_new_protected:cpn { __hook_apply_rule_xE:nnn } #1#2#3
1759 {
1760 \__hook_debug:n { \__hook_msg_pair_found:nnn {#1} {#2} {#3} }
1761 \msg_error:nnnnnn { hooks } { labels-incompatible }
1762 {#1} {#2} {#3} { 1 }
1763 \use:c { __hook_apply_rule_>:nnn } {#1} {#2} {#3}
1764 \use:c { __hook_apply_rule_<:nnn } {#1} {#2} {#3}
1765 }
1766 \cs_new_protected:cpn { __hook_apply_rule_xW:nnn } #1#2#3
1767 {
1768 \__hook_debug:n { \__hook_msg_pair_found:nnn {#1} {#2} {#3} }
1769 \msg_warning:nnnnnn { hooks } { labels-incompatible }
1770 {#1} {#2} {#3} { 0 }
1771 }

```


(End of definition for `__hook_apply_rule_xE:nnn` and `__hook_apply_rule_xW:nnn`.)

`__hook_apply_rule_>:nnn` If we see `->` we have to drop code for label #3 and carry on. We could do a little better
`__hook_apply_rule_<:nnn` and drop everything for that label since it doesn't matter where we put such empty code. However that would complicate the algorithm a lot with little gain.¹² So we still unnecessarily try to sort it in and depending on the rules that might result in a loop that is otherwise resolved. If that turns out to be a real issue, we can improve the code.

Here the code is removed from `\l__hook_cur_hook_tl` rather than #3 because the latter may be ??, and the default hook doesn't store any code. Removing it instead from `\l__hook_cur_hook_tl` makes the default rules `->` and `<-` work properly.

```

1772 \cs_new_protected:cpn { __hook_apply_rule_>:nnn } #1#2#3
1773   {
1774     \__hook_debug:n
1775     {
1776       \__hook_msg_pair_found:nnn {#1} {#2} {#3}
1777       \iow_term:x{--->~ Drop~ '#2'~ code~ from~
1778         \iow_char:N \ g__hook_ \l__hook_cur_hook_tl _code_prop ~
1779         because~ of~ '#1' }
1780     }
1781     \prop_put:Nnn \l__hook_work_prop {#2} { }
1782   }
1783 \cs_new_protected:cpn { __hook_apply_rule_<:nnn } #1#2#3
1784   {
1785     \__hook_debug:n
1786     {
1787       \__hook_msg_pair_found:nnn {#1} {#2} {#3}
1788       \iow_term:x{--->~ Drop~ '#1'~ code~ from~
1789         \iow_char:N \ g__hook_ \l__hook_cur_hook_tl _code_prop ~
1790         because~ of~ '#2' }
1791     }
1792     \prop_put:Nnn \l__hook_work_prop {#1} { }
1793   }

```

(End of definition for `__hook_apply_rule_>:nnn` and `__hook_apply_rule_<:nnn`.)

`__hook_apply_-rule_<:nnn` Reversed rules.

```

\__hook_apply_-rule_>:nnn 1794 \cs_new_eq:cc { __hook_apply_-rule_<:nnn } { __hook_apply_rule_>:nnn }
\__hook_apply_-rule_<:nnn 1795 \cs_new_eq:cc { __hook_apply_-rule_>:nnn } { __hook_apply_rule_<:nnn }
\__hook_apply_-rule_>:nnn 1796 \cs_new_eq:cc { __hook_apply_-rule_<:nnn } { __hook_apply_rule_<:nnn }
\__hook_apply_-rule_xW:nnn 1797 \cs_new_eq:cc { __hook_apply_-rule_>:nnn } { __hook_apply_rule_>:nnn }
\__hook_apply_-rule_xE:nnn 1798 \cs_new_eq:cc { __hook_apply_-rule_xE:nnn } { __hook_apply_rule_xE:nnn }
1799 \cs_new_eq:cc { __hook_apply_-rule_xW:nnn } { __hook_apply_rule_xW:nnn }

```

(End of definition for `__hook_apply_-rule_<:nnn` and others.)

`__hook_msg_pair_found:nnn` A macro to avoid moving this many tokens around.

```

1800 \cs_new_protected:Npn \__hook_msg_pair_found:nnn #1#2#3
1801   {
1802     \iow_term:x{~ \str_if_eq:nnTF {#3} {??} {default} {~normal} ~
1803       rule~ \__hook_label_pair:nn {#1} {#2}:~
1804     \use:c { g__hook_#3_rule_ \__hook_label_pair:nn {#1} {#2} _tl } ~

```

¹²This also has the advantage that the result of the sorting doesn't change, as it might otherwise do (for unrelated chunks) if we aren't careful.

```

1805         found}
1806     }

(End of definition for \__hook_msg_pair_found:nnn.)

```

__hook_debug_label_data:N

```

1807 \cs_new_protected:Npn \__hook_debug_label_data:N #1 {
1808   \iow_term:x{Code~ labels~ for~ sorting:}
1809   \iow_term:x{~ \seq_use:Nnnn\l__hook_labels_seq {~and~}{,~}{~and~} }
1810   \iow_term:x{^^J Data~ structure~ for~ label~ rules:}
1811   \prop_map_inline:Nn #1
1812     {
1813       \iow_term:x{~ ##1~ =~ \tl_use:c{ \__hook_tl_csname:n {##1} }~ ->~
1814         \seq_use:cnnn{ \__hook_seq_csname:n {##1} }{~>~}{~>~}{~>~}
1815     }
1816   }
1817   \iow_term:x{}
1818 }

(End of definition for \__hook_debug_label_data:N.)

```

\hook_show:n This writes out information about the hook given in its argument onto the .log file and the terminal, if \show_hook:n is used. Internally both share the same structure, except that at the end, \hook_show:n triggers T_EX's prompt.

\hook_log:n

__hook_log_line:x

__hook_log_line_indent:x

__hook_log:nN

```

1819 \cs_new_protected:Npn \hook_log:n #1
1820   {
1821     \cs_set_eq:NN \__hook_log_cmd:x \iow_log:x
1822     \__hook_normalize_hook_args:Nn \__hook_log:nN {#1} \tl_log:x
1823   }
1824 \cs_new_protected:Npn \hook_show:n #1
1825   {
1826     \cs_set_eq:NN \__hook_log_cmd:x \iow_term:x
1827     \__hook_normalize_hook_args:Nn \__hook_log:nN {#1} \tl_show:x
1828   }
1829 \cs_new_protected:Npn \__hook_log_line:x #1
1830   { \__hook_log_cmd:x { >~#1 } }
1831 \cs_new_protected:Npn \__hook_log_line_indent:x #1
1832   { \__hook_log_cmd:x { >~\@spaces #1 } }
1833 <latexrelease> \IncludeInRelease{2023/06/01}{\__hook_log:nN}
1834 <latexrelease> {Hooks-with-args}
1835 \cs_new_protected:Npn \__hook_log:nN #1 #2
1836   {
1837     \__hook_if_deprecated_generic:nT {#1}
1838     {
1839       \__hook_deprecated_generic_warn:n {#1}
1840       \__hook_do_deprecated_generic:Nn \__hook_log:nN {#1} #2
1841       \exp_after:wN \use_none:nnnnnnnn \use_none:nnnnn
1842     }
1843     \__hook_preamble_hook:n {#1}
1844     \__hook_log_cmd:x
1845     {
1846       ^^J ->~The~
1847       \__hook_if_generic:nT {#1} { generic~ }
1848       hook~'#1'

```

```

1849     \__hook_if_disabled:nF {#1}
1850     {
1851         \exp_args:Nne \__hook_print_args:nn {#1}
1852         {
1853             \int_eval:n
1854             { \str_count:e { \__hook_parameter:n {#1} } / 3 }
1855         }
1856     }
1857     :
1858 }

1859 \__hook_if_usable:nF {#1}
1860 { \__hook_log_line:x { The~hook~is~not~declared. } }
1861 \__hook_if_disabled:nT {#1}
1862 { \__hook_log_line:x { The~hook~is~disabled. } }
1863 \hook_if_empty:nTF {#1}
1864 { #2 { The~hook~is~empty } }
1865 {
1866     \__hook_log_line:x { Code~chunks: }
1867     \bool_lazy_or:nnTF
1868     { ! \prop_if_exist_p:c { g__hook_#1_code_prop } }
1869     { \prop_if_empty_p:c { g__hook_#1_code_prop } }
1870     { \__hook_log_line_indent:x { --- } }
1871     {
1872         \prop_map_inline:cn { g__hook_#1_code_prop }
1873         {
1874             \exp_after:wN \cs_set:Npn \exp_after:wN \__hook_tmp:w
1875             \c__hook_nine_parameters_tl {##2}
1876             \__hook_log_line_indent:x
1877             { ##1~>~\cs_replacement_spec:N \__hook_tmp:w }
1878         }
1879     }

```

If there is code in the top-level token list, print it:

```

1880     \__hook_log_line:x
1881     {
1882         Document-level~(top-level)~code
1883         \__hook_if_usable:nT {#1}
1884         { ~(executed~\__hook_if_reversed:nTF {#1} {first} {last} ) } :
1885     }
1886     \__hook_log_line_indent:x
1887     {
1888         \__hook_cs_if_empty:cTF { __hook_toplevel~#1 }
1889         { --- }
1890         { -> ~ \cs_replacement_spec:c { __hook_toplevel~#1 } }
1891     }

1892     \__hook_log_line:x { Extra-code-for-next-invocation: }
1893     \__hook_log_line_indent:x
1894     {
1895         \__hook_cs_if_empty:cTF { __hook_next~#1 }
1896         { --- }

```

If the token list is not empty we want to display it but without the first tokens (the code to clear itself) so we call a helper command to get rid of them.

```

1897     {
1898     -> ~ \exp_last_unbraced:Nf \__hook_log_next_code:w
1899         { \cs_replacement_spec:c { __hook_next~#1 } }
1900     }
1901 }

```

Loop through the rules in a hook and for every rule found, print it. If no rule is there, print ---. The boolean \l__hook_tmpa_bool here indicates if the hook has no rules.

```

1902 \__hook_log_line:x { Rules: }
1903 \bool_set_true:N \l__hook_tmpa_bool
1904 \__hook_list_rules:nn {#1}
1905 {
1906     \bool_set_false:N \l__hook_tmpa_bool
1907     \__hook_log_line_indent:x
1908     {
1909         ##2~ with~
1910         \str_if_eq:nnT {##3} {??} { default~ }
1911         relation~ ##1
1912     }
1913 }
1914 \bool_if:NT \l__hook_tmpa_bool
1915 { \__hook_log_line_indent:x { --- } }

```

When the hook is declared (that is, the sorting algorithm is applied to that hook) and not empty

```

1916 \bool_lazy_and:nnTF
1917 { \__hook_if_usable_p:n {#1} }
1918 { ! \hook_if_empty_p:n {#1} }
1919 {
1920     \__hook_log_line:x
1921     {
1922         Execution~order
1923         \bool_if:NTF \l__hook_tmpa_bool
1924         { \__hook_if_reversed:nT {#1} { ~(after~reversal) } }
1925         { ~(after~
1926             \__hook_if_reversed:nT {#1} { reversal~and~ }
1927             applying~rules)
1928         } :
1929     }
1930     #2 % \tl_show:n
1931     {
1932         \@spaces
1933         \clist_if_empty:cTF { g__hook_#1_labels_clist }
1934         { --- }
1935         { \clist_use:cn { g__hook_#1_labels_clist } { ,~ } }
1936     }
1937 }
1938 {
1939     \__hook_log_line:x { Execution~order: }
1940     #2
1941     {
1942         \@spaces Not~set~because~the~hook~ \__hook_if_usable:nTF {#1}
1943         { code~pool~is~empty }
1944         { is~\__hook_if_disabled:nTF {#1} {disabled} {undeclared} }

```

```

1945     }
1946   }
1947 }
1948 }
1949 <latexrelease> \EndIncludeInRelease
1950 %
1951 <latexrelease> \IncludeInRelease{2020/10/01}{\__hook_log:nN}
1952 <latexrelease> {Hooks-with-args}
1953 <latexrelease> \cs_new_protected:Npn \__hook_log:nN #1 #2
1954 <latexrelease> {
1955 <latexrelease>   \__hook_if_deprecated_generic:nT {#1}
1956 <latexrelease>   {
1957 <latexrelease>     \__hook_deprecated_generic_warn:n {#1}
1958 <latexrelease>     \__hook_do_deprecated_generic:Nn \__hook_log:nN {#1} #2
1959 <latexrelease>     \exp_after:wN \use_none:n \use_none:n \use_none:n
1960 <latexrelease>   }
1961 <latexrelease>   \__hook_preamble_hook:n {#1}
1962 <latexrelease>   \__hook_log_cmd:x
1963 <latexrelease>   { ^^J ->~The~ \__hook_if_generic:nT
1964 <latexrelease>     {#1} { generic~ } hook~'#1': }
1965 <latexrelease>   \__hook_if_usable:nF {#1}
1966 <latexrelease>   { \__hook_log_line:x { The~hook~is~not~declared. } }
1967 <latexrelease>   \__hook_if_disabled:nT {#1}
1968 <latexrelease>   { \__hook_log_line:x { The~hook~is~disabled. } }
1969 <latexrelease>   \hook_if_empty:nTF {#1}
1970 <latexrelease>   { #2 { The~hook~is~empty } }
1971 <latexrelease>   {
1972 <latexrelease>     \__hook_log_line:x { Code~chunks: }
1973 <latexrelease>     \prop_if_empty:cTF { g__hook_#1_code_prop }
1974 <latexrelease>     { \__hook_log_line_indent:x { --- } }
1975 <latexrelease>     {
1976 <latexrelease>       \prop_map_inline:cn { g__hook_#1_code_prop }
1977 <latexrelease>       { \__hook_log_line_indent:x
1978 <latexrelease>         { ##1-->~\tl_to_str:n {##2} } }
1979 <latexrelease>     }
1980 <latexrelease>     \__hook_log_line:x
1981 <latexrelease>     {
1982 <latexrelease>       Document-level~(top-level)~code
1983 <latexrelease>       \__hook_if_usable:nT {#1}
1984 <latexrelease>       { ~(executed~
1985 <latexrelease>         \__hook_if_reversed:nTF {#1} {first} {last} ) } :
1986 <latexrelease>     }
1987 <latexrelease>     \__hook_log_line_indent:x
1988 <latexrelease>     {
1989 <latexrelease>       \tl_if_empty:cTF { __hook_toplevel~#1 }
1990 <latexrelease>       { --- }
1991 <latexrelease>       { -> ~ \exp_args:Nv \tl_to_str:n
1992 <latexrelease>         { __hook_toplevel~#1 } }
1993 <latexrelease>     }
1994 <latexrelease>     \__hook_log_line:x { Extra~code~for~next~invocation: }
1995 <latexrelease>     \__hook_log_line_indent:x
1996 <latexrelease>     {
1997 <latexrelease>       \tl_if_empty:cTF { __hook_next~#1 }
1998 <latexrelease>       { --- }

```

```

1999 <latexrelease>         { ->~ \exp_args:Nv \_hook_log_next_code:n
2000 <latexrelease>         { \_hook_next~#1 } }
2001 <latexrelease>     }
2002 <latexrelease>     \_hook_log_line:x { Rules: }
2003 <latexrelease>     \bool_set_true:N \l__hook_tmpa_bool
2004 <latexrelease>     \_hook_list_rules:nn {#1}
2005 <latexrelease>     {
2006 <latexrelease>         \bool_set_false:N \l__hook_tmpa_bool
2007 <latexrelease>         \_hook_log_line_indent:x
2008 <latexrelease>         {
2009 <latexrelease>             ##2~ with~
2010 <latexrelease>             \str_if_eq:nnT {##3} {??} { default~ }
2011 <latexrelease>             relation~ ##1
2012 <latexrelease>         }
2013 <latexrelease>     }
2014 <latexrelease>     \bool_if:NT \l__hook_tmpa_bool
2015 <latexrelease>     { \_hook_log_line_indent:x { --- } }
2016 <latexrelease>     \bool_lazy_and:nnTF
2017 <latexrelease>     { \_hook_if_usable_p:n {#1} }
2018 <latexrelease>     { ! \hook_if_empty_p:n {#1} }
2019 <latexrelease>     {
2020 <latexrelease>         \_hook_log_line:x
2021 <latexrelease>         {
2022 <latexrelease>             Execution~order
2023 <latexrelease>             \bool_if:NTF \l__hook_tmpa_bool
2024 <latexrelease>             { \_hook_if_reversed:nT
2025 <latexrelease>               {#1}{ ~(after~reversal) } }
2026 <latexrelease>             { ~(after~
2027 <latexrelease>               \_hook_if_reversed:nT {#1} { reversal~and~
2028 <latexrelease>               applying~rules)
2029 <latexrelease>             } :
2030 <latexrelease>         }
2031 <latexrelease>     #2 % \tl_show:n
2032 <latexrelease>     {
2033 <latexrelease>         \@spaces
2034 <latexrelease>         \clist_if_empty:cTF { g__hook_#1_labels_clist }
2035 <latexrelease>         { --- }
2036 <latexrelease>         { \clist_use:cn
2037 <latexrelease>           { g__hook_#1_labels_clist } { ,~ } }
2038 <latexrelease>     }
2039 <latexrelease>     }
2040 <latexrelease>     {
2041 <latexrelease>         \_hook_log_line:x { Execution~order: }
2042 <latexrelease>     #2
2043 <latexrelease>     {
2044 <latexrelease>         \@spaces Not~set~because~the~hook~
2045 <latexrelease>         \_hook_if_usable:nTF {#1}
2046 <latexrelease>         { code~pool~is~empty }
2047 <latexrelease>         { is~\_hook_if_disabled:nTF
2048 <latexrelease>           {#1} {disabled} {undeclared} }
2049 <latexrelease>     }
2050 <latexrelease>     }
2051 <latexrelease>     }
2052 <latexrelease>     }

```

```
2053 <latexrelease>\EndIncludeInRelease
```

To display the code for next invocation only (i.e., from `\AddToHookNext` we have to remove the string `_hook_clear_next:n{<hook>}`, so the simplest is to use a macro delimited by a }₁2.

```
2054 <latexrelease>\IncludeInRelease{2023/06/01}{\_hook_log_next_code:n}
2055 <latexrelease> {Hooks-with-args}
\_hook_log_next_code:n 2056 \exp_last_unbraced:NNNNo
2057 \cs_new:Npn \_hook_log_next_code:w #1 \c_right_brace_str { }
2058 <latexrelease>\EndIncludeInRelease
2059 <latexrelease>\IncludeInRelease{2020/10/01}{\_hook_log_next_code:n}
2060 <latexrelease> {Hooks-with-args}
2061 <latexrelease>\cs_gset:Npn \_hook_log_next_code:n #1
2062 <latexrelease> { \exp_args:No \tl_to_str:n { \use_none:nn #1 } }
2063 <latexrelease>\EndIncludeInRelease
```

Pretty-prints the number of arguments of a hook.

```
2064 \cs_new:Npn \_hook_print_args:nn #1 #2
2065 {
2066   \int_compare:nNnT {#2} > { 0 }
2067   {
\_hook_print_args:n 2068     \_hook_if_declared:nT {#1} { \use_none:nnn }
2069     \_hook_if_cmd_hook:nT {#1}
2070     { \use_i:nnn { ~ (unknown ~ ) } }
2071     \use:n { ~ (#2 ~ ) }
2072     argument \int_compare:nNnT {#2} > { 1 } { s } )
2073   }
2074 }
```

(End of definition for `\hook_show:n` and others. These functions are documented on page 214.)

```
\_hook_list_rules:nn This macro takes a <hook> and an <inline function> and loops through each pair of
\_hook_list_one_rule:nnn <labels> in the <hook>, and if there is a relation between this pair of <labels>, the
\_hook_list_if_rule_exists:nnnF <inline function> is executed with #1 = <relation>, #2 = <label1>|<label2>, and
#3 = <hook> (the latter may be the argument #1 to \_hook_list_rules:nn, or ?? if it
is a default rule).
```

```
2075 \cs_new_protected:Npn \_hook_list_rules:nn #1 #2
2076 {
2077   \prop_if_exist:cT { g__hook_#1_code_prop }
2078   {
2079     \cs_set_protected:Npn \_hook_tmp:w ##1 ##2 ##3 {#2}
2080     \prop_map_inline:cn { g__hook_#1_code_prop }
2081     {
2082       \prop_map_inline:cn { g__hook_#1_code_prop }
2083       {
2084         \_hook_if_label_case:nnnnn {##1} {####1}
2085         { \prop_map_break: }
2086         { \_hook_list_one_rule:nnn {##1} {####1} }
2087         { \_hook_list_one_rule:nnn {####1} {##1} }
2088         {#1}
2089       }
2090     }
2091   }
2092 }
```

These two are quite similar to `__hook_apply_label_pair:nnn` and `__hook_label_if_exist_apply:nnnF`, respectively, but rather than applying the rule, they pass it to the *(inline function)*.

```

2093 \cs_new_protected:Npn \__hook_list_one_rule:nnn #1#2#3
2094 {
2095   \__hook_list_if_rule_exists:nnnF {#1} {#2} {#3}
2096   { \__hook_list_if_rule_exists:nnnF {#1} {#2} { ?? } { } }
2097 }
2098 \cs_new_protected:Npn \__hook_list_if_rule_exists:nnnF #1#2#3
2099 {
2100   \if_cs_exist:w g__hook_ #3 _rule_ #1 | #2 _tl \cs_end:
2101   \exp_args:Nv \__hook_tmp:w
2102   { g__hook_ #3 _rule_ #1 | #2 _tl } { #1 | #2 } {#3}
2103   \exp_after:wN \use_none:nn
2104   \fi:
2105   \use:n
2106 }

```

(End of definition for `__hook_list_rules:nn`, `__hook_list_one_rule:nnn`, and `__hook_list_if_rule_exists:nnnF`.)

`__hook_debug_print_rules:n` A shorthand for debugging that prints similar to `\prop_show:N`.

```

2107 \cs_new_protected:Npn \__hook_debug_print_rules:n #1
2108 {
2109   \iow_term:n { The~hook~#1~contains~the~rules: }
2110   \cs_set_protected:Npn \__hook_tmp:w ##1
2111   {
2112     \__hook_list_rules:nn {#1}
2113     {
2114       \iow_term:x
2115       {
2116         > ##1 {####2} ##1 => ##1 {####1}
2117         \str_if_eq:nnT {####3} {??} { ~(default) }
2118       }
2119     }
2120   }
2121   \exp_args:No \__hook_tmp:w { \use:nn { ~ } { ~ } }
2122 }

```

(End of definition for `__hook_debug_print_rules:n`.)

4.8 Specifying code for next invocation

`\hook_gput_next_code:nn`

```

2123 <latexrelease>\IncludeInRelease{2023/06/01}{\hook_gput_next_code:nn}
2124 <latexrelease>           {Hooks-with-args}
2125 \cs_new_protected:Npn \hook_gput_next_code:nn #1 #2
2126 {
2127   \__hook_replacing_args_false:
2128   \__hook_normalize_hook_args:Nn \__hook_gput_next_code:nn {#1} {#2}
2129
2130   \__hook_debug:n{\iow_term:x{[lthooks]~ Add~ to~
2131                   \__hook_if_usable:nF {#1} { undeclared~ }
2132                   hook~ '#1'~ ( next~ invocation~ only )

```



```

2132         \on@line
2133         ^^J [lthooks] \@spaces
2134         <-- \tl_to_str:n{#2}
2135     }
2136 }
2137 \__hook_replacing_args_reset:
2138 }
2139 \cs_new_protected:Npn \hook_gput_next_code_with_args:nn #1 #2
2140 {
2141     \__hook_replacing_args_true:
2142     \__hook_normalize_hook_args:Nn \__hook_gput_next_code:nn {#1} {#2}
2143     \__hook_debug:n{\iow_term:x{[lthooks]~ Add~ to~
2144         \__hook_if_usable:nF {#1} { undeclared~ }
2145         hook~ '#1'~ ( next~ invocation~ only )
2146         \on@line
2147         ^^J [lthooks] \@spaces
2148         <-- \tl_to_str:n{#2}
2149     }
2150 }
2151 \__hook_replacing_args_reset:
2152 }
2153 <latexrelease>\EndIncludeInRelease
2154 <latexrelease>\IncludeInRelease{2020/10/01}{\hook_gput_next_code:nn}
2155 <latexrelease>           {Hooks-with-args}
2156 <latexrelease>\cs_gset_protected:Npn \hook_gput_next_code:nn #1
2157 <latexrelease> { \__hook_normalize_hook_args:Nn
2158 <latexrelease>         \__hook_gput_next_code:nn {#1} }
2159 <latexrelease>\cs_gset_protected:Npn \hook_gput_next_code_with_args:nn #1 #2 { }
2160 <latexrelease>\EndIncludeInRelease

```

(End of definition for `\hook_gput_next_code:nn`. This function is documented on page 213.)

`__hook_gput_next_code:nn`

```

2161 \cs_new_protected:Npn \__hook_gput_next_code:nn #1 #2
2162 {
2163     \__hook_if_disabled:nTF {#1}
2164     { \msg_error:nnn { hooks } { hook-disabled } {#1} }
2165     {
2166         \__hook_if_structure_exist:nTF {#1}
2167         { \__hook_gput_next_do:nn }
2168         { \__hook_try_declaring_generic_next_hook:nn }
2169         {#1} {#2}
2170     }
2171 }

```

(End of definition for `__hook_gput_next_code:nn`.)

`__hook_gput_next_do:nn` Start by sanity-checking with `__hook_chk_args_allowed:nn`. Then check if the “next code” token list is empty: if so we need to add a `\tl_gclear:c` to clear it, so the code lasts for one usage only. The token list is cleared early so that nested usages don’t get lost. `\tl_gclear:c` is used instead of `\tl_gclear:N` in case the hook is used in an expansion-only context, so the token list doesn’t expand before `\tl_gclear:N`: that would make

an infinite loop. Also in case the main code token list is empty, the hook code has to be updated to add the next execution token list.

```

2172 <latexrelease>\IncludeInRelease{2023/06/01}{\__hook_gput_next_do:nn}
2173 <latexrelease>          {Hooks-with-args}
2174 \cs_new_protected:Npn \__hook_gput_next_do:nn #1
2175   {
2176     \__hook_init_structure:n {#1}
2177     \__hook_chk_args_allowed:nn {#1} { AddToHookNext }
2178     \__hook_cs_if_empty:cT { __hook~#1 }
2179     { \__hook_update_hook_code:n {#1} }
2180     \__hook_cs_if_empty:cT { __hook_next~#1 }
2181     { \__hook_next_gset:nn {#1} { \__hook_clear_next:n {#1} } }
2182     \__hook_cs_gput_right:nnn { _next } {#1}
2183   }
2184 <latexrelease>\EndIncludeInRelease
2185 <latexrelease>\IncludeInRelease{2020/10/01}{\__hook_gput_next_do:nn}
2186 <latexrelease>          {Hooks-with-args}
2187 <latexrelease>\cs_gset_protected:Npn \__hook_gput_next_do:nn #1
2188 <latexrelease>   {
2189 <latexrelease>     \exp_args:Nc \__hook_gput_next_do:Nnn
2190 <latexrelease>       { __hook_next~#1 } {#1}
2191 <latexrelease>   }
2192 <latexrelease>\cs_gset_protected:Npn \__hook_gput_next_do:Nnn #1 #2
2193 <latexrelease>   {
2194 <latexrelease>     \tl_if_empty:cT { __hook~#2 }
2195 <latexrelease>       { \__hook_update_hook_code:n {#2} }
2196 <latexrelease>     \tl_if_empty:NT #1
2197 <latexrelease>       { \__hook_tl_gset:Nn #1 { \__hook_clear_next:n {#2} } }
2198 <latexrelease>     \__hook_tl_gput_right:Nn #1
2199 <latexrelease>   }
2200 <latexrelease>\EndIncludeInRelease

```

(End of definition for __hook_gput_next_do:nn.)

\hook_gclear_next_code:n Discard anything set up for next invocation of the hook.

```

2201 \cs_new_protected:Npn \hook_gclear_next_code:n #1
2202   { \__hook_normalize_hook_args:Nn \__hook_clear_next:n {#1} }

```

(End of definition for \hook_gclear_next_code:n. This function is documented on page 213.)

__hook_clear_next:n

```

2203 <latexrelease>\IncludeInRelease{2023/06/01}{\__hook_clear_next:n}
2204 <latexrelease>          {Hooks-with-args}
2205 \cs_new_protected:Npn \__hook_clear_next:n #1
2206   { \__hook_next_gset:nn {#1} { } }
2207 <latexrelease>\EndIncludeInRelease
2208 <latexrelease>\IncludeInRelease{2020/10/01}{\__hook_clear_next:n}
2209 <latexrelease>          {Hooks-with-args}
2210 <latexrelease>\cs_gset_protected:Npn \__hook_clear_next:n #1
2211 <latexrelease>   { \cs_gset_eq:cN { __hook_next~#1 } \c_empty_tl }
2212 <latexrelease>\EndIncludeInRelease

```

(End of definition for __hook_clear_next:n.)

4.9 Using the hook

`\hook_use:n` as defined here is used in the preamble, where hooks aren't initialized by default. `__hook_use_initialized:n` is also defined, which is the non-`\protected` version for use within the document. Their definition is identical, except for the `__hook_preamble_hook:n` (which wouldn't hurt in the expandable version, but it would be an unnecessary extra expansion).

`__hook_use_initialized:n` holds the expandable definition while in the preamble. `__hook_preamble_hook:n` initializes the hook in the preamble, and is redefined to `\use_none:n` at `\begin{document}`.

Both versions do the same thing internally: they check that the hook exists as given, and if so they use it as quickly as possible.

At `\begin{document}`, all hooks are initialized, and any change in them causes an update, so `\hook_use:n` can be made expandable. This one is better not protected so that it can expand into nothing if containing no code. Also important in case of generic hooks that we do not generate a `\relax` as a side effect of checking for a csname. In contrast to the \TeX low-level `\csname ... \endcsname` construct `\tl_if_exist:c` is careful to avoid this.

```

2213 <latexrelease>\IncludeInRelease{2023/06/01}{\hook_use:n}
2214 <latexrelease>           {Hooks-with-args}
2215 \cs_new_protected:Npn \hook_use:n #1
2216   {
2217     \__hook_preamble_hook:n {#1}
2218     \__hook_use_initialized:n {#1}
2219   }
2220 \cs_new:Npn \__hook_use_initialized:n #1
2221   {
2222     \if_cs_exist:w __hook~#1 \cs_end:
2223       \cs:w __hook~#1 \use_i:nn
2224     \fi:
2225     \use_none:n
2226     \cs_end:
2227   }
2228 \cs_new_protected:Npn \__hook_preamble_hook:n #1
2229   {
2230     \if_cs_exist:w __hook~#1 \cs_end:
2231       \__hook_initialize_hook_code:n {#1}
2232     \fi:
2233   }
2234 <latexrelease>\EndIncludeInRelease
2235 <latexrelease>\IncludeInRelease{2021/11/15}{\hook_use:n}
2236 <latexrelease>           {Standardize-generic-hook-names}
2237 <latexrelease>\cs_new_protected:Npn \hook_use:n #1
2238 <latexrelease>   {
2239 <latexrelease>     \tl_if_exist:cT { __hook~#1 }
2240 <latexrelease>     {
2241 <latexrelease>       \__hook_preamble_hook:n {#1}
2242 <latexrelease>       \cs:w __hook~#1 \cs_end:
2243 <latexrelease>     }
2244 <latexrelease>   }
2245 <latexrelease>\cs_new:Npn \__hook_use_initialized:n #1
2246 <latexrelease>   {

```

```

2247 <latexrelease> \if_cs_exist:w __hook~#1 \cs_end:
2248 <latexrelease> \cs:w __hook~#1 \exp_after:wN \cs_end:
2249 <latexrelease> \fi:
2250 <latexrelease> }
2251 <latexrelease> \cs_new_protected:Npn \__hook_preamble_hook:n #1
2252 <latexrelease> { \__hook_initialize_hook_code:n {#1} }
2253 <latexrelease> \cs_new:Npn \hook_use:nnw #1 { }
2254 <latexrelease> \EndIncludeInRelease

2255 <latexrelease> \IncludeInRelease{2020/10/01}{\hook_use:n}
2256 <latexrelease> {Standardize-generic-hook-names}
2257 <latexrelease> \cs_new_protected:Npn \hook_use:n #1
2258 <latexrelease> {
2259 <latexrelease> \tl_if_exist:cTF { __hook~#1 }
2260 <latexrelease> {
2261 <latexrelease> \__hook_preamble_hook:n {#1}
2262 <latexrelease> \cs:w __hook~#1 \cs_end:
2263 <latexrelease> }
2264 <latexrelease> { \__hook_use:wn #1 / \s__hook_mark {#1} }
2265 <latexrelease> }
2266 <latexrelease> \cs_new:Npn \__hook_use_initialized:n #1
2267 <latexrelease> {
2268 <latexrelease> \if_cs_exist:w __hook~#1 \cs_end:
2269 <latexrelease> \else:
2270 <latexrelease> \__hook_use_undefined:w
2271 <latexrelease> \fi:
2272 <latexrelease> \cs:w __hook~#1 \__hook_use_end:
2273 <latexrelease> }
2274 <latexrelease> \cs_new:Npn \__hook_use_undefined:w
2275 <latexrelease> #1 #2 __hook~#3 \__hook_use_end:
2276 <latexrelease> {
2277 <latexrelease> #1 % fi
2278 <latexrelease> \__hook_use:wn #3 / \s__hook_mark {#3}
2279 <latexrelease> }
2280 <latexrelease> \cs_new_protected:Npn \__hook_preamble_hook:n #1
2281 <latexrelease> { \__hook_initialize_hook_code:n {#1} }
2282 <latexrelease> \cs_new_eq:NN \__hook_use_end: \cs_end:
2283 <latexrelease> \cs_new:Npn \hook_use:nnw #1 { }
2284 <latexrelease> \EndIncludeInRelease

```

(End of definition for \hook_use:n, __hook_use_initialized:n, and __hook_preamble_hook:n. This function is documented on page 212.)

\hook_use:nnw

__hook_use_initialized:nnw

```

2285 <latexrelease> \IncludeInRelease{2023/06/01}{\hook_use:nnw}
2286 <latexrelease> {Hooks-with-args}
2287 \cs_new_protected:Npn \hook_use:nnw #1
2288 {
2289 \__hook_preamble_hook:n {#1}
2290 \__hook_use_initialized:nnw {#1}
2291 }
2292 \cs_new:Npn \__hook_use_initialized:nnw #1 #2
2293 {
2294 \cs:w
2295 \if_cs_exist:w __hook~#1 \cs_end:

```

```

2296     __hook~#1
2297     \else:
2298         use_none: \prg_replicate:nn {#2} { n }
2299     \fi:
2300     \cs_end:
2301 }
2302 \<latexrelease>\EndIncludeInRelease
2303 \<latexrelease>\IncludeInRelease{2020/10/01}{\hook_use:nnw}
2304 \<latexrelease>           {Hooks-with-args}
2305 \<latexrelease>\cs_gset:Npn \hook_use:nnw #1 #2
2306 \<latexrelease> { \use:c { use_none: \prg_replicate:nn {#2} { n } } }
2307 \<latexrelease>\EndIncludeInRelease

```

(End of definition for \hook_use:nnw and __hook_use_initialized:nnw. This function is documented on page 212.)

__hook_post_initialization_defs:

```

2308 \<latexrelease>\IncludeInRelease{2023/06/01}{\__hook_post_initialization_defs:}
2309 \<latexrelease>           {Hooks-with-args}
2310 \cs_new_protected:Npn \__hook_post_initialization_defs:
2311 {
2312     \cs_gset_eq:NN \hook_use:n \__hook_use_initialized:n
2313     \cs_gset_eq:NN \hook_use:nnw \__hook_use_initialized:nnw
2314     \cs_gset_eq:NN \__hook_preamble_hook:n \use_none:n
2315     \cs_gset_eq:NN \__hook_post_initialization_defs: \prg_do_nothing:
2316 }
2317 \<latexrelease>\EndIncludeInRelease
2318 \<latexrelease>\IncludeInRelease{2020/10/01}{\__hook_post_initialization_defs:}
2319 \<latexrelease>           {Hooks-with-args}
2320 \<latexrelease>\cs_undefine:N \__hook_post_initialization_defs:
2321 \<latexrelease>\EndIncludeInRelease

```

(End of definition for __hook_post_initialization_defs:.)

__hook_use:wn __hook_use:wn does a quick check to test if the current hook is a file hook: those need a special treatment. If it is not, the hook does not exist. If it is, then __hook_try_file_hook:n __hook_if_usable_use:n try_file_hook:n is called, and checks that the current hook is a file-specific hook using __hook_if_file_hook:wTF. If it's not, then it's a generic file/ hook and is used if it exist.

If it is a file-specific hook, it passes through the same normalization as during declaration, and then it is used if defined. __hook_if_usable_use:n checks if the hook exist, and calls __hook_preamble_hook:n if so, then uses the hook.

```

2322 \<latexrelease>\IncludeInRelease{2021/11/15}{\__hook_use:wn}
2323 \<latexrelease>           {Standardize-generic-hook-names}
2324 \<latexrelease>\EndIncludeInRelease
2325 \<latexrelease>\IncludeInRelease{2020/10/01}{\__hook_use:wn}
2326 \<latexrelease>           {Standardize-generic-hook-names}
2327 \<latexrelease>\cs_new:Npn \__hook_use:wn #1 / #2 \s__hook_mark #3
2328 \<latexrelease> {
2329     \str_if_eq:nnTF {#1} { file }
2330     { \__hook_try_file_hook:n {#3} }
2331     { } % Hook doesn't exist
2332 \<latexrelease> }

```

```

2333 <latexrelease> \cs_new_protected:Npn \__hook_try_file_hook:n #1
2334 <latexrelease> {
2335 <latexrelease>   \__hook_if_file_hook:wTF #1 / / \s__hook_mark
2336 <latexrelease>   {
2337 <latexrelease>     \exp_args:Ne \__hook_if_usable_use:n
2338 <latexrelease>     { \exp_args:Ne \__hook_file_hook_normalize:n {#1} }
2339 <latexrelease>   }
2340 <latexrelease>   { \__hook_if_usable_use:n {#1} }
2341 <latexrelease>     % file/ generic hook (e.g. file/before)
2342 <latexrelease> }

2343 <latexrelease> \cs_new_protected:Npn \__hook_if_usable_use:n #1
2344 <latexrelease> {
2345 <latexrelease>   \tl_if_exist:cT { __hook~#1 }
2346 <latexrelease>   {
2347 <latexrelease>     \__hook_preamble_hook:n {#1}
2348 <latexrelease>     \cs:w __hook~#1 \cs_end:
2349 <latexrelease>   }
2350 <latexrelease> }
2351 <latexrelease> \EndIncludeInRelease

```

(End of definition for __hook_use:wn, __hook_try_file_hook:n, and __hook_if_usable_use:n.)

`\hook_use_once:n` For hooks that can and should be used only once we have a special use command that further inhibits the hook from getting more code added to it. This has the effect that any further code added to the hook is executed immediately rather than stored in the hook.

The code needs some gymnastics to prevent space trimming from the hook name, since `\hook_use:n` and `\hook_use_once:n` are documented to not trim spaces.

```

2352 <latexrelease> \IncludeInRelease{2023/06/01}{\hook_use_once:nnw}
2353 <latexrelease>   {Hooks-with-args}
2354 \cs_new_protected:Npn \hook_use_once:n #1
2355 {
2356   \__hook_if_execute_immediately:nF {#1}
2357   { \__hook_normalize_hook_args:Nn \__hook_use_once:nn
2358     { \use:n {#1} } { 0 } }
2359 }
2360 \cs_new_protected:Npn \hook_use_once:nnw #1 #2
2361 {
2362   \__hook_if_execute_immediately:nF {#1}
2363   { \__hook_normalize_hook_args:Nn \__hook_use_once:nn
2364     { \use:n {#1} } {#2} }
2365 }
2366 <latexrelease> \EndIncludeInRelease

```

(End of definition for `\hook_use_once:n` and `\hook_use_once:nnw`. These functions are documented on page 212.)

```

2367 <latexrelease> \IncludeInRelease{2020/10/01}{\hook_use_once:nnw}
2368 <latexrelease>   {Hooks-with-args}
2369 <latexrelease> \cs_gset_protected:Npn \hook_use_once:n #1
2370 <latexrelease> {
2371 <latexrelease>   \__hook_if_execute_immediately:nF {#1}
2372 <latexrelease>   { \__hook_normalize_hook_args:Nn \__hook_use_once:n
2373 <latexrelease>     { \use:n {#1} } } }

```

```

2374 <latexrelease> }
2375 <latexrelease> \cs_gset:Npn \hook_use_once:nnw #1 #2
2376 <latexrelease> { \use:c { use_none: \prg_replicate:nn {#2} { n } } }
2377 <latexrelease> \EndIncludeInRelease

```

`__hook_use_once:nn`

```

2378 <latexrelease> \IncludeInRelease{2023/06/01}{\__hook_use_once:nn}
2379 <latexrelease> {Hooks-with-args}
2380 \cs_new_protected:Npn \__hook_use_once:nn #1 #2
2381 {
2382   \__hook_preamble_hook:n {#1}
2383   \__hook_use_once_set:n {#1}

```

When a hook has arguments, the call to `__hook_use_initialized:n`, should be the very last thing to happen, otherwise the arguments grabbed will be wrong. So, to clean up after the hook we need to cheat a bit and sneak the cleanup code at the end of the hook, along with the next execution code.

```

2384   \__hook_replacing_args_false:
2385   \__hook_cs_gput_right:nnn { _next } {#1}
2386   { \__hook_use_once_clear:n {#1} }
2387   \__hook_replacing_args_reset:
2388   \__hook_if_usable:nTF {#1}
2389   { \__hook_use_initialized:n {#1} }
2390   {
2391     \int_compare:nNnT {#2} > { 0 }
2392     { \use:c { use_none: \prg_replicate:nn {#2} { n } } }
2393   }
2394 }

```

```

2395 <latexrelease> \EndIncludeInRelease
2396 %
2397 <latexrelease> \IncludeInRelease{2020/10/01}{\__hook_use_once:nn}
2398 <latexrelease> {Hooks-with-args}
2399 <latexrelease> \cs_gset_protected:Npn \__hook_use_once:n #1
2400 <latexrelease> {
2401 <latexrelease>   \__hook_preamble_hook:n {#1}
2402 <latexrelease>   \__hook_use_once_set:n {#1}
2403 <latexrelease>   \__hook_use_initialized:n {#1}
2404 <latexrelease>   \__hook_use_once_clear:n {#1}
2405 <latexrelease> }
2406 <latexrelease> \cs_undefine:N \__hook_use_once:nn
2407 <latexrelease> \EndIncludeInRelease

```

(End of definition for `__hook_use_once:nn`.)

`__hook_use_once_set:n` `__hook_use_once_set:n` is used before the actual hook code is executed so that any usage of `\AddToHook` inside the hook causes the code to execute immediately. Setting `\g__hook_<hook>_reversed_tl` to I prevents further code from being added to the hook. `__hook_use_once_clear:n` then clears the hook so that any further call to `\hook_use:n` or `\hook_use_once:n` will expand to nothing.

```

2408 <latexrelease> \IncludeInRelease{2023/06/01}{\__hook_use_once_clear:n}
2409 <latexrelease> {Hooks-with-args}
2410 \cs_new_protected:Npn \__hook_use_once_set:n #1
2411 { \__hook_tl_gset:cn { g__hook_#1_reversed_tl } { I } }
2412 \cs_new_protected:Npn \__hook_use_once_clear:n #1

```

```

2413 {
2414   \__hook_code_gset:nn {#1} { }
2415   \__hook_next_gset:nn {#1} { }
2416   \__hook_toplevel_gset:nn {#1} { }
2417   \prop_gclear_new:c { g__hook_#1_code_prop }
2418 }
2419 <latexrelease>\EndIncludeInRelease

2420 <latexrelease>\IncludeInRelease{2020/10/01}{\__hook_use_once_clear:n}
2421 <latexrelease>           {Hooks-with-args}
2422 <latexrelease>\cs_new_protected:Npn \__hook_use_once_clear:n #1
2423 <latexrelease> {
2424 <latexrelease>   \__hook_tl_gclear:c { __hook~#1 }
2425 <latexrelease>   \__hook_tl_gclear:c { __hook_next~#1 }
2426 <latexrelease>   \__hook_tl_gclear:c { __hook_toplevel~#1 }
2427 <latexrelease>   \prop_gclear_new:c { g__hook_#1_code_prop }
2428 <latexrelease> }
2429 <latexrelease>\EndIncludeInRelease

```

(End of definition for __hook_use_once_set:n and __hook_use_once_clear:n.)

__hook_if_execute_immediately_p:n To check whether the code being added should be executed immediately (that is, if
 __hook_if_execute_immediately:nTF the hook is a one-time hook), we check if \g__hook_⟨hook⟩_reversed_tl is I. The
 gymnastics around \if:w is there to allow the reversed token list to be empty.

```

2430 \prg_new_conditional:Npnn \__hook_if_execute_immediately:n #1 { T, F, TF }
2431 {
2432   \exp_after:wN \__hook_use_none_delimit_by_s_mark:w
2433   \if:w I
2434     \if_cs_exist:w g__hook_#1_reversed_tl \cs_end:
2435     \cs:w g__hook_#1_reversed_tl \exp_after:wN \cs_end:
2436     \fi:
2437     X
2438     \s__hook_mark \prg_return_true:
2439   \else:
2440     \s__hook_mark \prg_return_false:
2441   \fi:
2442 }

```

(End of definition for __hook_if_execute_immediately:nTF.)

4.10 Querying a hook

Simpler data types, like token lists, have three possible states; they can exist and be empty, exist and be non-empty, and they may not exist, in which case emptiness doesn't apply (though \tl_if_empty:N returns false in this case).

Hooks are a bit more complicated: they have several other states as discussed in 4.4.2. A hook may exist or not, and either way it may or may not be empty (even a hook that doesn't exist may be non-empty) or may be disabled.

A hook is said to be empty when no code was added to it, either to its permanent code pool, or to its “next” token list. The hook doesn't need to be declared to have code added to its code pool (it may happen that a package *A* defines a hook `foo`, but it's loaded after package *B*, which adds some code to that hook. In this case it is important that the code added by package *B* is remembered until package *A* is loaded).

All other states can only be queried with internal tests as the different states are irrelevant for package code.

`\hook_if_empty_p:n` Test if a hook is empty (that is, no code was added to that hook). A `\hook` being empty means that all three of its `\g__hook_<hook>_code_prop`, its `__hook_toplevel_<hook>` and its `__hook_next_<hook>` are empty.

```

2443 <latexrelease>\IncludeInRelease{2023/06/01}{\hook_if_empty:n}
2444 <latexrelease>          {Hooks-with-args}
2445 \prg_new_conditional:Npnn \hook_if_empty:n #1 { p , T , F , TF }
2446 {
2447   \if:w
2448     T
2449     \prop_if_exist:cT { g__hook_#1_code_prop }
2450     { \prop_if_empty:cF { g__hook_#1_code_prop } { F } }
2451     \__hook_cs_if_empty:cF { __hook_toplevel~#1 } { F }
2452     \__hook_cs_if_empty:cF { __hook_next~#1 } { F }
2453     T
2454     \prg_return_true:
2455   \else:
2456     \prg_return_false:
2457   \fi:
2458 }
2459 <latexrelease>\EndIncludeInRelease

2460 <latexrelease>\IncludeInRelease{2020/10/01}{\hook_if_empty:n}
2461 <latexrelease>          {Hooks-with-args}
2462 <latexrelease>\prg_new_conditional:Npnn \hook_if_empty:n #1 { p , T , F , TF }
2463 <latexrelease> {
2464 <latexrelease>   \__hook_if_structure_exist:nTF {#1}
2465 <latexrelease>   {
2466 <latexrelease>     \bool_lazy_and:nnTF
2467 <latexrelease>     { \prop_if_empty_p:c { g__hook_#1_code_prop } }
2468 <latexrelease>     {
2469 <latexrelease>       \bool_lazy_and_p:nn
2470 <latexrelease>       { \tl_if_empty_p:c { __hook_toplevel~#1 } }
2471 <latexrelease>       { \tl_if_empty_p:c { __hook_next~#1 } }
2472 <latexrelease>     }
2473 <latexrelease>     { \prg_return_true: }
2474 <latexrelease>     { \prg_return_false: }
2475 <latexrelease>   }
2476 <latexrelease>   { \prg_return_true: }
2477 <latexrelease> }
2478 <latexrelease>\EndIncludeInRelease

```

(End of definition for `\hook_if_empty:nTF`. This function is documented on page 214.)

`__hook_if_usable_p:n` A hook is usable if the token list that stores the sorted code for that hook, `__hook_<hook>`, exists. The property list `\g__hook_<hook>_code_prop` cannot be used here because often it is necessary to add code to a hook without knowing if such hook was already declared, or even if it will ever be (for example, in case the package that defines it isn't loaded).

```

2479 \prg_new_conditional:Npnn \__hook_if_usable:n #1 { p , T , F , TF }
2480 {
2481   \cs_if_exist:cTF { __hook~#1 }

```

```

2482     { \prg_return_true: }
2483     { \prg_return_false: }
2484   }

```

(End of definition for `_hook_if_usable:nTF`.)

`_hook_if_structure_exist_p:n` An internal check if the hook has already its basic internal structure set up with `_hook_init_structure:n`. This means that the hook was already used somehow (a code chunk or rule was added to it), but it still wasn't declared with `\hook_new:n`.

```

2485 \prg_new_conditional:Npnn \_hook_if_structure_exist:n #1 { p , T , F , TF }
2486   {
2487     \prop_if_exist:cTF { g__hook_#1_code_prop }
2488     { \prg_return_true: }
2489     { \prg_return_false: }
2490   }

```

(End of definition for `_hook_if_structure_exist:nTF`.)

`_hook_if_declared_p:n` Internal test to check if the hook was officially declared with `\hook_new:n` or a variant.

```

\_hook_if_declared:nTF 2491 \prg_new_conditional:Npnn \_hook_if_declared:n #1 { p , T , F , TF }
2492   {
2493     \tl_if_exist:cTF { g__hook_#1_declared_tl }
2494     { \prg_return_true: }
2495     { \prg_return_false: }
2496   }

```

(End of definition for `_hook_if_declared:nTF`.)

`_hook_if_reversed_p:n` An internal conditional that checks if a hook is reversed.

```

\_hook_if_reversed:nTF 2497 \prg_new_conditional:Npnn \_hook_if_reversed:n #1 { p , T , F , TF }
2498   {
2499     \exp_after:wN \_hook_use_none_delimit_by_s_mark:w
2500     \if:w - \cs:w g__hook_#1_reversed_tl \cs_end:
2501     \s__hook_mark \prg_return_true:
2502     \else:
2503     \s__hook_mark \prg_return_false:
2504     \fi:
2505   }

```

(End of definition for `_hook_if_reversed:nTF`.)

`_hook_if_generic_p:n` An internal conditional that checks if a name belongs to a generic hook. The deprecated version needs to check if #3 is empty to avoid returning true on file/before, for example.

```

\_hook_if_generic:nTF 2506 \prg_new_conditional:Npnn \_hook_if_generic:n #1 { T , TF }
\_hook_if_deprecated_generic_p:n 2507   { \_hook_if_generic:w #1 / / / \s__hook_mark }
\_hook_if_deprecated_generic:nTF 2508 \cs_new:Npn \_hook_if_generic:w #1 / #2 / #3 / #4 \s__hook_mark
2509   {
2510     \cs_if_exist:cTF { c__hook_generic_#1/#3_tl }
2511     { \prg_return_true: }
2512     { \prg_return_false: }
2513   }
2514 \prg_new_conditional:Npnn \_hook_if_deprecated_generic:n #1 { T , TF }
2515   { \_hook_if_deprecated_generic:w #1 / / / \s__hook_mark }
2516 \cs_new:Npn \_hook_if_deprecated_generic:w #1 / #2 / #3 / #4 \s__hook_mark
2517   {

```

```

2518 \cs_if_exist:cTF { c__hook_deprecated_#1/./#2_tl }
2519 {
2520   \tl_if_empty:nTF {#3}
2521     { \prg_return_false: }
2522     { \prg_return_true: }
2523 }
2524 { \prg_return_false: }
2525 }

```

(End of definition for `__hook_if_generic:nTF` and `__hook_if_deprecated_generic:nTF`.)

`__hook_if_cmd_hook_p:n` An internal conditional that checks if a given hook is a valid generic cmd hook.

```

\__hook_if_cmd_hook:nTF 2526 <latexrelease>\IncludeInRelease{2023/06/01}{\__hook_if_cmd_hook:n}
\__hook_if_cmd_hook_p:w 2527 <latexrelease> {Hooks-with-args}
\__hook_if_cmd_hook:wTF 2528 \prg_new_conditional:Npnn \__hook_if_cmd_hook:n #1 { T }
2529 { \__hook_if_cmd_hook:w #1 / / / \s__hook_mark }
2530 \cs_new:Npn \__hook_if_cmd_hook:w #1 / #2 / #3 / #4 \s__hook_mark
2531 {
2532   \if:w Y
2533     \str_if_eq:nnF {#1} { cmd } { N }
2534     \tl_if_exist:cF { c__hook_generic_#1/./#3_tl } { N }
2535     Y
2536     \prg_return_true:
2537   \else:
2538     \prg_return_false:
2539   \fi:
2540 }
2541 <latexrelease>\EndIncludeInRelease
2542 <latexrelease>\IncludeInRelease{2020/10/01}{\__hook_if_cmd_hook:n}
2543 <latexrelease> {Hooks-with-args}
2544 <latexrelease>\cs_undefine:N \__hook_if_cmd_hook:nTF
2545 <latexrelease>\EndIncludeInRelease

```

(End of definition for `__hook_if_cmd_hook:nTF` and `__hook_if_cmd_hook:wTF`.)

`__hook_if_generic_reversed_p:n` An internal conditional that checks if a name belongs to a generic reversed hook.

```

\__hook_if_generic_reversed:nTF 2546 \prg_new_conditional:Npnn \__hook_if_generic_reversed:n #1 { T }
2547 { \__hook_if_generic_reversed:w #1 / / / \scan_stop: }
2548 \cs_new:Npn \__hook_if_generic_reversed:w #1 / #2 / #3 / #4 \scan_stop:
2549 {
2550   \if_charcode:w - \cs:w c__hook_generic_#1/./#3_tl \cs_end:
2551     \prg_return_true:
2552   \else:
2553     \prg_return_false:
2554   \fi:
2555 }

```

(End of definition for `__hook_if_generic_reversed:nTF`.)

`__hook_if_replacing_args:TF` An internal conditional that checks if the code being added to the hook contains arguments.

```

\__hook_misused_if_replacing_args:nn
\__hook_replacing_args_true: 2556 \seq_new:N \g__hook_replacing_stack_seq
\__hook_replacing_args_false: 2557 \cs_new:Npn \__hook_misused_if_replacing_args:nn #1 #2
\__hook_replacing_args_reset: 2558 {
\g__hook_replacing_stack_seq 2559 \msg_expandable_error:nnm { latex2e } { should-not-happen }

```

```

2560     { Misused-\_hook_if_replacing_args:. }
2561   }
2562 \cs_new:Npn \_hook_if_replacing_args:TF
2563   { \_hook_misused_if_replacing_args:nn }
2564 \cs_new_protected:Npn \_hook_replacing_args_true:
2565   {
2566     \seq_gpush:No \g__hook_replacing_stack_seq
2567     { \_hook_if_replacing_args:TF }
2568     \cs_set:Npn \_hook_if_replacing_args:TF { \use_i:nn }
2569   }
2570 \cs_new_protected:Npn \_hook_replacing_args_false:
2571   {
2572     \seq_gpush:No \g__hook_replacing_stack_seq
2573     { \_hook_if_replacing_args:TF }
2574     \cs_set:Npn \_hook_if_replacing_args:TF { \use_ii:nn }
2575   }
2576 \cs_new_protected:Npn \_hook_replacing_args_reset:
2577   {
2578     \seq_gpop:NN \g__hook_replacing_stack_seq \l__hook_return_tl
2579     \cs_gset_eq:NN \_hook_if_replacing_args:TF \l__hook_return_tl
2580   }

```

(End of definition for _hook_if_replacing_args:TF and others.)

4.11 Messages

Hook errors are LaTeX kernel errors:

```

2581 \prop_gput:Nnn \g_msg_module_type_prop { hooks } { LaTeX }
And so are kernel errors (this should move elsewhere eventually).
2582 \prop_gput:Nnn \g_msg_module_type_prop { latex2e } { LaTeX }
2583 \prop_gput:Nnn \g_msg_module_name_prop { latex2e } { kernel }

2584 \msg_new:nnnn { hooks } { labels-incompatible }
2585   {
2586     Labels~'#1'~and~'#2'~are~incompatible
2587     \str_if_eq:nnF {#3} {??} { ~in-hook~'#3' } .~
2588     \int_compare:nNnTF {#4} = { 1 }
2589       { The~ code~ for~ both~ labels~ will~ be~ dropped. }
2590       { You~ may~ see~ errors~ later. }
2591   }
2592   { LaTeX~found~two~incompatible~labels~in~the~same~hook.~
2593     This~indicates~an~incompatibility~between~packages. }

2594 \msg_new:nnnn { hooks } { exists }
2595   { Hook~'#1'~ has~ already~ been~ declared. }
2596   { There~ already~ exists~ a~ hook~ declaration~ with~ this~
2597     name.\}
2598   { Please~ use~ a~ different~ name~ for~ your~ hook.}

2599 \langle latexrelease \rangle \IncludeInRelease{2023/06/01}{too-many-args}
2600 \langle latexrelease \rangle           {Hooks-with-args}

2601 \msg_new:nnnn { hooks } { too-many-args }
2602   { Too~many~arguments~for~hook~'#1'. }
2603   {

```

```

2604     You~tried~to~declare~a~hook~with~#2~arguments,~but~a~
2605     hook~can~only~have~up~to~nine.~LaTeX~will~define~this~
2606     hook~with~nine~arguments.
2607 }

2608 \msg_new:nnnn { hooks } { without-args }
2609 { Hook~'#1'~has~no~arguments. }
2610 {
2611     You~tried~to~use~\iow_char:N\|#2WithArguments~
2612     on~a~hook~that~takes~no~arguments.\\
2613     Check~the~usage~of~the~hook~or~use~\iow_char:N\|#2~instead.\\
2614     \\
2615     LaTeX~will~use~\iow_char:N\|#2.
2616 }

2617 \msg_new:nnnn { hooks } { one-time-args }
2618 { You~can't~have~arguments~in~used~one-time~hook~'#1'. }
2619 {
2620     You~tried~to~use~\iow_char:N\|#2WithArguments~
2621     on~a~one-time~hook~that~has~already~been~used.~
2622     You~have~to~add~the~code~before~the~hook~is~used,~
2623     or~add~the~code~without~arguments~using~\iow_char:N\|#2~instead.\\
2624     \\
2625     LaTeX~will~use~\iow_char:N\|#2.
2626 }

2627 <latexrelease>\EndIncludeInRelease
2628 <latexrelease>\IncludeInRelease{2020/10/01}{too-many-args}
2629 <latexrelease>          {Hooks-with-args}
2630 <latexrelease>\EndIncludeInRelease

2631 \msg_new:nnnn { hooks } { hook-disabled }
2632 { Cannot~add~code~to~disabled~hook~'#1'. }
2633 {
2634     The~hook~'#1'~you~tried~to~add~code~to~was~previously~disabled~
2635     with~\iow_char:N\hook_disable_generic:n-or~
2636     \iow_char:N\DisableGenericHook,~so~
2637     it~cannot~have~code~added~to~it.
2638 }

2639 \msg_new:nnn { hooks } { empty-label }
2640 {
2641     Empty~code~label~\msg_line_context:~
2642     Using~'\_hook_currname_or_default:'~instead.
2643 }

2644 \msg_new:nnn { hooks } { empty-hook }
2645 {
2646     Empty~hook~name~\msg_line_context:~
2647 }

2648 \msg_new:nnn { hooks } { no-default-label }
2649 {
2650     Missing~(empty)~default~label~\msg_line_context:. \\
2651     This~command~was~ignored.
2652 }

2653 \msg_new:nnnn { hooks } { unknown-rule }

```

```

2654 {
2655   Unknown~ relationship~ '#3'~
2656   between~ labels~ '#2'~ and~ '#4'~
2657   \str_if_eq:nnF {#1} {??} { ~in-hook~'#1' }. ~
2658   Perhaps~ a~ misspelling?
2659 }
2660 {
2661   The~ relation~ used~ not~ known~ to~ the~ system.~ Allowed~ values~ are~
2662   'before'~ or~ '<',~
2663   'after'~ or~ '>',~
2664   'incompatible-warning',~
2665   'incompatible-error',~
2666   'voids'~ or~
2667   'unrelated'.
2668 }
2669 \msg_new:nnnn { hooks } { rule-too-late }
2670 {
2671   Sorting-rule-for~'#1'~hook-applied-too-late.\\
2672   Try~setting-this-rule-earlier.
2673 }
2674 {
2675   You~tried~to~set~the~ordering~of~hook~'#1'~using\\
2676   \ \ \iow_char:N\DeclareHookRule{#1}{#2}{#3}{#4}\\
2677   but~hook~'#1'~was~already~used~as~a~one-time~hook,~
2678   thus~sorting-is\\
2679   no~longer~possible.~Declare~the~rule~
2680   before~the~hook-is-used.
2681 }
2682 \msg_new:nnnn { hooks } { misused-top-level }
2683 {
2684   Illegal~use-of~\iow_char:N \AddToHook{#1}[top-level]{...}.\\
2685   'top-level'~is~reserved~for~the~user's~document.
2686 }
2687 {
2688   The~'top-level'~label~is~meant~for~user~code~only,~and~should~only~
2689   be~used~(sparingly)~in~the~main~document.~Use~the~default~label~
2690   '\_hook_currname_or_default:'~for~this~\@cls@pkg,~or~another~
2691   suitable~label.
2692 }
2693 \msg_new:nnn { hooks } { set-top-level }
2694 {
2695   You~cannot~change~the~default~label~#1~'top-level'.~Illegal \
2696   \use:nn { ~ } { ~ } \iow_char:N \#2{#3} \
2697   \msg_line_context:.
2698 }
2699 \msg_new:nnn { hooks } { extra-pop-label }
2700 {
2701   Extra~\iow_char:N \PopDefaultHookLabel. \
2702   This~command~will~be~ignored.
2703 }
2704 \msg_new:nnn { hooks } { missing-pop-label }
2705 {
2706   Missing~\iow_char:N \PopDefaultHookLabel. \

```

```

2707   The~label~'~#1'~was~pushed~but~never~popped.~Something~is~wrong.
2708   }
2709 \msg_new:nnn { latex2e } { should-not-happen }
2710   {
2711     This~should~not~happen.~#1 \\\
2712     Please~report~at~https://github.com/latex3/latex2e.
2713   }
2714 \msg_new:nnn { hooks } { activate-disabled }
2715   {
2716     Cannot~ activate~ hook~ '~#1'~ because~ it~ is~ disabled!
2717   }
2718 \msg_new:nnn { hooks } { cannot-remove }
2719   {
2720     Cannot~remove~chunk~'~#2'~from~hook~'~#1'~because~
2721     \__hook_if_structure_exist:nTF {#1}
2722       { it~does~not~exist~in~that~hook. }
2723       { the~hook~does~not~exist. }
2724   }
2725 \msg_new:nnn { hooks } { generic-deprecated }
2726   {
2727     Generic~hook~'~#1/#2/#3'~is~deprecated. \\\
2728     Use~hook~'~#1/#3/#2'~instead.
2729   }

```

4.12 L^AT_EX 2_ε package interface commands

\NewHook Declaring new hooks ...

```

\NewReversedHook 2730 \NewDocumentCommand \NewHook           { m }
\NewMirroredHookPair 2731   { \hook_new:n {#1} }
2732 \NewDocumentCommand \NewReversedHook     { m }
2733   { \hook_new_reversed:n {#1} }
2734 \NewDocumentCommand \NewMirroredHookPair { mm }
2735   { \hook_new_pair:nn {#1}{#2} }

```

(End of definition for \NewHook, \NewReversedHook, and \NewMirroredHookPair. These functions are documented on page 199.)

\NewHookWithArguments Declaring new hooks with arguments...

```

\NewReversedHookWithArguments 2736 <latexrelease> \IncludeInRelease{2023/06/01}{\NewHookWithArguments}
\NewMirroredHookPairWithArguments 2737 <latexrelease>           {Hooks-with-args}
2738 \NewDocumentCommand \NewHookWithArguments           { mm }
2739   { \hook_new_with_args:nn {#1} {#2} }
2740 \NewDocumentCommand \NewReversedHookWithArguments   { mm }
2741   { \hook_new_reversed_with_args:nn {#1} {#2} }
2742 \NewDocumentCommand \NewMirroredHookPairWithArguments { mmm }
2743   { \hook_new_pair_with_args:nnn {#1} {#2} {#3} }
2744 <latexrelease> \EndIncludeInRelease
2745 <latexrelease> \IncludeInRelease{2020/10/01}{\NewHookWithArguments}
2746 <latexrelease>           {Hooks-with-args}
2747 <latexrelease> \cs_new_protected:Npn \NewHookWithArguments #1 #2 { }
2748 <latexrelease> \cs_new_protected:Npn \NewReversedHookWithArguments #1 #2 { }
2749 <latexrelease> \cs_new_protected:Npn \NewMirroredHookPairWithArguments #1 #2 #3{ }
2750 <latexrelease> \EndIncludeInRelease

```

(End of definition for `\NewHookWithArguments`, `\NewReversedHookWithArguments`, and `\NewMirroredHookPairWithArguments`. These functions are documented on page 200.)

```
2751 <latexrelease> \IncludeInRelease{2021/06/01}{\ActivateGenericHook}
2752 <latexrelease>           {Providing-hooks}
```

`\ActivateGenericHook` Providing new hooks ...

```
2753 \NewDocumentCommand \ActivateGenericHook { m }
2754   { \hook_activate_generic:n {#1} }
```

(End of definition for `\ActivateGenericHook`. This function is documented on page 201.)

`\DisableGenericHook` Disabling a generic hook.

```
2755 \NewDocumentCommand \DisableGenericHook { m }
2756   { \hook_disable_generic:n {#1} }
```

(End of definition for `\DisableGenericHook`. This function is documented on page 200.)

```
2757 <latexrelease> \EndIncludeInRelease
2758 <latexrelease> \IncludeInRelease{2020/10/01}{\ActivateGenericHook}
2759 <latexrelease>           {Providing-hooks}
2760 <latexrelease> \def \ActivateGenericHook #1 { }
2761 <latexrelease> \def \DisableGenericHook #1 { }
2762 <latexrelease> \EndIncludeInRelease
```

`\AddToHook`

`\AddToHookWithArguments`

```
2763 <latexrelease> \IncludeInRelease{2023/06/01}{\AddToHookWithArguments}
2764 <latexrelease>           {Hooks-with-args}
2765 \NewDocumentCommand \AddToHook { m o +m }
2766   { \hook_gput_code:nnn {#1} {#2} {#3} }
2767 \NewDocumentCommand \AddToHookWithArguments { m o +m }
2768   { \hook_gput_code_with_args:nnn {#1} {#2} {#3} }
2769 <latexrelease> \EndIncludeInRelease
2770 <latexrelease> \IncludeInRelease{2020/10/01}{\AddToHookWithArguments}
2771 <latexrelease>           {Hooks-with-args}
2772 <latexrelease> \cs_new_protected:Npn \AddToHookWithArguments #1 #2 #3 { }
2773 <latexrelease> \EndIncludeInRelease
```

(End of definition for `\AddToHook` and `\AddToHookWithArguments`. These functions are documented on page 202.)

`\AddToHookNext`

`\AddToHookNextWithArguments`

```
2774 <latexrelease> \IncludeInRelease{2023/06/01}{\AddToHookNextWithArguments}
2775 <latexrelease>           {Hooks-with-args}
2776 \NewDocumentCommand \AddToHookNext { m +m }
2777   { \hook_gput_next_code:nn {#1} {#2} }
2778 \NewDocumentCommand \AddToHookNextWithArguments { m +m }
2779   { \hook_gput_next_code_with_args:nn {#1} {#2} }
2780 <latexrelease> \EndIncludeInRelease
2781 <latexrelease> \IncludeInRelease{2020/10/01}{\AddToHookNextWithArguments}
2782 <latexrelease>           {Hooks-with-args}
2783 <latexrelease> \cs_new_protected:Npn \AddToHookNextWithArguments #1 #2 { }
2784 <latexrelease> \EndIncludeInRelease
```

(End of definition for `\AddToHookNext` and `\AddToHookNextWithArguments`. These functions are documented on page 204.)

`\ClearHookNext`

```
2785 \NewDocumentCommand \ClearHookNext { m }
2786   { \hook_gc_clear_next_code:n {#1} }
```

(End of definition for `\ClearHookNext`. This function is documented on page 204.)

`\RemoveFromHook`

```
2787 \NewDocumentCommand \RemoveFromHook { m o }
2788   { \hook_gremove_code:nn {#1} {#2} }
```

(End of definition for `\RemoveFromHook`. This function is documented on page 203.)

`\SetDefaultHookLabel` Now define a wrapper that replaces the top of the stack with the argument, and updates
`\PushDefaultHookLabel` `\g__hook_hook_curr_name_tl` accordingly.
`\PopDefaultHookLabel`

```
2789 \NewDocumentCommand \SetDefaultHookLabel { m }
2790   { \__hook_set_default_hook_label:n {#1} }
```

The label is only automatically updated with `\@onefilewithoptions` (`\usepackage` and `\documentclass`), but some packages, like `TikZ`, define package-like interfaces, like `\usetikzlibrary` that are wrappers around `\input`, so they inherit the default label currently in force (usually `top-level`, but it may change if loaded in another package). To provide a package-like behavior also for hooks in these files, we provide high-level access to the default label stack.

```
2791 \NewDocumentCommand \PushDefaultHookLabel { m }
2792   { \__hook_curr_name_push:n {#1} }
2793 \NewDocumentCommand \PopDefaultHookLabel { }
2794   { \__hook_curr_name_pop: }
```

The current label stack holds the labels for all files but the current one (more or less like `\@currnamestack`), and the current label token list, `\g__hook_hook_curr_name_tl`, holds the label for the current file. However `\@pushfilename` happens before `\@currname` is set, so we need to look ahead to get the `\@currname` for the label. `expl3` also requires the current file in `\@pushfilename`, so here we abuse `\@expl@push@filename@aux@@` to do `__hook_curr_name_push:n`.

```
2795 \cs_gset_protected:Npn \@expl@push@filename@aux@@ #1#2#3
2796   {
2797     \__hook_curr_name_push:n {#3}
2798     \str_gset:Nx \g_file_curr_name_str {#3}
2799     #1 #2 {#3}
2800   }
```

(End of definition for `\SetDefaultHookLabel`, `\PushDefaultHookLabel`, and `\PopDefaultHookLabel`. These functions are documented on page 207.)

`\UseHook`

`\UseOneTimeHook`

`\UseHookWithArguments`

`\UseOneTimeHookWithArguments`

Avoid the overhead of `xparse` and its protection that we don't want here (since the hook should vanish without trace if empty)!

```
2801 <latexrelease> \IncludeInRelease{2023/06/01}{\UseHookWithArguments}
2802 <latexrelease>           {Hooks-with-args}
2803 \cs_new:Npn \UseHook      { \hook_use:n }
2804 \cs_new:Npn \UseOneTimeHook { \hook_use_once:n }
2805 \cs_new:Npn \UseHookWithArguments { \hook_use:nw }
2806 \cs_new:Npn \UseOneTimeHookWithArguments { \hook_use_once:nw }
2807 <latexrelease> \EndIncludeInRelease
2808 <latexrelease> \IncludeInRelease{2020/10/01}{\UseHookWithArguments}
```

```

2809 <latexrelease> {Hooks-with-args}
2810 <latexrelease> \cs_new:Npn \UseHookWithArguments #1 #2 { }
2811 <latexrelease> \cs_new:Npn \UseOneTimeHookWithArguments #1 #2 { }
2812 <latexrelease> \EndIncludeInRelease

```

(End of definition for \UseHook and others. These functions are documented on page 201.)

\ShowHook

\LogHook

```

2813 \cs_new_protected:Npn \ShowHook { \hook_show:n }
2814 \cs_new_protected:Npn \LogHook { \hook_log:n }

```

(End of definition for \ShowHook and \LogHook. These functions are documented on page 210.)

\DebugHooksOn

\DebugHooksOff

```

2815 \cs_new_protected:Npn \DebugHooksOn { \hook_debug_on: }
2816 \cs_new_protected:Npn \DebugHooksOff { \hook_debug_off: }

```

(End of definition for \DebugHooksOn and \DebugHooksOff. These functions are documented on page 211.)

\DeclareHookRule

```

2817 \NewDocumentCommand \DeclareHookRule { m m m m }
2818 { \hook_gset_rule:nmmm {#1}{#2}{#3}{#4} }

```

(End of definition for \DeclareHookRule. This function is documented on page 208.)

\DeclareDefaultHookRule

This declaration is only supported before \begin{document}.

```

2819 \NewDocumentCommand \DeclareDefaultHookRule { m m m }
2820 { \hook_gset_rule:nmmm {??}{#1}{#2}{#3} }
2821 \@onlypreamble\DeclareDefaultHookRule

```

(End of definition for \DeclareDefaultHookRule. This function is documented on page 209.)

\ClearHookRule

A special setup rule that removes an existing relation. Basically _hook_rule_gclear:nmm plus fixing the property list for debugging.

FMi: Needs perhaps an L3 interface, or maybe it should get dropped?

```

2822 \NewDocumentCommand \ClearHookRule { m m m }
2823 { \hook_gset_rule:nmmm {#1}{#2}{unrelated}{#3} }

```

(End of definition for \ClearHookRule. This function is documented on page 208.)

\IfHookEmptyTF

\IfHookEmptyT

\IfHookEmptyF

Here we avoid the overhead of xparse, since \IfHookEmptyTF is used in \end (that is, every L^AT_EX environment). As a further optimization, use \let rather than \def to avoid one expansion step.

```

2824 \cs_new_eq:NN \IfHookEmptyTF \hook_if_empty:nTF
2825 \cs_new_eq:NN \IfHookEmptyT \hook_if_empty:nT
2826 \cs_new_eq:NN \IfHookEmptyF \hook_if_empty:nF

```

(End of definition for \IfHookEmptyTF, \IfHookEmptyT, and \IfHookEmptyF. These functions are documented on page 210.)

\IfHookExistsTF

Marked for removal and no longer documented in the doc section!

PhO: \IfHookExistsTF is used in jlreq.cls, pxatbegshi.sty, pxeverysel.sty, pxeveryshe.sty, so the public name may be an alias of the internal conditional for a while. Regardless, those packages' use for \IfHookExistsTF is not really correct and can be changed.

```

2827 \cs_new_eq:NN \IfHookExistsTF \_hook_if_usable:nTF

```

(End of definition for \IfHookExistsTF.)

4.13 Deprecated that needs cleanup at some point

```

\hook_disable:n      Deprecated.
\hook_provide:n     2828 \cs_new_protected:Npn \hook_disable:n
\hook_provide_reversed:n 2829 {
  \hook_provide_pair:nn 2830   \__hook_deprecated_warn:nn
  \__hook_activate_generic_reversed:n 2831   { hook_disable:n }
  \__hook_activate_generic_pair:nn 2832   { hook_disable_generic:n }
  2833   \hook_disable_generic:n
  2834 }
  2835 \cs_new_protected:Npn \hook_provide:n
  2836 {
  2837   \__hook_deprecated_warn:nn
  2838   { hook_provide:n }
  2839   { hook_activate_generic:n }
  2840   \hook_activate_generic:n
  2841 }
  2842 \cs_new_protected:Npn \hook_provide_reversed:n
  2843 {
  2844   \__hook_deprecated_warn:nn
  2845   { hook_provide_reversed:n }
  2846   { hook_activate_generic:n }
  2847   \__hook_activate_generic_reversed:n
  2848 }
  2849 \cs_new_protected:Npn \hook_provide_pair:nn
  2850 {
  2851   \__hook_deprecated_warn:nn
  2852   { hook_provide_pair:nn }
  2853   { hook_activate_generic:n }
  2854   \__hook_activate_generic_pair:nn
  2855 }
  2856 \cs_new_protected:Npn \__hook_activate_generic_reversed:n #1
  2857 { \__hook_normalize_hook_args:Nn \__hook_activate_generic:nn {#1} { - } }
  2858 \cs_new_protected:Npn \__hook_activate_generic_pair:nn #1#2
  2859 { \hook_activate_generic:n {#1} \__hook_activate_generic_reversed:n {#2} }

```

(End of definition for \hook_disable:n and others.)

```

\DisableHook      Deprecated.
\ProvideHook     2860 \cs_new_protected:Npn \DisableHook
\ProvideReversedHook 2861 {
\ProvideMirroredHookPair 2862   \__hook_deprecated_warn:nn
  2863   { DisableHook }
  2864   { DisableGenericHook }
  2865   \hook_disable_generic:n
  2866 }
  2867 \cs_new_protected:Npn \ProvideHook
  2868 {
  2869   \__hook_deprecated_warn:nn
  2870   { ProvideHook }
  2871   { ActivateGenericHook }
  2872   \hook_activate_generic:n
  2873 }
  2874 \cs_new_protected:Npn \ProvideReversedHook

```

```

2875 {
2876   \__hook_deprecated_warn:nn
2877     { ProvideReversedHook }
2878     { ActivateGenericHook }
2879   \__hook_activate_generic_reversed:n
2880 }
2881 \cs_new_protected:Npn \ProvideMirroredHookPair
2882 {
2883   \__hook_deprecated_warn:nn
2884     { ProvideMirroredHookPair }
2885     { ActivateGenericHook }
2886   \__hook_activate_generic_pair:nn
2887 }

```

(End of definition for \DisableHook and others.)

__hook_deprecated_warn:nn Warns about a deprecation, telling what should be used instead.

```

2888 \cs_new_protected:Npn \__hook_deprecated_warn:nn #1 #2
2889 { \msg_warning:nmmm { hooks } { deprecated } {#1} {#2} }
2890 \msg_new:nnn { hooks } { deprecated }
2891 {
2892   Command~\iow_char:N\|#1~is-deprecated~and~will~be~removed~in~a~
2893   future~release. \\ \\
2894   Use~\iow_char:N\|#2~instead.
2895 }

```

(End of definition for __hook_deprecated_warn:nn.)

4.14 Internal commands needed elsewhere

Here we set up a few horrible (but consistent) L^AT_EX_{2 ϵ} names to allow for internal commands to be used outside this module. We have to unset the @@ since we want double “at” sign in place of double underscores.

```

2896 <@@=>

```

```

\@expl@@@initialize@all@@
\@expl@@@hook@curr@name@pop@@
2897 \cs_new_eq:NN \@expl@@@initialize@all@@
2898   \__hook_initialize_all:
2899 \cs_new_eq:NN \@expl@@@hook@curr@name@pop@@
2900   \__hook_curr_name_pop:

```

(End of definition for \@expl@@@initialize@all@@ and \@expl@@@hook@curr@name@pop@@.)

Rolling back here doesn’t undefine the interface commands as they may be used in packages without rollback functionality. So we just make them do nothing which may or may not work depending on the code usage.

```

2901 %
2902 <latexrelease> \IncludeInRelease{0000/00/00}{lthooks}
2903 <latexrelease>           {The~hook~management}%
2904 <latexrelease>
2905 <latexrelease> \def \NewHook#1{}
2906 <latexrelease> \def \NewReversedHook#1{}
2907 <latexrelease> \def \NewMirroredHookPair#1#2{}
2908 <latexrelease>

```

```

2909 <latexrelease> \def \DisableGenericHook #1{
2910 <latexrelease>
2911 <latexrelease> \long\def \AddToHookNext#1#2{
2912 <latexrelease>
2913 <latexrelease> \def \AddToHook#1{\@gobble@AddToHook@args}
2914 <latexrelease> \providecommand \@gobble@AddToHook@args [2] [] {}
2915 <latexrelease>
2916 <latexrelease> \def \RemoveFromHook#1{\@gobble@RemoveFromHook@arg}
2917 <latexrelease> \providecommand \@gobble@RemoveFromHook@arg [1] [] {}
2918 <latexrelease>
2919 <latexrelease> \def \UseHook #1{
2920 <latexrelease> \def \UseOneTimeHook #1{
2921 <latexrelease> \def \ShowHook #1{
2922 <latexrelease> \let \DebugHooksOn \@empty
2923 <latexrelease> \let \DebugHooksOff \@empty
2924 <latexrelease>
2925 <latexrelease> \def \DeclareHookRule #1#2#3#4{
2926 <latexrelease> \def \DeclareDefaultHookRule #1#2#3{
2927 <latexrelease> \def \ClearHookRule #1#2#3{

```

If the hook management is not provided we make the test for existence false and the test for empty true in the hope that this is most of the time reasonable. If not a package would need to guard against running in an old kernel.

```

2928 <latexrelease> \long\def \IfHookExistsTF #1#2#3{#3}
2929 <latexrelease> \long\def \IfHookEmptyTF #1#2#3{#2}
2930 <latexrelease>
2931 <latexrelease> \EndModuleRelease
2932 <@@=hook>
2933 <latexrelease> \cs:w __hook_rollback_tidying: \cs_end:
2934 <latexrelease> \bool_lazy_and:nnT
2935 <latexrelease> { \int_compare_p:nNn { \sourceLaTeXdate } > { 20230600 } }
2936 <latexrelease> { \int_compare_p:nNn { \requestedLaTeXdate } < { 20230601 } }
2937 <latexrelease> {
2938 <latexrelease> \cs_gset_protected:Npn \__hook_rollback_tidying:
2939 <latexrelease> {
2940 <latexrelease> \@@latex@error { Rollback-code-executed-twice }
2941 <latexrelease> {
2942 <latexrelease> Something-went-wrong-(unless-this-was-
2943 <latexrelease> done-on-purpose-in-a-testing-environment).
2944 <latexrelease> }
2945 <latexrelease> \use_none:nnnn
2946 <latexrelease> }
2947 <latexrelease> \cs_set:Npn \__hook_tmp:w #1 #2
2948 <latexrelease> {
2949 <latexrelease> \__hook_tl_gset:cx { __hook#1~#2 }
2950 <latexrelease> {
2951 <latexrelease> \exp_args:No \exp_not:o
2952 <latexrelease> {
2953 <latexrelease> \cs:w __hook#1~#2 \exp_last_unbraced:Ne \cs_end:
2954 <latexrelease> { \__hook_braced_cs_parameter:n
2955 <latexrelease> { __hook#1~#2 } }
2956 <latexrelease> }
2957 <latexrelease> }

```

```

2958 <latexrelease>      }
2959 <latexrelease>      \seq_map_inline:Nn \g__hook_all_seq
2960 <latexrelease>      {
2961 <latexrelease>          \exp_after:wN \cs_gset_nopar:Npn
2962 <latexrelease>          \cs:w g__hook_#1_code_prop \exp_args:NNo \exp_args:No
2963 <latexrelease>          \cs_end: { \cs:w g__hook_#1_code_prop \cs_end: }
2964 <latexrelease>          \__hook_tmp:w { _toplevel } {#1}
2965 <latexrelease>          \__hook_tmp:w { _next } {#1}
2966 <latexrelease>      }
2967 <latexrelease> }
2968 \ExplSyntaxOff
2969 </2ekernel | latexrelease>
2970 <@@=>

```

File 09

ltxcmdhooks.dtx

1 Introduction

This file implements generic hooks for (arbitrary) commands. In theory every command `\langle name \rangle` offers now two associated hooks to which code can be added using `\AddToHook`,¹³ `\AddToHookNext`, `\AddToHookWithArguments`, and `\AddToHookNextWithArguments`.¹⁴

However, this is only true “in theory”. In practice there are a number of restrictions that makes it impossible to use such generic command hooks in a number of cases, so please read all of section 2 to understand what may prevent you from using them successfully.

The generic command hooks are:

cmd/⟨name⟩/before This hook is executed at the very start of the command, right after its arguments (if any) are parsed. The hook `⟨code⟩` runs in the command inside a call to `\UseHookWithArguments`. Any code added to this hook using `\AddToHookWithArguments` or `\AddToHookNextWithArguments` can access the command’s arguments using `#1`, `#2`, etc., up to the number of arguments of the command. If `\AddToHook` or `\AddToHookNext` are used, the arguments cannot be accessed (see the `lthooks` documentation¹⁵ on hooks with arguments).

cmd/⟨name⟩/after This hook is similar to `cmd/⟨name⟩/before`, but it is executed at the very end of the command body. This hook is implemented as a reversed hook.

The hooks are not physically present before `\begin{document}`¹⁶ (i.e., using a command in the preamble will never execute the hook) and if nobody has declared any code for them, then they are not added to the command code ever. For example, if we have the following definition

```
\newcommand\foo[2]{Code #1 for #2!}
```

then executing `\foo{A}{B}` will simply run `Code_A_for_B!` as it was always the case. However, if somebody, somewhere (e.g., in a package) adds

```
\AddToHook{cmd/foo/before}{<before code>}
```

then, after `\begin{document}` the definition of `\foo` will be:

```
\renewcommand\foo[2]{%
  \UseHookWithArguments{cmd/foo/before}{2}{#1}{#2}%
  Code #1 for #2!}
```

and similarly `\AddToHook{cmd/foo/after}{<after code>}` alters the definition to

¹³In this documentation, when something is being said about `\AddToHook`, the same will be valid for `\AddToHookWithArguments`, unless that particular paragraph is highlighting the differences between both. The same is true for the other hook-related functions and their `...WithArguments` counterparts.

¹⁴In practice this is not supported for all types of commands, see section 2.2 for the restrictions that apply and what happens if one tries to use this with commands for which this is not supported.

¹⁵`texdoc lthooks-doc`

¹⁶More specifically, they are inserted in the commands after the `begindocument` hook, so they are also not present while \LaTeX is reading the `.aux` file.

```

\renewcommand\foo[2]{%
  Code #1 for #2!%
  \UseHookWithArguments{cmd/foo/after}{2}{#1}{#2}}

```

In other words, the mechanism is similar to what `etoolbox` offers with `\pretocmd` and `\apptocmd` with the important differences

- that code can be prepended or appended (i.e., added to the hooks) even if the command itself is not (yet) defined, because the defining package has not been loaded at this point;
- and that by using the hook management interface it is now possible to define how the code chunks added in these places are ordered, if different packages want to add code at these points.

2 Restrictions and Operational details

Adding arbitrary material to commands is tricky because most of the time we do not know what the macro expects as arguments when expanding and \TeX doesn't have a reliable way to see that, so some guesswork has to be employed.

We can do this in most cases when commands are defined using `\NewDocumentCommand` or `\newcommand` (with a few exceptions). For commands defined with `\def` the situation is less good. Common cases where the command hooks will not work are:

- Commands that use special catcode settings within their definition. In that case it is usually not possible to augment the definition (see 2.1).
- If a command is defined while `\ExplSyntaxOn` is in force **and** the command contains `~` characters to represent spaces, then it can't be patched to include the command hooks. In fact in some very special circumstances you might even get a low-level error rather than the information that the command can't be patched (see, for example, <https://github.com/latex3/latex2e/issues/1430>).
- Commands that have arguments as far as the user is concerned (e.g., `\section` or `\caption`), but are defined in a way that these arguments are not read by the user level command but only later during the processing. In that case the `after` hook doesn't work at all. The before hook only works with `\AddToHook` but not with `\AddToHookWithArguments` because the arguments haven't been read at that point where the hook is patched in. See section 2.2.
- Adding a specific generic command hook is only attempted once per command, thus after redefining a command such hooks will no longer be there and will also not being re-added, see section 2.1.1.

All this means that you have to have a good understanding of how commands are defined when you attempt to make use of such hooks and something goes wrong. What can help in that case is to turn on `\DebugHooksOn` in which case you get much more (low-level) details on why something fails and what was tried to enable the hooks.

2.1 Patching

The code here tries to find out if a command was defined with `\newcommand` or `\DeclareRobustCommand` or `\NewDocumentCommand`, and if so it *assumes* that the argument specification of the command is as expected (which is not fail-proof, if someone redefines the internals of these commands in devious ways, but is a reasonable assumption).

If the command is one of the defined types, the code here does a sandboxed expansion of the command such that it can be redefined again exactly as before, but with the hook code added.

If however the command is not a known type (it was defined with `\def`, for example), then the code uses an approach similar to `etoolbox`'s `\patchcmd` to retokenize the command with the hook code in place. This procedure, however, is more likely to fail if the `catcode` settings are not the same as the ones at the time of command's definition, so not always adding a hook to a command will work.

2.1.1 Timing

When `\AddToHook` (or its `expl3` equivalent) is called with a generic `cmd` hook, say, `cmd/foo/before`, for the first time (that is, no code was added to that same hook before), in the preamble of a document, it will store a patch instruction for that command until `\begin{document}`, and only then all the commands which had hooks added will be patched in one go. That means that no command in the preamble will have hooks patched into them.

At `\begin{document}` all the delayed patches will be executed, and if the command doesn't exist the code is still added to the hook, but it will not be executed. After `\begin{document}`, when `\AddToHook` is called with a generic `cmd` hook the first time, the command will be immediately patched to include the hook, and if it doesn't exist or if it can't be patched for any reason, an error is thrown; if `\AddToHook` was already used in the preamble no new patching is attempted.

This has the consequence that a command defined or redefined after `\begin{document}` only uses generic `cmd` hook code if `\AddToHook` is called for the first time after the definition is made, or if the command explicitly uses the generic hook in its definition by declaring it with `\NewHookPair` adding `\UseHook` as part of the code.¹⁷

2.2 Commands that look ahead

Some commands are defined in different “steps” and they look ahead in the input stream to find more arguments. If you try to add some code to the `cmd/<name>/after` hook of such command, it will not work, and it is not possible to detect that programmatically, so the user has to know (or find out) which commands can or cannot have hooks attached to them.

One good example is the `\section` command. You can add something to the `cmd/section/before` hook (but only with `\AddToHook` not `\AddToHookWithArguments`), but if you try to add anything to the `cmd/section/after` hook, `\section` will no longer work at all. That happens because the `\section` macro takes no argument, but instead calls a few internal `LATEX` macros to look for the optional and mandatory arguments. By adding code to the `cmd/section/after` hook, you get in the way of that scanning.

¹⁷We might change this behavior in the main document slightly after gaining some usage experience.

In such a case, where it is known that a specific generic command hook does not work if code is added to it, the package author can add a `\DisableGenericHook`¹⁸ declaration to prevent this from happening in user documents and thereby avoiding obscure errors.

3 Package Author Interface

The `cmd` hooks are, by default, available for all commands that can be patched to add the hooks. For some commands, however, the very beginning or the very end of the code is not the best place to put the hooks, for example, if the command looks ahead for arguments (see section 2.2).

If you are a package author and you want to add the hooks to your own commands in the proper position you can define the command and manually add the `\UseHookWithArguments` calls inside the command in the proper positions, and manually define the hooks with `\NewHookWithArguments` or `\NewReversedHookWithArguments`. When the hooks are explicitly defined, patching is not attempted so you can make sure your command works properly. For example, an (admittedly not really useful) command that typesets its contents in a framed box with width optionally given in parentheses:

```
\newcommand\fancybox{\@ifnextchar({\@fancybox}{\@fancybox(5cm)}}
\def\@fancybox(#1)#2{\fbox{\parbox{#1}{#2}}}
```

If you try that definition, then add some code after it with

```
\AddToHook{cmd/fancybox/after}{<code>}
```

and then use the `\fancybox` command you will see that it will be completely broken, because the hook will get executed in the middle of parsing for optional (...) argument.

If, on the other hand, you want to add hooks to your command you can do something like:

```
\newcommand\fancybox{\@ifnextchar({\@fancybox}{\@fancybox(5cm)}}
\def\@fancybox(#1)#2{\fbox{%
    \UseHookWithArguments{cmd/fancybox/before}{2}{#1}{#2}%
    \parbox{#1}{#2}%
    \UseHookWithArguments{cmd/fancybox/after}{2}{#1}{#2}}}
\NewHookWithArguments{cmd/fancybox/before}{2}
\NewReversedHookWithArguments{cmd/fancybox/after}{2}
```

then the hooks will be executed where they should and no patching will be attempted. It is important that the hooks are declared with `\NewHookWithArguments` or `\NewReversedHookWithArguments`, otherwise the command hook code will try to patch the command. Note also that the call to `\UseHookWithArguments{cmd/fancybox/before}` does not need to be in the definition of `\fancybox`, but anywhere it makes sense to insert it (in this case in the internal `\@fancybox`).

Alternatively, if for whatever reason your command does not support the generic hooks provided here, you can disable a hook with `\DisableGenericHook`¹⁹, so that when someone tries to add code to it they will get an error. Or if you don't want the error, you can simply declare the hook with `\NewHook` and never use it.

¹⁸Please use `\DisableGenericHook` if at all, only on hooks that you “own”, i.e., for commands your package or class defines and not second guess whether or not hooks of other packages should get disabled!

¹⁹Please use `\DisableGenericHook` if at all, only on hooks that you “own”, i.e., for commands your package or class defines and not second guess whether or not hooks of other packages should get disabled!

The above approach is useful for really complex commands where for one or the other reason the hooks can't be placed at the very beginning and end of the command body and some hand-crafting is needed. However, in the example above the real (and in fact only) issue is the cascading argument parsing in the style developed long ago in L^AT_EX 2.09. Thus, a much simpler solution for this case is to replace it with the modern `\NewDocumentCommand` syntax and define the command as follows:

```
\DeclareDocumentCommand\fancybox{D()}{5cm}m}{\fbox{\parbox{#1}{#2}}}
```

If you do that then both hooks automatically work and are patched into the right places.

3.1 Arguments and redefining commands

The code in `ltxcmdhooks` does its best to find out how many arguments a given command has, and to insert the appropriate call to `\UseHookWithArguments`, so that the arguments seen by the hook are exactly those grabbed by the command (the hook, after all, is a macro call, so the arguments have to be placed in the right order, or they won't match).

When using the package writer interface, as discussed in section 3, to change the position of the hooks in your commands, you are also free to change how the hook code in your command sees its arguments. When a `cmd` hook is declared with `\NewHook` (or `\NewHookWithArguments` or other variations of that), it loses its “generic” nature and works as a regular hook. This means that you may choose to declare it without arguments regardless if the command takes arguments or not, or declare it with arguments, even if the command takes none.

However, this flexibility should not be abused. When using a nonstandard configuration for the hook arguments, think reasonably: a user will expect that the argument `#1` in the hook corresponds to the argument's first argument, and so on. Any other configuration is likely to cause confusion and, if used, will have to be well documented.

This flexibility, however, allows you to “correct” the arguments for the hooks. For example, L^AT_EX's `\refstepcounter` has a single argument, the name of the counter. The `cleveref` package adds an optional argument to `\refstepcounter`, making the name of the counter argument `#2`. If the author of `cleveref` wanted, for whatever reason, to add hooks to `\refstepcounter`, to preserve compatibility he could write something along the lines of:

```
\NewHookWithArguments{cmd/refstepcounter/before}{1}
\renewcommand\refstepcounter[2][<default>]{%
  \UseHookWithArguments{cmd/refstepcounter/before}{1}{#2}%
  <code for \refstepcounter>}
```

so that the mandatory argument, which is arg `#2` in the definition, would still be seen as `#1` in the hook code.

Another possibility would be to place the optional argument as the second argument for the hook, so that people looking for it would be able to use it. In either case, it would have to be well documented to cause as little confusion as possible.

4 The Implementation

4.1 Execution plan

To add `before` and `after` hooks to a command we will need to peek into the definition of a command, which is always a tricky thing to do. Some cases are easy because we

know how the command was defined, so we can assume how its *(parameter text)* looks like (for example a command defined with `\newcommand` may have an optional argument followed by a run of mandatory arguments), so we can just expand that command and make it grab `#1`, `#2`, etc. as arguments and define it all back with the hooks added.

Life's usually not that easy, so with some commands we can't do that (a `#1` might as well be `#12112` instead of the expected `#6112`, for example) so we need to resort to "patching" the command: read its `\meaning`, and tokenize it again with `\scantokens` and hope for the best.

So the overall plan is:

1. Check if a command is of a known type (that is, defined with `\newcommand`²⁰, `\DeclareRobustCommand`, or `\New(Expandable)DocumentCommand`), and if is, take appropriate action.
2. If the command is not a known type, we'll check if the command can be patched. Two things will prevent a command from being patched: if it was defined in a nonstandard catcode setting, or if it is an internal `expl3` command with `__<module>` in its name, in which case we refuse to patch.
3. If the command was defined in nonstandard catcode settings, we will try a few standard ones to try our best to carry out the patching. If this doesn't help either, the code will give up and throw an error.

```

1 <@@=hook>
2 <*2kernel | latexrelease>
3 \ExplSyntaxOn
4 <latexrelease> \NewModuleRelease{2021/06/01}{ltxcmdhooks}
5 <latexrelease> {The-hook-management-system-for-commands}

```

4.2 Variables

`\g_hook_patch_action_list_tl` Pairs of `\if<cmd>..\patch<cmd>` to be used with `\robust@command@act` when looking for a known patching rule. This token list is exposed because we see some future applications (with very specialized packages, such as `etoolbox` that may want to extend the pairs processed. It is not meant for general use which is why it is not documented in the interface documentation above.

```

6 \tl_new:N \g_hook_patch_action_list_tl

```

(End of definition for `\g_hook_patch_action_list_tl`.)

`\l__hook_patch_num_args_int` The number of arguments in a macro being patched.

```

7 \int_new:N \l__hook_patch_num_args_int

```

(End of definition for `\l__hook_patch_num_args_int`.)

`\l__hook_patch_prefixes_tl` The prefixes and parameters of the definition for the macro being patched.

```

\l__hook_param_text_tl 8 \tl_new:N \l__hook_patch_prefixes_tl
\l__hook_replace_text_tl 9 \tl_new:N \l__hook_param_text_tl
10 \tl_new:N \l__hook_replace_text_tl

```

²⁰It's not always possible to reliably detect this case because a command defined with no optional argument is indistinguishable from a `\defed` command.

(End of definition for `\l__hook_patch_prefixes_tl`, `\l__hook_param_text_tl`, and `\l__hook_replace_text_tl`.)

`\c__hook_hash_tl` `\c__hook_hashes_tl` Two constant token lists that contain one and two parameter tokens.

```

11 \tl_const:Nn \c__hook_hash_tl { # }
12 \tl_const:Nn \c__hook_hashes_tl { # # }

```

(End of definition for `\c__hook_hash_tl` and `\c__hook_hashes_tl`.)

`__hook_exp_not:NN` `__hook_def_cmd:w` Two temporary macros that change depending on the macro being patched.

```

13 \cs_new_eq:NN \__hook_exp_not:NN ?
14 \cs_new_eq:NN \__hook_def_cmd:w ?

```

(End of definition for `__hook_exp_not:NN` and `__hook_def_cmd:w`.)

`\q__hook_recursion_tail` `\q__hook_recursion_stop` Internal quarks for recursion: they can't appear in any macro being patched.

```

15 \quark_new:N \q__hook_recursion_tail
16 \quark_new:N \q__hook_recursion_stop

```

(End of definition for `\q__hook_recursion_tail` and `\q__hook_recursion_stop`.)

`\g__hook_delayed_patches_prop` A list containing the patches delayed to `\begin{document}`, so that patching is not attempted twice.

```

17 \prop_new:N \g__hook_delayed_patches_prop

```

(End of definition for `\g__hook_delayed_patches_prop`.)

`__hook_patch_debug:x` A helper for patching debug info.

```

18 \cs_new_protected:Npn \__hook_patch_debug:x #1
19   { \__hook_debug:n { \iow_term:x { [lthooks]~#1 } } }

```

(End of definition for `__hook_patch_debug:x`.)

4.3 Variants

`\tl_rescan:nV` `expl3` function variants used throughout the code.

```

20 \cs_generate_variant:Nn \tl_rescan:nn { nV }

```

(End of definition for `\tl_rescan:nV`.)

4.4 Patching or delaying

Before `\begin{document}` all patching is delayed.

`__hook_try_put_cmd_hook:n` `__hook_try_put_cmd_hook:w` This function is called from within `\AddToHook`, when code is first added to a generic `cmd` hook. If it is called within in the preamble, it delays the action until `\begin{document}`; otherwise it tries to update the hook.

```

21 <latexrelease>\IncludeInRelease{2024/12/22}{\__hook_try_put_cmd_hook:n}%
22 <latexrelease>           {Don't~define~command}
23 \cs_new_protected:Npn \__hook_try_put_cmd_hook:n #1
24   { \__hook_try_put_cmd_hook:w #1 / / \s__hook_mark {#1} }
25 \cs_new_protected:Npn \__hook_try_put_cmd_hook:w
26   #1 / #2 / #3 / #4 \s__hook_mark #5
27   {
28     \__hook_debug:n { \iow_term:n { ->~Adding~cmd-hook~to~'#2'~(#{3}): } }

```

`__hook_patch_cmd_or_delay:Nnn` expects the command to be patched as its first argument so we need to construct it from its name (`#2`). However, at this moment it may not exist yet, so using `\cs:w` would incorrectly turn it from “undefined” into `\relax`. We therefore use the following curious construction: we start a group and expand out of it to call `\cs:w`. If the command is now changed to `\relax` the `\group_end:` will undo that change, but the token is nevertheless there to be consumed by `__hook_patch_cmd_or_delay:Nnn`.

```

29   \group_begin:
30     \exp_after:wN
31   \group_end:
32     \exp_after:wN
33   \__hook_patch_cmd_or_delay:Nnn
34     \cs:w #2\cs_end:
35     {#2} {#3}
36   }
37 <latexrelease>\EndIncludeInRelease

38 <latexrelease>\IncludeInRelease{2021/11/15}{\__hook_try_put_cmd_hook:n}%
39 <latexrelease>           {Standardise-generic-hook-names}
40 <latexrelease>\cs_new_protected:Npn \__hook_try_put_cmd_hook:n #1
41 <latexrelease>  { \__hook_try_put_cmd_hook:w #1 / / / \s__hook_mark {#1} }
42 <latexrelease>\cs_new_protected:Npn \__hook_try_put_cmd_hook:w
43 <latexrelease>    #1 / #2 / #3 / #4 \s__hook_mark #5
44 <latexrelease>  {
45 <latexrelease>    \__hook_debug:n { \iow_term:n { ->~Adding~cmd~hook~to~'~#2'~(~#3): } }
46 <latexrelease>    \exp_args:Nc \__hook_patch_cmd_or_delay:Nnn {#2} {#2} {#3}
47 <latexrelease>  }
48 <latexrelease>\EndIncludeInRelease

49 <latexrelease>\IncludeInRelease{2021/06/01}{\__hook_try_put_cmd_hook:n}%
50 <latexrelease>           {Standardise-generic-hook-names}
51 <latexrelease>\cs_new_protected:Npn \__hook_try_put_cmd_hook:n #1
52 <latexrelease>  { \__hook_try_put_cmd_hook:w #1 / / / \s__hook_mark {#1} }
53 <latexrelease>\cs_new_protected:Npn \__hook_try_put_cmd_hook:w
54 <latexrelease>    #1 / #2 / #3 / #4 \s__hook_mark #5
55 <latexrelease>  {
56 <latexrelease>    \__hook_debug:n { \iow_term:n { ->~Adding~cmd~hook~to~'~#2'~(~#3): } }
57 <latexrelease>    \str_case:nnTF {#3}
58 <latexrelease>      { { before } { } { after } { } }
59 <latexrelease>      { \exp_args:Nc \__hook_patch_cmd_or_delay:Nnn {#2} {#2} {#3} }
60 <latexrelease>      { \msg_error:nnnn { hooks } { wrong-cmd-hook } {#2} {#3} }
61 <latexrelease>  }
62 <latexrelease>\EndIncludeInRelease

```

(End of definition for `__hook_try_put_cmd_hook:n` and `__hook_try_put_cmd_hook:w`.)

`__hook_patch_cmd_or_delay:Nnn`
`__hook_cmd_begindocument_code:`

In the preamble, `__hook_patch_cmd_or_delay:Nnn` just adds the patch instruction to a property list to be executed later.

```

63 \cs_new_protected:Npn \__hook_patch_cmd_or_delay:Nnn #1 #2 #3
64   {
65     \__hook_debug:n { \iow_term:n { ->~Add~generic~cmd~hook~for~#2~(~#3). } }
66     \__hook_debug:n
67     { \iow_term:n { !~In~the~preamble:~delaying. } }
68     \prop_gput:Nnn \g__hook_delayed_patches_prop { #2 / #3 }
69     { \__hook_cmd_try_patch:nn {#2} {#3} }

```

```
70 }
```

The delayed patches are added to a property list to prevent duplication, and the code stored in the property list for each key is executed. The function `__hook_patch_cmd_or_delay:Nnn` is also redefined to be `__hook_patch_command:Nnn` so that no further delaying is attempted.

```
71 \cs_new_protected:Npn \__hook_cmd_begindocument_code:
72 {
73   \cs_gset_eq:NN \__hook_patch_cmd_or_delay:Nnn \__hook_patch_command:Nnn
74   \prop_map_function:NN \g__hook_delayed_patches_prop { \use_ii:nn }
75   \prop_gc_clear:N \g__hook_delayed_patches_prop
76   \cs_undefine:N \__hook_cmd_begindocument_code:
77 }
78 \g@addto@macro \@kernel@after@begindocument
79 { \__hook_cmd_begindocument_code: }
```

(End of definition for `__hook_patch_cmd_or_delay:Nnn` and `__hook_cmd_begindocument_code:.`)

```
\__hook_cmd_try_patch:nn
```

At `\begin{document}` tries patching the command if the hook was not manually created in the meantime. If the document does not exist, no error is raised here as it may hook into a package that wasn't loaded. Hooks added to commands in the document body still raise an error if the command is not defined.

```
80 \cs_new_protected:Npn \__hook_cmd_try_patch:nn #1 #2
81 {
82   \__hook_debug:n
83   { \iow_term:x { ->-\string\begin{document}-try-cmd / #1 / #2. } }
84   \__hook_if_declared:nTF { cmd / #1 / #2 }
85   {
86     \__hook_debug:n
87     { \iow_term:n { .->-Giving-up:~hook~already~created. } }
88   }
89   {
90     \cs_if_exist:cT {#1}
91     { \exp_args:Nc \__hook_patch_command:Nnn {#1} {#1} {#2} }
92   }
93 }
```

(End of definition for `__hook_cmd_try_patch:nn.`)

4.5 Patching commands

```
\__hook_patch_command:Nnn
\__hook_patch_check:NNnn
\__hook_if_public_command:NTF
\__hook_if_public_command:w
```

`__hook_patch_command:Nnn` will do some sanity checks on the argument to detect if it is possible to add hooks to the command, and raises an error otherwise. If the command can contain hooks, then it uses `\robust@command@act` to find out what type is the command, and patch it accordingly.

```
94 \cs_new_protected:Npn \__hook_patch_command:Nnn #1 #2 #3
95 {
96   \__hook_patch_debug:x { analyzing~'\token_to_str:N #1' }
97   \__hook_patch_debug:x { \token_to_str:N #1 = \token_to_meaning:N #1 }
98   \__hook_patch_check:NNnn \cs_if_exist:NTF #1 { undef }
99   {
100     \__hook_patch_debug:x { ++~control~sequence~is~defined }
101     \__hook_patch_check:NNnn \token_if_macro:NTF #1 { macro }
102     {
```

```

103         \__hook_patch_debug:x { ++~control~sequence~is~a~macro }
104         \__hook_patch_check:NNnn \__hook_if_public_command:NTF #1 { expl3 }
105         {
106             \__hook_patch_debug:x { ++~macro~is~not~private }
107             \robust@command@act
108             \g_hook_patch_action_list_tl #1
109             \__hook_retokenize_patch:Nnn { #1 {#2} {#3} }
110         }
111     }
112 }
113 }

```

And here's the auxiliary used above:

```

114 \cs_new_protected:Npn \__hook_patch_check:NNnn #1 #2 #3 #4
115 {
116     #1 #2 {#4}
117     {
118         \msg_error:nxxx { hooks } { cant-patch }
119         { \token_to_str:N #2 } {#3}
120     }
121 }

```

and a conditional `__hook_if_public_command:NTF` to check if a command has `__` in its name (no other checking is performed). Primitives with `:D` in their name could be included here, but they are already discarded in the `\token_if_macro:NTF` test above.

```

122 \use:x
123 {
124     \prg_new_protected_conditional:Npnn
125     \exp_not:N \__hook_if_public_command:N ##1 { TF }
126     {
127         \exp_not:N \exp_last_unbraced:Nf
128         \exp_not:N \__hook_if_public_command:w
129         { \exp_not:N \cs_to_str:N ##1 }
130         \tl_to_str:n { _ _ } \s__hook_mark
131     }
132 }
133 \exp_last_unbraced:NNNNo
134 \cs_new_protected:Npn \__hook_if_public_command:w
135 #1 \tl_to_str:n { _ _ } #2 \s__hook_mark
136 {
137     \tl_if_empty:nTF {#2}
138     { \prg_return_true: }
139     { \prg_return_false: }
140 }

```

(End of definition for `__hook_patch_command:Nnn` and others.)

4.5.1 Patching by expansion and redefinition

`\g_hook_patch_action_list_tl` This is the list of known command types and the function that patches the command hooks into them. The conditionals are taken from `\ShowCommand`, `\NewCommandCopy` and `__kernel_cmd_if_xparse:NTF` defined in `ltxcmd`.

```

141 \tl_gset:Nn \g_hook_patch_action_list_tl
142 {
143     { \@if@DeclareRobustCommand \__hook_patch_DeclareRobustCommand:Nnn }

```



```

144     { \@if@newcommand \__hook_patch_newcommand:Nnn }
145     { \__kernel_cmd_if_xparse:NTF \__hook_cmd_patch_xparse:Nnn }
146   }

```

(End of definition for \g_hook_patch_action_list_tl.)

__hook_patch_DeclareRobustCommand:Nnn

At this point we know that the commands can be patched by expanding then redefining. These are the cases of commands defined with `\newcommand` with an optional argument or with `\DeclareRobustCommand`.

With `__hook_patch_DeclareRobustCommand:Nnn` we check if the command has an optional argument (with a test counter-intuitively called `\@if@newcommand`; also make sure the command doesn't take args by calling `\robust@command@chk@safe`). If so, we pass the patching action to `__hook_patch_newcommand:Nnn`, otherwise we call the patching engine `__hook_patch_expand_redefine:NNnn` with a `\c_false_bool` to indicate that there is no optional argument.

```

147 \cs_new_protected:Npn \__hook_patch_DeclareRobustCommand:Nnn #1
148   {
149     \exp_args:Nc \__hook_patch_DeclareRobustCommand_aux:Nnn
150     { \cs_to_str:N #1 ~ }
151   }
152 \cs_new_protected:Npn \__hook_patch_DeclareRobustCommand_aux:Nnn #1
153   {
154     \robust@command@chk@safe #1
155     { \@if@newcommand #1 }
156     { \use_ii:nn }
157     { \__hook_patch_newcommand:Nnn }
158     { \__hook_patch_expand_redefine:NNnn \c_false_bool }
159     #1
160   }

```

(End of definition for __hook_patch_DeclareRobustCommand:Nnn.)

__hook_patch_newcommand:Nnn

If the command was defined with `\newcommand` and an optional argument, call the patching engine with a `\c_true_bool` to flag the presence of an optional argument, and with `\command` to patch the actual code for `\command`.

```

161 \cs_new_protected:Npn \__hook_patch_newcommand:Nnn #1
162   {
163     \exp_args:NNc \__hook_patch_expand_redefine:NNnn \c_true_bool
164     { \c_backslash_str \cs_to_str:N #1 }
165   }

```

(End of definition for __hook_patch_newcommand:Nnn.)

__hook_cmd_patch_xparse:Nnn

And for commands defined by the xparse commands use this for patching:

```

166 \cs_new_protected:Npn \__hook_cmd_patch_xparse:Nnn #1
167   {
168     \exp_args:NNc \__hook_patch_expand_redefine:NNnn \c_false_bool
169     { \cs_to_str:N #1 ~ code }
170   }

```

(End of definition for __hook_cmd_patch_xparse:Nnn.)

```

\__hook_patch_expand_redefine:NNnn
\__hook_redefine_with_hooks:Nnnn
\__hook_make_prefixes:w

```

Now the real action begins. Here we have in #1 a boolean indicating if the command has a leading [...] delimited argument, in #2 the command control sequence, in #3 the name of the command (note that #1 ≠ \csname#2\endcsname at this point!), and in #4 the hook position, either before or after.

Patching with expansion+redefinition is trickier than it looks like at first glance. Suppose the simple definition:

```
\def\foo#1{#1##2}
```

When defined, its *replacement text* will be a token list containing:

```
out_param 1, mac_param #, character 2
```

Then, after expanding \foo{##1} (here ## denotes a single #6) we end up with a token list with out_param 1 replaced:

```
mac_param #, character 1, mac_param #, character 2
```

that is, the definition would be:

```
\def\foo#1{#1#2}
```

which obviously fails, because the original input in the definition was ## but TeX reduced that to a single parameter token #6 when carrying out the definition. That leaves no room for a clever solution with (say) \unexpanded, because anything that would double the second #6, would also (incorrectly) double the first, so there's not much to do other than a manual solution.

There are three cases we can distinguish to make things hopefully faster on simpler cases:

1. a macro with no parameters;
2. a macro with no parameter tokens in its definition;
3. a macro with parameters and parameter tokens.

The first case is trivial: if the macro has no parameters, we can just use \unexpanded around it, and if there is a parameter token in it, it is handled correctly (the macro can be treated as a tl variable).

The second case requires looking at the *replacement text* of the macro to see if it has a parameter token in there. If it does not, then there is no worry, and the macro can be redefined normally (without \unexpanded).

The third case, as usual, is the devious one. Here we'll have to loop through the definition token by token, and double every parameter token, so that this case can be handled like the previous one.

```

171 <latexrelease>\IncludeInRelease{2023/06/01}{\__hook_patch_expand_redefine:NNnn}
172 <latexrelease> {cmd~hooks~with~args}
173 \cs_new_protected:Npn \__hook_patch_expand_redefine:NNnn #1 #2 #3 #4
174 {
175   \__hook_patch_debug:x { ++~command~can~be~patched~without~rescanning }

```

We'll start by counting the number of arguments in the command by counting the number of characters in the `\cs_parameter_spec:N` of the macro, divided by two, and subtracting one if the command has an optional argument (that is, an extra `[]` in its `\langle parameter text \rangle`).

```

176   \int_set:Nn \l__hook_patch_num_args_int
177   {
178     \exp_args:Nf \str_count:n { \__kernel_cs_parameter_spec:N #2 } / 2
179     \bool_if:NT #1 { -1 }
180   }

```

Now build two token lists:

`\l__hook_param_text_tl` will contain the `\langle parameter text \rangle` to be used when redefining the macro. It should be identical to the `\langle parameter text \rangle` used when originally defining that macro.

`\l__hook_replace_text_tl` will contain braced pairs of `\c__hook_hashes_tl\langle num \rangle` to feed to the macro when expanded. This token list as well as the previous will have the first item surrounded by `[...]` in the case of an optional argument.

The use of `\c__hook_hashes_tl` here is to differentiate actual parameters in the macro from parameter tokens in the original definition of the macro. Later on, `\c__hook_hashes_tl` is either replaced by actual parameter tokens, or expanded into them.

```

181   \int_compare:nNnTF { \l__hook_patch_num_args_int } > { \c_zero_int }
182   {

```

We'll first check if the command has any parameter token in its definition (feeding it empty arguments), and set `__hook_exp_not:n` accordingly. `__hook_exp_not:n` will be used later to either leave `\c__hook_hashes_tl` or expand it, and also to remember the result of `__hook_if_has_hash:nTF` to avoid testing twice (the test can be rather slow).

```

183     \tl_set:Nx \l__hook_tmpa_tl { \bool_if:NTF #1 { [] } { { } } }
184     \int_step_inline:nnn { 2 } { \l__hook_patch_num_args_int }
185     { \tl_put_right:Nn \l__hook_tmpa_tl { { } } }
186     \exp_args:NNo \exp_args:No \__hook_if_has_hash:nTF
187     { \exp_after:wN #2 \l__hook_tmpa_tl }
188     { \cs_set_eq:NN \__hook_exp_not:n \exp_not:n }
189     { \cs_set_eq:NN \__hook_exp_not:n \use:n }
190     \cs_set_protected:Npn \__hook_tmp:w ##1 ##2
191     {
192       ##1 \l__hook_param_text_tl { \use:n ##2 }
193       ##1 \l__hook_replace_text_tl { \__hook_exp_not:n {##2} }
194     }

```

Here we'll conditionally add `[...]` around the first parameter:

```

195     \bool_if:NTF #1
196     { \__hook_tmp:w \tl_set:Nx { [ \c__hook_hashes_tl 1 ] } }
197     { \__hook_tmp:w \tl_set:Nx { { \c__hook_hashes_tl 1 } } }

```

Then, for every parameter from the second, just add it normally:

```

198     \int_step_inline:nnn { 2 } { \l__hook_patch_num_args_int }
199     { \__hook_tmp:w \tl_put_right:Nx { { \c__hook_hashes_tl ##1 } } }

```

Now, if the command has any parameter token in its definition (then `__hook_exp_not:n` is `\exp_not:n`), call `__hook_double_hashes:n` to double them, and replace every `\c__hook_hashes_tl` by #:

```

200     \tl_set:Nx \l__hook_replace_text_tl
201     { \exp_not:N #2 \exp_not:V \l__hook_replace_text_tl }
202     \tl_set:Nx \l__hook_replace_text_tl
203     {
204         \token_if_eq_meaning:NNTF \__hook_exp_not:n \exp_not:n
205         { \exp_args:NNV \exp_args:No \__hook_double_hashes:n }
206         { \exp_args:NV \exp_not:o }
207         \l__hook_replace_text_tl
208     }

```

And now, set a few auxiliaries for the case that the macro has parameters, so it won't be passed through `\unexpanded` (twice):

```

209     \cs_set_eq:NN \__hook_def_cmd:w \tex_gdef:D
210     \cs_set_eq:NN \__hook_exp_not:NN \prg_do_nothing:
211 }
212 {

```

In the case the macro has no parameters, we'll treat it as a token list and things are much simpler (expansion control looks a bit complicated, but it's just a pair of `\exp_not:n` preventing another `\exp_not:n` from expanding):

```

213     \tl_clear:N \l__hook_param_text_tl
214     \tl_set_eq:NN \l__hook_replace_text_tl #2
215     \cs_set_eq:NN \__hook_def_cmd:w \tex_xdef:D
216     \cs_set:Npn \__hook_exp_not:NN ##1 { \exp_not:N ##1 \exp_not:N }
217 }

```

Before redefining, we need to also get the prefixes used when defining the command. Here we ensure that the `\escapechar` is printable, otherwise a macro defined with prefixes `\protected\long` will have it `\meaning` printed as `protectedlong`, making life unnecessarily complicated. Here the `\escapechar` is changed to `/`, then we loop between pairs of `/.../` extracting the prefixes.

```

218     \group_begin:
219     \int_set:Nn \tex_escapechar:D { '\ / }
220     \use:x
221     {
222     \group_end:
223     \tl_set:Nx \exp_not:N \l__hook_patch_prefixes_tl
224     { \exp_not:N \__hook_make_prefixes:w \cs_prefix_spec:N #2 / / }
225     }

```

Here we redefine the hook to have the right number of arguments. Disabling the hook, undefining the `parameter` token list then calling `__hook_make_usable:nn` are enough to redefine the hook to the extent we want. Code stored in the hook and other metadata about it are not lost in the process.

```

226     \__hook_disable:n { cmd / #3 / #4 }
227     \cs_undefine:c { c__hook_cmd / #3 / #4_parameter_tl }
228     \__hook_make_usable:nn { cmd / #3 / #4 } { \l__hook_patch_num_args_int }

```

Now call `__hook_redefine_with_hooks:Nnnn` with the macro being redefined in #1, then `\UseHook{cmd/<name>/before}` in #2 or `\UseHook{cmd/<name>/after}` in #3 (one is always empty), and in #4 the *<replacement text>* of the macro.

```

229 \use:e
230 {
231   \_hook_redefine_with_hooks:Nnnn \exp_not:N #2
232   \str_if_eq:nnTF {#4} { after }
233   { \use_ii_i:nn }
234   { \use:nn }
235   { {
236     \_hook_exp_not:NN \exp_not:N \UseHookWithArguments
237     { cmd / #3 / #4 } { \int_use:N \l__hook_patch_num_args_int }
238     \_hook_braced_parameter:n { cmd / #3 / #4 }
239   } }
240   { { } }
241   { \_hook_exp_not:NN \exp_not:V \l__hook_replace_text_tl }
242 }

```

Finally, update the hook code.

```

243   \_hook_update_hook_code:n { cmd / #3 / #4 }
244 }
245 <latexrelease>\EndIncludeInRelease
246 <latexrelease>\IncludeInRelease{2021/06/01}{\_hook_patch_expand_redefine:NNnn}
247 <latexrelease>      {cmd-hooks-with-args}
248 <latexrelease>\cs_gset_protected:Npn \_hook_patch_expand_redefine:NNnn #1 #2 #3 #4
249 <latexrelease> {
250 <latexrelease>   \_hook_patch_debug:x { +++command-can-be-patched-without-rescanning }
251 <latexrelease>   \int_set:Nn \l__hook_patch_num_args_int
252 <latexrelease>   {
253 <latexrelease>     \exp_args:Nf \str_count:n { \_kernel_cs_parameter_spec:N #2 } / 2
254 <latexrelease>     \bool_if:NT #1 { -1 }
255 <latexrelease>   }
256 <latexrelease>   \int_compare:nNnTF { \l__hook_patch_num_args_int } > { \c_zero_int }
257 <latexrelease>   {
258 <latexrelease>     \tl_set:Nx \l__hook_tmpa_tl { \bool_if:NTF #1 { [ ] } { { } } }
259 <latexrelease>     \int_step_inline:nnn { 2 } { \l__hook_patch_num_args_int }
260 <latexrelease>     { \tl_put_right:Nn \l__hook_tmpa_tl { { } } }
261 <latexrelease>     \exp_args:NNo \exp_args:No \_hook_if_has_hash:nTF
262 <latexrelease>     { \exp_after:wN #2 \l__hook_tmpa_tl }
263 <latexrelease>     { \cs_set_eq:NN \_hook_exp_not:n \exp_not:n }
264 <latexrelease>     { \cs_set_eq:NN \_hook_exp_not:n \use:n }
265 <latexrelease>     \cs_set_protected:Npn \_hook_tmp:w ##1 ##2
266 <latexrelease>     {
267 <latexrelease>       ##1 \l__hook_param_text_tl { \use:n ##2 }
268 <latexrelease>       ##1 \l__hook_replace_text_tl { \_hook_exp_not:n {##2} }
269 <latexrelease>     }
270 <latexrelease>     \bool_if:NTF #1
271 <latexrelease>     { \_hook_tmp:w \tl_set:Nx { [ \c__hook_hash_tl 1 ] } }
272 <latexrelease>     { \_hook_tmp:w \tl_set:Nx { { \c__hook_hash_tl 1 } } }
273 <latexrelease>     \int_step_inline:nnn { 2 } { \l__hook_patch_num_args_int }
274 <latexrelease>     { \_hook_tmp:w \tl_put_right:Nx { { \c__hook_hash_tl ##1 } } }
275 <latexrelease>     \tl_set:Nx \l__hook_replace_text_tl
276 <latexrelease>     { \exp_not:N #2 \exp_not:V \l__hook_replace_text_tl }
277 <latexrelease>     \tl_set:Nx \l__hook_replace_text_tl
278 <latexrelease>     {
279 <latexrelease>       \token_if_eq_meaning:NNTF \_hook_exp_not:n \exp_not:n
280 <latexrelease>       { \exp_args:NNV \exp_args:No \_hook_double_hashes:n }
281 <latexrelease>       { \exp_args:NV \exp_not:o }

```

```

282 <latexrelease>          \l__hook_replace_text_tl
283 <latexrelease>          }
284 <latexrelease>          \cs_set_eq:NN \__hook_def_cmd:w \tex_gdef:D
285 <latexrelease>          \cs_set_eq:NN \__hook_exp_not:NN \prg_do_nothing:
286 <latexrelease>          }
287 <latexrelease>          {
288 <latexrelease>          \tl_clear:N \l__hook_param_text_tl
289 <latexrelease>          \tl_set_eq:NN \l__hook_replace_text_tl #2
290 <latexrelease>          \cs_set_eq:NN \__hook_def_cmd:w \tex_xdef:D
291 <latexrelease>          \cs_set:Npn \__hook_exp_not:NN ##1 { \exp_not:N ##1 \exp_not:N }
292 <latexrelease>          }
293 <latexrelease>          \group_begin:
294 <latexrelease>          \int_set:Nn \tex_escapechar:D { '\ }
295 <latexrelease>          \use:x
296 <latexrelease>          {
297 <latexrelease>          \group_end:
298 <latexrelease>          \tl_set:Nx \exp_not:N \l__hook_patch_prefixes_tl
299 <latexrelease>          { \exp_not:N \__hook_make_prefixes:w \cs_prefix_spec:N #2 / / }
300 <latexrelease>          }
301 <latexrelease>          \use:x
302 <latexrelease>          {
303 <latexrelease>          \__hook_redefine_with_hooks:Nnnn \exp_not:N #2
304 <latexrelease>          \str_if_eq:nnTF {#4} { after }
305 <latexrelease>          { \use_ii_i:nn }
306 <latexrelease>          { \use:nn }
307 <latexrelease>          { { \__hook_exp_not:NN \exp_not:N \UseHook { cmd / #3 / #4 } } }
308 <latexrelease>          { { } }
309 <latexrelease>          { \__hook_exp_not:NN \exp_not:N \l__hook_replace_text_tl }
310 <latexrelease>          }
311 <latexrelease>          }
312 <latexrelease> \EndIncludeInRelease

```

Now that all the needed tools are ready, without further ado we'll redefine the command. The definition uses the prefixes gathered in `\l__hook_patch_prefixes_tl`, a primitive `__hook_def_cmd:w` (which is `\tex_gdef:D` or `\tex_xdef:D`) to avoid adding extra prefixes, and the `<parameter text>` from `\l__hook_param_text_tl`.

Then finally, in the body of the definition, we insert `#2`, which is `cmd/#1/before` or empty, `#4` which is the `<replacement text>`, and `#3` which is `cmd/#1/after` or empty.

```

313 \cs_new_protected:Npn \__hook_redefine_with_hooks:Nnnn #1 #2 #3 #4
314 {
315   \l__hook_patch_prefixes_tl
316   \exp_after:wN \__hook_def_cmd:w
317   \exp_after:wN #1 \l__hook_param_text_tl
318   { #2 #4 #3 }
319 }

```

Here's the auxiliary that makes the prefix control sequences for the redefinition. Each item has to be `\tl_trim_spaces:n'd` because the last item (and not any other) has a trailing space.

```

320 \cs_new:Npn \__hook_make_prefixes:w / #1 /
321 {
322   \tl_if_empty:nF {#1}
323   {
324     \exp_not:c { tex_ \tl_trim_spaces:n {#1} :D }

```

```

325     \__hook_make_prefixes:w /
326   }
327 }

```

(End of definition for `__hook_patch_expand_redefine:Nn`, `__hook_redefine_with_hooks:Nnn`, and `__hook_make_prefixes:w`.)

Here are some auxiliaries for the contraption above.

`__hook_if_has_hash_p:n` `__hook_if_has_hash:nTF` searches the token list #1 for a catcode 6 token, and if any is found, it returns `true`, and `false` otherwise. The searching doesn't care about preserving groups or spaces: we can ignore those safely (braces are removed) so that searching is as fast as possible.

```

328 \prg_new_conditional:Npnn \__hook_if_has_hash:n #1 { TF }
329 { \__hook_if_has_hash:w #1 ## \s__hook_mark }
330 \cs_new:Npn \__hook_if_has_hash:w #1
331 {
332   \tl_if_single_token:nTF {#1}
333   {
334     \token_if_eq_catcode:NNTF ## #1
335     { \__hook_if_has_hash_check:w }
336     { \__hook_if_has_hash:w }
337   }
338   { \__hook_if_has_hash:w #1 }
339 }
340 \cs_new:Npn \__hook_if_has_hash_check:w #1 \s__hook_mark
341 { \tl_if_empty:nTF {#1} { \prg_return_false: } { \prg_return_true: } }

```

(End of definition for `__hook_if_has_hash:nTF`, `__hook_if_has_hash:w`, and `__hook_if_has_hash_check:w`.)

`__hook_double_hashes:n` `__hook_double_hashes:w` loops through the token list #1 and duplicates any catcode 6 token, and expands tokens `\ifx-equal` to `\c__hook_hashes_tl`, and leaves all other tokens `\notexpanded` with `\exp_not:N`. Unfortunately pairs of explicit catcode 1 and catcode 2 character tokens are normalised to `{_1` and `}_1` because it's not feasible to expandably detect the character code (*maybe* it could be done using something along the lines of <https://tex.stackexchange.com/a/527538>, but it's far too much work for close to zero benefit).

`__hook_double_hashes:w` is the tail-recursive loop macro, that tests which of the three types of item is in the head of the token list.

```

342 \cs_new:Npn \__hook_double_hashes:n #1
343 { \__hook_double_hashes:w #1 \q__hook_recursion_tail \q__hook_recursion_stop }
344 \cs_new:Npn \__hook_double_hashes:w #1 \q__hook_recursion_stop
345 {
346   \tl_if_head_is_N_type:nTF {#1}
347   { \__hook_double_hashes_output:N }
348   {
349     \tl_if_head_is_group:nTF {#1}
350     { \__hook_double_hashes_group:n }
351     { \__hook_double_hashes_space:w }
352   }
353   #1 \q__hook_recursion_stop
354 }

```

`__hook_double_hashes_output:N` checks for the end of the token list, then checks if the token is `\c__hook_hashes_tl`, and if so just leaves it.

```

355 \cs_new:Npn \__hook_double_hashes_output:N #1
356 {
357   \if_meaning:w \q__hook_recursion_tail #1
358   \__hook_double_hashes_stop:w
359   \fi:
360   \if:w ?
361     \if_meaning:w \c__hook_hash_tl #1 ! \fi:
362     \if_meaning:w \c__hook_hashes_tl #1 ! \fi:
363     ?
364   \else:

```

(this `\use_i:nnnn` uses `\fi:` and consumes `\use:n`, the whole `\if_catcode:w` block, and the `\exp_not:N`, leaving just `#1` which is `\c__hook_hashes_tl`.)

```

365   \use_i:nnnn
366   \fi:
367   \use:n
368   {

```

If `#1` is not `\c__hook_hashes_tl`, then check if its catcode is 6, and if so, leave it doubled in `\exp_not:n` and consume the following `\exp_not:N #1`.

```

369     \if_catcode:w ## \exp_not:N #1
370     \exp_after:wN \use_ii:nnnn
371     \fi:
372     \use_none:n
373     { \exp_not:n { #1 #1 } }
374   }

```

If both previous tests returned `false`, then leave the token unexpanded and resume the loop.

```

375   \exp_not:N #1
376   \__hook_double_hashes:w
377 }
378 \cs_new:Npn \__hook_double_hashes_stop:w #1 \q__hook_recursion_stop { \fi: }

```

Dealing with spaces and grouped tokens is trivial:

```

379 \cs_new:Npn \__hook_double_hashes_group:n #1
380 { { \__hook_double_hashes:n {#1} } \__hook_double_hashes:w }
381 \exp_last_unbraced:NNo
382 \cs_new:Npn \__hook_double_hashes_space:w \c_space_tl
383 { ~ \__hook_double_hashes:w }

```

(End of definition for `__hook_double_hashes:n` and others.)

4.5.2 Patching by retokenization

At this point we've drained the possibilities of patching a command by expansion-and-redefinition, so we have to resort to patching by retokenizing the command. Patching by retokenization is done by getting the `\meaning` of the command, doing the necessary manipulations on the generated string, and the retokenizing that again by using `\scantokens`.

Patching by retokenization is definitely a riskier business, because it relies that the tokens printed by `\meaning` produce the exact same tokens as the ones in the original

definition. That is, the catcode régime must be exactly(ish) the same, and there is no way of telling except by trial and error.

`__hook_retokenize_patch:Nnn` This is the macro that will control the whole process. First we'll try out one final, rather trivial case, of a command with no arguments; that is, a token list. This case can be patched with the expand-and-redefine routine but it has to be the very last case tested for, because most (all?) robust commands start with a top-level macro with no arguments, so testing this first would short-circuit `\robust@command@act` and the top-level macros would be incorrectly patched. In that case, we just check if the `\cs_parameter_spec:N` is empty, and call `__hook_patch_expand_redefine:NNnn`.

```

384 \cs_new_protected:Npn \__hook_retokenize_patch:Nnn #1 #2 #3
385   {
386     \str_if_eq:eeTF { \__kernel_cs_parameter_spec:N #1 } { }
387     { \__hook_patch_expand_redefine:NNnn \c_false_bool #1 {#2} {#3} }
388     {
389       \__hook_patch_debug:x { ..~command~can~only~be~patched~by~rescanning }

```

Otherwise, we start the actual patching by retokenization job. The code calls `__hook_try_patch_with_catcodes:Nnnnw` with a different catcode setting:

- The current catcode setting;
- Switching the catcode of @;
- Switching the expl3 syntax on or off;
- Both of the above.

If patching succeeds, `__hook_try_patch_with_catcodes:Nnnnw` has the side-effect of patching the macro #1 (which may be an internal from the command whose name is #2).

```

390     \tl_set:Nx \l__hook_tmpa_tl
391     {
392       \int_compare:nNnTF { \char_value_catcode:n {'\@ } } = { 12 }
393       { \exp_not:N \makeatletter } { \exp_not:N \makeatother }
394     }
395     \tl_set:Nx \l__hook_tmpb_tl
396     {
397       \bool_if:NTF \l__kernel_expl_bool
398       { \ExplSyntaxOff } { \ExplSyntaxOn }
399     }
400     \use:x
401     {
402       \exp_not:N \__hook_try_patch_with_catcodes:Nnnnw
403       \exp_not:n { #1 {#2} {#3} }
404       { \prg_do_nothing: }
405       { \exp_not:V \l__hook_tmpa_tl } % @
406       { \exp_not:V \l__hook_tmpb_tl } % _:
407       {
408         \exp_not:V \l__hook_tmpa_tl % @
409         \exp_not:V \l__hook_tmpb_tl % _:
410       }
411     }
412     \q_recursion_tail \q_recursion_stop

```

If no catcode setting succeeds, give up and raise an error. The command isn't changed in any way in that case.

```

413     {
414         \msg_error:nxxx { hooks } { cant-patch }
415         { \c_backslash_str #2 } { retok }
416     }
417 }
418 }

```

(End of definition for `__hook_retokenize_patch:Nnn`.)

`__hook_try_patch_with_catcodes:Nnnnw`

This function is a simple wrapper around `__hook_cmd_if_scanable:NnTF` and `__hook_patch_retokenize:Nnnn` if the former returns `<true>`, plus some debug messages.

```

419 <latexrelease> \IncludeInRelease{2023/06/01}{\__hook_try_patch_with_catcodes:Nnnnw}
420 <latexrelease> {cmd-hooks-with-args}
421 \cs_new_protected:Npn \__hook_try_patch_with_catcodes:Nnnnw #1 #2 #3 #4
422 {
423     \quark_if_recursion_tail_stop_do:nn {#4} { \use:n }
424     \__hook_patch_debug:x { ++~trying~to~patch~by~retokenization }
425     \__hook_cmd_if_scanable:NnTF {#1} {#4}
426     {
427         \__hook_patch_debug:x { ++~macro~can~be~retokenized~cleanly }
428         \__hook_patch_debug:x { ==~retokenizing~macro~now }
429         \__hook_patch_retokenize:Nnnn #1 { cmd / #2 / #3 } {#3} {#4}
430         \use_i_delimit_by_q_recursion_stop:nw \use_none:n
431     }
432     {
433         \__hook_patch_debug:x { ---macro~cannot~be~retokenized~cleanly }
434         \__hook_try_patch_with_catcodes:Nnnnw #1 {#2} {#3}
435     }
436 }
437 <latexrelease> \EndIncludeInRelease
438 <latexrelease> \IncludeInRelease{2021/06/01}{\__hook_try_patch_with_catcodes:Nnnnw}
439 <latexrelease> {cmd-hooks-with-args}
440 <latexrelease> \cs_gset_protected:Npn \__hook_try_patch_with_catcodes:Nnnnw #1 #2 #3 #4
441 <latexrelease> {
442 <latexrelease>     \quark_if_recursion_tail_stop_do:nn {#4} { \use:n }
443 <latexrelease>     \__hook_patch_debug:x { ++~trying~to~patch~by~retokenization }
444 <latexrelease>     \__hook_cmd_if_scanable:NnTF {#1} {#4}
445 <latexrelease>     {
446 <latexrelease>         \__hook_patch_debug:x { ++~macro~can~be~retokenized~cleanly }
447 <latexrelease>         \__hook_patch_debug:x { ==~retokenizing~macro~now }
448 <latexrelease>         \__hook_patch_retokenize:Nnnn #1 {#2} {#3} {#4}
449 <latexrelease>         \use_i_delimit_by_q_recursion_stop:nw \use_none:n
450 <latexrelease>     }
451 <latexrelease>     {
452 <latexrelease>         \__hook_patch_debug:x { ---macro~cannot~be~retokenized~cleanly }
453 <latexrelease>         \__hook_try_patch_with_catcodes:Nnnnw #1 {#2} {#3}
454 <latexrelease>     }
455 <latexrelease> }
456 <latexrelease> \EndIncludeInRelease

```

(End of definition for `__hook_try_patch_with_catcodes:Nnnnw`.)

`\kerneltmpDoNotUse` This is an oddity required to be safe (as safe as reasonably possible) when patching the command. The entirety of

```
⟨prefixes⟩ \def ⟨cs⟩ ⟨parameter text⟩ {⟨replacement text⟩}
```

will go through `\scantokens`. The `⟨parameter text⟩` and `⟨replacement text⟩` are what we are trying to retokenize, so not much worry there. The other items, however, should “just work”, so some care is needed to not use too fancy catcode settings. Therefore we can’t use an `expl3`-named macro for `⟨cs⟩`, nor the `expl3` versions of `\def` or the `⟨prefixes⟩`. That is why the definitions that will eventually go into `\scantokens` will use the oddly (but hopefully clearly)-named `\kerneltmpDoNotUse`:

```
457 \cs_new_eq:NN \kerneltmpDoNotUse !
```

PhO: Maybe this can be avoided by running the `⟨parameter text⟩` and the `⟨replacement text⟩` separately through `\scantokens` and then putting everything together at the end.

(End of definition for `\kerneltmpDoNotUse`.)

`_hook_patch_required_catcodes:` Here are the catcode settings that are *mandatory* when retokenizing commands. These are the minimum necessary settings to perform the definitions: they identify control sequences, which must be escaped with `_0`, delimit the definition with `{_1` and `}_2`, and mark parameters with `#_6`. Everything else may be changed, but not these.

```
458 \cs_new_protected:Npn \_hook_patch_required_catcodes:
459 {
460   \char_set_catcode_escape:N \_
461   \char_set_catcode_group_begin:N \{
462   \char_set_catcode_group_end:N \}
463   \char_set_catcode_parameter:N \#
464   % \int_set:Nn \tex_endlinechar:D { -1 }
465   % \int_set:Nn \tex_newlinechar:D { -1 }
466 }
```

PhO: `etoolbox` sets the `\endlinechar` and `\newlinechar` when patching, but as far as I tested these didn’t make much of a difference, so I left them out for now. Maybe `\newlinechar=-1` avoids a space token being added after the definition.

PhO: If the patching is split by `⟨parameter text⟩` and `⟨replacement text⟩`, then only `#` will have to stay in that list.

PhO: Actually now that we patch `\UseHook{cmd/foo/before}`, all the tokens there need to have the right catcodes, so this list now includes all lowercase letters, `U` and `H`, the slash, and whatever characters in the command name... sigh...

(End of definition for `_hook_patch_required_catcodes:`.)

`_hook_cmd_if_scanable:NnTF` Here we’ll do a quick test if the command being patched can in fact be retokenized with the specific catcode setting without changing in meaning. The test is straightforward:

1. apply `\meaning` to the command;
2. split the `⟨prefixes⟩`, `⟨parameter text⟩` and `⟨replacement text⟩` and arrange them as

```
⟨prefixes⟩ \def \kerneltmpDoNotUse ⟨parameter text⟩ {⟨replacement text⟩}
```

3. rescan that with the given catcode settings, and do the definition; then finally

4. compare `\kerneltmpDoNotUse` with the original command.

If both are `\ifx-equal`, the command can be safely patched.

```

467 \prg_new_protected_conditional:Npnn \__hook_cmd_if_scanable:Nn #1 #2 { TF }
468 {
469   \cs_set_eq:NN \kerneltmpDoNotUse \scan_stop:
470   \cs_set_eq:NN \__hook_tmp:w \scan_stop:
471   \use:x
472   {
473     \cs_set:Npn \__hook_tmp:w
474       #####1 \tl_to_str:n { macro: } #####2 -> #####3 \s__hook_mark
475       { #####1 \def \kerneltmpDoNotUse #####2 {#####3} }
476     \tl_set:Nx \exp_not:N \l__hook_tmpa_tl
477       { \exp_not:N \__hook_tmp:w \token_to_meaning:N #1 \s__hook_mark }
478   }
479   \tl_rescan:nV { #2 \__hook_patch_required_catcodes: } \l__hook_tmpa_tl
480   \token_if_eq_meaning:NNTF #1 \kerneltmpDoNotUse
481     { \prg_return_true: }
482     { \prg_return_false: }
483 }

```

(End of definition for `__hook_cmd_if_scanable:NnTF`.)

`__hook_guess_arg_count:NN` Looks at the parameter text of a macro, and counts the parameters by looking at the number after a #, and checking if they are sequential. This macro assumes that all parameters are marked with hashes, and not other characters, and that there is no “trick parameter”.

```

484 <latexrelease>\IncludeInRelease{2023/06/01}{\__hook_guess_arg_count:NN}
485 <latexrelease> {cmd~hooks~with~args}
486 \cs_new_protected:Npn \__hook_guess_arg_count:NN #1
487 {
488   \exp_after:wN \__hook_guess_arg_count:wN
489   \token_to_meaning:N #1 \s__hook_mark
490 }
491 \exp_last_unbraced:NNNNo
492 \cs_new_protected:Npx \__hook_guess_arg_count:wN
493   #1 { \tl_to_str:n { macro: } } #2 \s__hook_mark #3
494 {
495   \int_set:Nn #3
496   {
497     \exp_not:N \__hook_guess_arg_count:nw { 0 } #2
498     \c_hash_str 0 \s__hook_mark
499   }
500 }
501 \use:e
502 { \cs_new:Npn \exp_not:N \__hook_guess_arg_count:nw #1 #2 \c_hash_str #3 }
503 {
504   \int_compare:nNnTF { #1 + 1 } = {#3}
505     { \__hook_guess_arg_count:nw {#3} }
506     { #1 \__hook_use_none_delimit_by_s_mark:w }
507 }
508 <latexrelease>\EndIncludeInRelease
509 <latexrelease>\IncludeInRelease{2021/06/01}{\__hook_guess_arg_count:NN}
510 <latexrelease> {cmd~hooks~with~args}

```

```

511 <latexrelease>\cs_undefine:N \__hook_guess_arg_count:NN
512 <latexrelease>\EndIncludeInRelease

```

(End of definition for __hook_guess_arg_count:NN, __hook_guess_arg_count:wN, and __hook_guess_arg_count:nw.)

__hook_patch_retokenize:Nnnn

Then, if __hook_cmd_if_scanable:NnTF returned true, we can go on and patch the command.

```

513 <latexrelease>\IncludeInRelease{2023/06/01}{\__hook_patch_retokenize:Nnnn}
514 <latexrelease>          {cmd-hooks-with-args}
515 \cs_new_protected:Npn \__hook_patch_retokenize:Nnnn #1 #2 #3 #4
516   {

```

Here, when patching by retokenization, we can only guess the number of arguments of the macro.

```

517   \__hook_guess_arg_count:NN #1 \l__hook_patch_num_args_int

```

Then we redefine the hook to have the right number of arguments. Disabling the hook, undefining the `parameter` token list then calling `__hook_make_usable:nn` are enough to redefine the hook to the extent we want. Code stored in the hook and other metadata about it are not lost in the process.

```

518   \__hook_disable:n {#2}
519   \cs_undefine:c { c_hook_#2_parameter_tl }
520   \__hook_make_usable:nn {#2} { \l__hook_patch_num_args_int }
521   \tl_set:Ne \l__hook_tmpa_tl
522     { \exp_args:Ne \tl_to_str:n { \__hook_braced_parameter:n {#2} } }
523   \use:x
524   {
525     \str_replace_all:Nnn \exp_not:N \l__hook_tmpa_tl
526     { #### } { \c_hash_str }
527   }

```

Then, make some things `\relax` to avoid lots of `\noexpand` below.

```

528   \cs_set_eq:NN \kerneltmpDoNotUse \scan_stop:
529   \cs_set_eq:NN \__hook_tmp:w \scan_stop:
530   \use:x
531   {

```

Now we'll define `__hook_tmp:w` such that it splits the `\meaning` of the macro (`#1`) into its three parts:

```

####1. <prefixes>

####2. <parameter text>

####3. <replacement text>

```

and arrange that a complete definition, then place the `before` or `after` hooks around the `<replacement text>`: accordingly.

```

532   \cs_set:Npn \__hook_tmp:w
533     ####1 \tl_to_str:n { macro: } ####2 -> ####3 \s__hook_mark
534   {
535     ####1 \def \kerneltmpDoNotUse ####2
536     {
537       \str_if_eq:nnT {#3} { before }
538     }

```

```

539         \token_to_str:N \UseHookWithArguments {#2}
540         { \int_use:N \l__hook_patch_num_args_int }
541         \l__hook_tmpa_tl
542     }
543     #####3
544     \str_if_eq:nnT {#3} { after }
545     {
546         \token_to_str:N \UseHookWithArguments {#2}
547         { \int_use:N \l__hook_patch_num_args_int }
548         \l__hook_tmpa_tl
549     }
550 }
551 }

```

Now we just have to get the `\meaning` of the command being patched and pass it through the meat grinder above.

```

552     \tl_set:Nx \exp_not:N \l__hook_tmpa_tl
553     { \exp_not:N \__hook_tmp:w \token_to_meaning:N #1 \s__hook_mark }
554 }

```

Now rescan with the given catcode settings (overridden by the `__hook_patch_required_catcodes:`), and implicitly (by using the rescanned token list) carry out the definition from above.

```

555     \tl_rescan:nV { #4 \__hook_patch_required_catcodes: } \l__hook_tmpa_tl

```

And to close, copy the newly-defined command into the old name and the patching is finally completed:

```

556     \cs_gset_eq:NN #1 \kerneltmpDoNotUse

```

Finally, update the hook code.

```

557     \__hook_update_hook_code:n {#2}
558 }
559 <latexrelease> \EndIncludeInRelease
560 <latexrelease> \IncludeInRelease{2021/06/01}{\__hook_patch_retokenize:Nnnn}
561 <latexrelease> {cmd-hooks-with-args}
562 <latexrelease> \cs_gset_protected:Npn \__hook_patch_retokenize:Nnnn #1 #2 #3 #4
563 <latexrelease> {
564 <latexrelease>     \cs_set_eq:NN \kerneltmpDoNotUse \scan_stop:
565 <latexrelease>     \cs_set_eq:NN \__hook_tmp:w \scan_stop:
566 <latexrelease>     \use:x
567 <latexrelease>     {
568 <latexrelease>         \cs_set:Npn \__hook_tmp:w
569 <latexrelease>             #####1 \tl_to_str:n { macro: } #####2 -> #####3 \s__hook_mark
570 <latexrelease>             {
571 <latexrelease>                 #####1 \def \kerneltmpDoNotUse #####2
572 <latexrelease>                 {
573 <latexrelease>                     \str_if_eq:nnT {#3} { before }
574 <latexrelease>                     { \token_to_str:N \UseHook { cmd / #2 / #3 } }
575 <latexrelease>                     #####3
576 <latexrelease>                     \str_if_eq:nnT {#3} { after }
577 <latexrelease>                     { \token_to_str:N \UseHook { cmd / #2 / #3 } }
578 <latexrelease>                 }
579 <latexrelease>             }
580 <latexrelease>         \tl_set:Nx \exp_not:N \l__hook_tmpa_tl
581 <latexrelease>         { \exp_not:N \__hook_tmp:w \token_to_meaning:N #1 \s__hook_mark }

```

```

582 <latexrelease>      }
583 <latexrelease>      \tl_rescan:nV { #4 \__hook_patch_required_catcodes: } \l__hook_tmpa_tl
584 <latexrelease>      \cs_gset_eq:NN #1 \kerneltmpDoNotUse
585 <latexrelease>    }
586 <latexrelease>\EndIncludeInRelease

```

(End of definition for __hook_patch_retokenize:Nnnn.)

4.6 Messages

```

587 <latexrelease>\IncludeInRelease{2023/06/01}{wrong-cmd-hook}%
588 <latexrelease>      {Standardise-generic-hook-names}
589 <latexrelease>\EndIncludeInRelease
590 <latexrelease>\IncludeInRelease{2021/06/01}{wrong-cmd-hook}%
591 <latexrelease>      {Standardise-generic-hook-names}
592 <latexrelease>\msg_new:nnnn { hooks } { wrong-cmd-hook }
593 <latexrelease> {
594 <latexrelease>   Generic-hook~'cmd/#1/#2'~is~invalid.
595 <latexrelease>%   The-hook~should~be~'cmd/#1/before'~or~'cmd/#1/after'.
596 <latexrelease> }
597 <latexrelease> {
598 <latexrelease>   You~tried~to~add~a~generic~hook~to~command~\iow_char:N \#1,~but~'#2'~
599 <latexrelease>   is~an~invalid~component.~Only~'before'~or~'after'~are~allowed.
600 <latexrelease> }
601 <latexrelease>\EndIncludeInRelease
602 \msg_new:nnnn { hooks } { cant-patch }
603 {
604   Generic-hooks~cannot~be~added~to~'#1'.
605 }
606 {
607   You~tried~to~add~a~hook~to~'#1',~but~LaTeX~was~unable~to~
608   patch~the~command~because~it~\__hook_unpatchable_cases:n {#2}.
609 }
610 \cs_new:Npn \__hook_unpatchable_cases:n #1
611 {
612   \str_case:nn {#1}
613   {
614     { undef } { doesn't~exist }
615     { macro } { is~not~a~macro }
616     { expl3 } { is~a~private~expl3~macro }
617     { retok } { can't~be~retokenized~cleanly }
618   }
619 }
620 <latexrelease>\IncludeInRelease{0000/00/00}{ltxcmdhooks}%
621 <latexrelease>      {The~hook~management~system~for~commands}
622 <latexrelease>

```

The command `__hook_cmd_begindocument_code:` is used in an internal hook, so we need to make sure it has a harmless definition after rollback as that will not remove it from the kernel hook.

```

623 <latexrelease>\cs_set_eq:NN \__hook_cmd_begindocument_code: \prg_do_nothing:
624 <latexrelease>
625 <latexrelease>\EndModuleRelease
626 \ExplSyntaxOff

```

627 </2kernel | latexrelease>
628 <@@=>

File 10

ltsockets.dtx

Abstract

This code implements sockets which are places in the code into which predeclared chunks of code (plugs) can be placed. Both the sockets and the plugs are “named” and each socket is assigned exactly one plug at any given time.

1 Introduction

A L^AT_EX source file is transformed into a typeset document by executing code for each command or environment in the document source. Through various steps this code transforms the input and eventually generates typeset output appearing in a “galley” from which individual pages are cut off in an asynchronous way. This page generating process is normally not directly associated with commands in the input²¹ but is triggered whenever the galley has received enough material to form another page (giving current settings).

As part of this transformation input data may get stored in some form and later reused, for example, as part of the output routine processing.

2 Configuration of the transformation process

There are three different major methods offered by L^AT_EX to configure the transformation process:

- through the template mechanism,
- through the hook mechanism, or
- through sockets and plugs.

They offer different possibilities (with different features and limitations) and are intended for specific use cases, though it is possible to combine them.

2.1 The template mechanism

The template mechanism is intended for more complex document-level elements (e.g., headings such as `\section` or environments like `itemize`). The template code implements the overall processing logic for such an element and offers a set of parameters to influence the final result.

The document element is then implemented by (a) selecting a suitable template (there may be more than one available for the kind of document element) and (b) by setting its parameters to desired values. This then forms a so-called instance which is executed when the document element is found in the source.

By altering the parameter values (in a document class or in the document preamble) or, if more drastic layout changes are desired, by selecting a different template and

²¹Excepts for directives such as `\newpage`.

then adjusting its parameters, a wide variety of layouts can be realized through simple configuration setups without the need to develop new code.

The target audience of this method are therefore document class developers or users who wish to alter an existing layout (implemented by a document class) in certain (minor) ways.

The template mechanism is currently documented as part of the `xtemplate` package and one more elaborate implementation can be found as part of the `latex-lab` code for lists (to be documented further).

2.2 The hook mechanism

Hooks are places in the kernel code (or in packages) that offer packages the possibility to inject additional code at specific points in the processing in a controlled way without the need to replace the existing code block (and thereby overwriting modifications/extensions made by other packages). The target audience is therefore mainly package developers, even though some hooks can be useful for document authors.

Obviously, what can reasonably be added into a hook depends on the individual hook (hopefully documented as part of the hook documentation), but in general the idea behind hooks is that more than one package could add code into the hook at the same time. Perhaps the most famous hook (that L^AT_EX had for a very long time) is `begindocument` into which many packages add code to through `\AtBeginDocument{<code>}` (which is nowadays implemented as a shorthand for `\AddToHook{begindocument}{<code>}`). To resolve possible conflicts between injections by different packages there is a rule mechanism by which code chunks in a hook can be ordered in a certain way and by which incompatible packages can be detected if a resolution is impossible.

In contrast to template code, there is no standard configuration method through parameters for hooks, i.e., the code added to a hook “is” the configuration. If it wants to provide for configuration through parameters it has to also provide its own method to set such parameters in some way. However, in that case it is likely that using a hook is not the right approach and the developer better calls a template instance instead which then offers configuration through a key/value interface.

In most cases, hooks do not take any arguments as input. Instead, the data that they can (and are allowed to) access depends on the surrounding context.

For example, the various hooks available during the page shipout process in L^AT_EX’s output routine can (and have to) access the accumulated page material stored in a box named `\ShipoutBox`. This way, code added to, say, the `shipout/before` hook could access the page content, alter it, and then write it back into `\ShipoutBox` and any other code added to this hook could then operate on the modified content. Of course, for such a scheme to work the code prior to executing the hook would need to setup up data in appropriate places and the hook documentation would need to document what kind of storage can be accessed (and possibly altered) by the hook.

There are also hooks that take arguments (typically portions of document data) and in that case the hook code can access these arguments through `#1`, `#2`, etc.

The hook mechanism is documented in `lthooks-doc.pdf`.

2.3 The socket mechanism

In some cases there is code that implements a certain programming logic (for example, combining footnotes, floats, and the text for the current page to be shipped out) and

if this logic should change (e.g., footnotes to be placed above bottom floats instead of below) then this whole code block needs to be replaced with different code.

In theory, this could be implemented with templates, i.e., the code simply calls some instance that implements the logic and that instance is altered by selecting a different templates and/or adjusting their parameters. However, in many cases customization through parameters is overkill in such a case (or otherwise awkward, because parameterization is better done on a higher level instead of individually for small blocks of code) and using the template mechanism just to replace one block of code with a different one results in a fairly high performance hit. It is therefore usually not a good choice.

In theory, it would also be possible to use a hook, but again that is basically a misuse of the concept, because in this use case there should never be more than one block of code inside the hook; thus, to alter the processing logic one would need to set up rules that replace code rather than (as intended) execute all code added to the hook.

For this reason L^AT_EX now offers a third mechanism: “sockets” into which one can place exactly one code block — a “plug”.

In a nutshell: instead of having a fixed code block somewhere as part of the code, implementing a certain programming logic there is a reference to a named socket at this point. This is done by first declaring the named socket with:

```
\NewSocket{<socket-name>}{<number-of-inputs>}
```

This is then referenced at the point where the replaceable code block should be executed with:

```
\UseSocket{<socket-name>}
```

or, if the socket should take a number of inputs (additional arguments beside the name) with

```
\UseSocket{<socket-name>}{<arg1>}...{<argnumber-of-inputs>}
```

In addition, several code blocks (a.k.a. plugs) implementing different logic for this socket are set up, each with a declaration of the form:

```
\NewSocketPlug{<socket-name>}{<socket-plug-name>}{<code>}
```

Finally, one of them is assigned to the socket:

```
\AssignSocketPlug{<socket-name>}{<socket-plug-name>}
```

If the programming logic should change, then all that is necessary is to make a new assignment with `\AssignSocketPlug` to a different `{<socket-plug-name>}`. This assignment obeys scope so that an environment can alter a socket without the need to restore the previous setting manually.

If the socket takes inputs, then those need to be provided to `\UseSocket` and in that case they can be referenced in the `<code>` argument of `\NewSocketPlug` with `#1`, `#2`, etc.

In most cases a named socket is used only in a single place, but there is, of course, nothing wrong with using it in several places, as long as the code in all places is supposed to change in the same way.

2.3.1 Examples

We start by declaring a new socket named `foo` that expects two inputs:

```
\NewSocket{foo}{2}
```

Such a declaration has to be unique across the whole L^AT_EX run. Thus, if another package attempts to use the same name (regardless of the number of inputs) it will generate an error:

```
\NewSocket{foo}{2}
\NewSocket{foo}{1}
```

Both declarations would therefore produce:

```
! LaTeX socket Error: Socket 'foo' already declared!
```

You also get an error if you attempt to declare some socket plug and the socket name is not yet declared, e.g.,

```
\NewSocketPlug{baz}{undeclared}{some code}
```

generates

```
! LaTeX socket Error: Socket 'baz' undeclared!
```

Setting up plugs for the socket is done like this:

```
\NewSocketPlug{foo}{plug-A}
  {\begin{quote}\itshape foo-A: #1!#2\end{quote}}
\NewSocketPlug{foo}{plug-B}
  {\begin{quote}\sffamily foo-B: #2\textsuperscript{2}\end{quote}}
```

This will set up the plugs `plug-A` and `plug-B` for this socket.

We still have to assign one or the other to the socket, thus without doing that the line

```
\UseSocket{foo}{hello}{world}
```

produces nothing because the default plug for sockets with 2 inputs is `noop` (which grabs the additional arguments and throws them away).²²

So let's do the assignment

```
\AssignSocketPlug{foo}{plug-A}
```

and then

```
\UseSocket{foo}{hello}{world}
```

will properly typeset

```
foo-A: hello!world
```

and after

```
\AssignSocketPlug{foo}{plug-B}
```

²²If socket `foo` would have been a socket with one input, then the default plug would be `identity`, in which case the socket input would remain without braces and gets typeset!

and another call to

```
\UseSocket{foo}{hello}{world}
```

we get

```
foo-B: world2
```

If we attempt to assign a plug that was not defined, e.g.,

```
\AssignSocketPlug{foo}{plug-C}
```

then we get an error during the assignment

```
! LaTeX socket Error: Plug 'plug-C' for socket 'foo' undeclared!
```

and the previous assignment remains in place.

To see what is known about a socket and its plugs you can use `\ShowSocket` or `\LogSocket` which displays information similar to this on the terminal or in the transcript file:

```
Socket foo:
  number of inputs = 2
  available plugs = noop, plug-A, plug-B
  current plug = plug-B
  definition = \long macro:#1#2->\begin {quote}\sffamily
foo-B: #2\textsuperscript {2}\end {quote}
```

2.3.2 Details and semantics

In this section we collect some normative statements.

- From a functional point of view sockets are like simple TeX macros, i.e., they expect 0 to 9 mandatory arguments (the socket inputs) and get replaced by their “expansion”
- A socket is “named” and the name consists of ASCII letters [a-z], [A-Z], [0-9], [-/@] only
- Socket names have to be unique, i.e., there can be only one socket named *<name>*. This is ensured by declaring each socket with `\NewSocket`.

However, there is no requirement that sockets and hook names have to be different. In fact, if a certain action that could otherwise be specified as hook code has to be executed always last (or first) one could ensure this by placing a socket (single action) after a hook (or vice versa) and using the same name to indicate the relationship, e.g.,

```
\UseHook{foo}           % different package can add code here
\UseSocket{foo}         % only one package can assign a plug
```

This avoids the need to order the hook code to ensure that something is always last.

- Best practice naming conventions are ... *to be documented*

- A socket has documented inputs which are
 - the positional arguments (if any) with a description of what they contain when used
 - implicit data (registers and other 2e/expl3 data stores) that the socket is allowed to make use of, with a documented description of what they contain (if relevant for the task at hand—no need to describe the whole L^AT_EX universe)
 - information about the state of the T_EX engine (again when relevant), e.g. is called in mmode or vmode or in the output routine or ...
 - ... anything missing?
- A socket has documented results/outputs which can be
 - what kind of data it should write to the current list (if that is part of its task)
 - what kind of registers and other 2e/expl3 data stores it should modify and in what way
 - what kind of state changes it should do (if any)
 - ... *anything else?*
- At any time a socket has one block of code (a plug :-)) associated with it. Such code is itself named and the association is done by linking the socket name to the code name (putting a plug into the socket).
- The name of a plug consists of ASCII letters [a-z], [A-Z], [0-9], [-/@] only.
- Socket plug names have to be unique within on a per socket basis, but it is perfectly allowed (and sensible in some cases) to use the same plug name with different sockets (where based on the sockets' purposes, different actions may be associated with the plug name). For example `noop` is a plug name declared for every socket, yet its action “grab the socket inputs and throw them away” obviously differs depending on how many inputs the socket has.
- When declaring a plug it is stated for which socket it is meant (i.e., its code can only be used with that socket). This means that the same plug name can be used with different sockets referring to different code in each case.
- Configuration of a socket can only be done by linking different code to it. Nevertheless the code linked to it can provide its own means of configuration (but this is outside of the spec).
- Technically execution of a socket (`\UseSocket`) involves
 - doing any house keeping (like writing debugging info, ...);
 - looking up the current code association (what plug is in the socket);
 - executing this code which will pick up the mandatory arguments (happens at this point, not before), i.e., it is like calling a csname defined with


```
\def\foo#1#2...{...#1...#2...}
```
 - do some further house keeping (if needed).
- A socket is typically only used in one place in code, but this is not a requirement, i.e., if the same operation with the same inputs need to be carried out in several places the same named socket can be used.

2.3.3 Command syntax

We give both the L^AT_EX 2_ε and the L3 programming layer command names.

<code>\NewSocket</code>	<code>\NewSocket</code>	<code>{\socket-name}</code>	<code>{\number-of-inputs}</code>
<code>\socket_new:nn</code>	<code>\socket_new:nn</code>	<code>{\socket-name}</code>	<code>{\number-of-inputs}</code>

Declares a new socket with name `\socket-name` having `\number-of-inputs` inputs. There is automatically a plug `noop` declared for it, which does nothing, i.e., it gobbles the socket inputs (if any). This is made the default plug except for sockets with one input which additionally define the plug `identity` and assign that as their default.

This `identity` plug simply returns the socket input without its outer braces. The use case for this plug are situations like this:

```
\UseSocket{taggsupport/footnote}{\code}
```

If tagging is not active and the socket contains the plug `identity` then this returns `\code` without the outer braces and to activate tagging all that is necessary is to change the plug to say `tagpdf` so that it surrounds `\code` by some tagging magic. This is the most common use case for sockets with one input, which is why they have this special default.

The socket documentation should describe its purpose, its inputs and the expected results as discussed above.

The declaration is only allowed at top-level, i.e., not inside a group.

<code>\NewSocketPlug</code>	<code>\NewSocketPlug</code>	<code>{\socket-name}</code>	<code>{\socket-plug-name}</code>	<code>{\code}</code>
<code>\socket_new_plug:nnn</code>	<code>\socket_new_plug:nnn</code>	<code>{\socket-name}</code>	<code>{\socket-plug-name}</code>	<code>{\code}</code>
<code>\socket_set_plug:nnn</code>	<code>\socket_set_plug:nnn</code>	<code>{\socket-name}</code>	<code>{\socket-plug-name}</code>	<code>{\code}</code>

Declares a new plug for socket `\socket-name` that runs `\code` when executing. It complains if the plug was already declared previously.

The form `\socket_set_plug:nnn` changes an existing plug. As this should normally not be necessary, we currently have only an L3 layer name for the few cases it might be useful.

The declarations can be made inside a group and obey scope, i.e., they vanish if the group ends.

<code>\AssignSocketPlug</code>	<code>\AssignSocketPlug</code>	<code>{\socket-name}</code>	<code>{\socket-plug-name}</code>
<code>\socket_assign_plug:nn</code>	<code>\socket_assign_plug:nn</code>	<code>{\socket-name}</code>	<code>{\socket-plug-name}</code>

Assigns the plug `\socket-plug-name` to the socket `\socket-name`. It errors if either socket or plug is not defined.

The assignment is local, i.e., it obeys scope.

<code>\UseSocket</code>	<code>\UseSocket</code>	<code>{\socket-name}</code>
<code>\socket_use:nw</code>	<code>\socket_use:nnn</code>	<code>{\socket-name}</code> <code>{\socket-arg₁}</code> <code>{\socket-arg₂}</code>
<code>\socket_use:n</code>	Executes the socket <code>\socket-name</code> by retrieving the <code>\code</code> of the current plug assigned to the socket. This is the only command that would appear inside macro code in packages.	
<code>\socket_use:nn</code>	For performance reasons there is no explicit check that the socket was declared!	
<code>\socket_use:nnn</code>	The different L3 programming layer commands are really doing the same thing: they grab as many arguments as defined as inputs for the socket and then pass them to the plug. The different names are only there to make the code more readable, i.e., to indicate how many arguments are grabbed in total (note that no runtime check is made to verify that this is actually true). We only provide them for sockets with up to 3 inputs (most likely those with zero or one input would have been sufficient). If you happen to have a socket with more inputs, use <code>\socket_use:nw</code> .	
<code>\socket_use:nnnn</code>		

<code>\socket_use_expandable:nw</code>	<code>\socket_use_expandable:n</code>	<code>{\socket-name}</code>
<code>\socket_use_expandable:n</code>	<code>\socket_use_expandable:n</code>	<code>*</code>

Fully expandable variant of `\socket_use:n`. This can be used in macro code to retrieve code from sockets which need to appear in an expandable context.

This usually requires the plug to only contain expandable code and should therefore only be used for sockets which are clearly documented to be used in an expandable context. This command does not print any debugging info when `\DebugSocketsOn` is active and should therefore be avoided whenever possible.

For performance reasons there is no explicit check that the socket was declared!

<code>\ShowSocket</code>	<code>\ShowSocket</code>	<code>{\socket-name}</code>
<code>\LogSocket</code>	<code>\socket_show:n</code>	<code>{\socket-name}</code>
<code>\socket_show:n</code>	Displays information about the socket <code>\socket-name</code> and its state then stops and waits for further instructions — at the moment some what rudimentary.	
<code>\socket_log:n</code>	<code>\LogSocket</code> and <code>\socket_log:n</code> only differ in that they don't stop.	

It is sometimes necessary/helpful to know if a particular socket or plug exists (or is assigned to a certain socket) and based on that take different actions.

<code>\IfSocketExistsTF</code>	<code>\IfSocketExistsTF</code>	<code>{\socket-name}</code> <code>{\true code}</code> <code>{\false code}</code>
<code>\socket_if_exist:nnTF</code>	<code>*</code>	If socket <code>\socket-name</code> exists then execute <code>\true code</code> otherwise <code>\false code</code> . Variants with only T or F are also available.

<code>\IfSocketPlugExistsTF</code>	<code>\IfSocketPlugExistsTF</code>	<code>{\socket-name}</code> <code>{\plug-name}</code>
<code>\socket_if_plug_exist:nnTF</code>	<code>*</code>	<code>{\true code}</code> <code>{\false code}</code>

If plug `\plug-name` for socket `\socket-name` exists then execute `\true code` otherwise `\false code`. Variants with only T or F are also available.

<code>\IfSocketPlugAssignedTF</code>	<code>\IfSocketPlugAssignedTF</code>	<code>{\socket-name}</code> <code>{\plug-name}</code>
<code>\socket_if_plug_assigned:nnTF</code>	<code>*</code>	<code>{\true code}</code> <code>{\false code}</code>

If plug `\plug-name` is assigned to socket `\socket-name` then execute `\true code` otherwise `\false code`. Variants with only T or F are also available.

<code>\DebugSocketsOn</code>	<code>\DebugSocketsOn ... \DebugSocketsOff</code>
<code>\DebugSocketsOff</code>	Turns debugging of sockets on or off.
<code>\socket_debug_on:</code>	
<code>\socket_debug_off:</code>	

2.3.4 Rationale for error handling

The errors during the declarations are produced to help with typos—after all, such declarations might be part of a document preamble (not that likely, but possible). However, `\UseSocket` is not doing much checking, e.g.,

```
\UseSocket{misspelled-socket}{hello}{world}
```

will generate a rather low-level error and then typesets “helloworld” because there is no dedicated runtime check if `misspelled-socket` is a known socket.

The reason is that if the misspelling is in the code, then this is a programming error in the package and for speed reasons L^AT_EX does not repeatedly make runtime checks for coding errors unless they can or are likely to be user introduced.

3 The Implementation

The implementation of the socket mechanism should be (partially) redone and we should probably store the different code chunks in a property list so that we can have a decent `\ShowSocket` command that shows the available alternatives.

TODO: *implement?*

```

1 <*2kernel | latexrelease>
2 \ExplSyntaxOn
3 <@@=socket>
4 <latexrelease> \NewModuleRelease{2023/11/01}{ltsockets}
5 <latexrelease> {The~socket~management~system}
```

3.1 Debugging the socket structures

Code and commands in this section are not final, it needs more experimentation to see what kind of tracing information is going to be useful in practice. For now the tracing is mainly meant to be used for code testing and not so much for application testing.

It is quite likely that the commands and the behavior of the tracing might change in the future once we gained some experience with it.

<code>\g__socket_debug_bool</code>	Holds the current debugging state.
<code>6 \bool_new:N \g__socket_debug_bool</code>	
	<i>(End of definition for \g__socket_debug_bool.)</i>
<code>\socket_debug_on:</code>	Turns debugging on and off by redefining <code>__socket_debug:n</code> and <code>__socket_debug_-</code>
<code>\socket_debug_off:</code>	<code>term:n</code> . By default they do nothing.
<code>__socket_debug:n</code>	<code>7 \cs_new_eq:NN __socket_debug:n \use_none:n</code>
<code>__socket_debug_term:n</code>	<code>8 \cs_new_eq:NN __socket_debug_term:n \use_none:n</code>
<code>__socket_debug_gset:</code>	

```

9 \cs_new_protected:Npn \socket_debug_on:
10 {
11   \bool_gset_true:N \g__socket_debug_bool
12   \__socket_debug_gset:
13 }
14 \cs_new_protected:Npn \socket_debug_off:
15 {
16   \bool_gset_false:N \g__socket_debug_bool
17   \__socket_debug_gset:
18 }
19 \cs_new_protected:Npn \__socket_debug_gset:
20 {
21   \cs_gset_protected:Npx \__socket_debug:n ##1
22   { \bool_if:NT \g__socket_debug_bool {##1} }
23   \cs_gset_protected:Npx \__socket_debug_term:n ##1
24   { \bool_if:NT \g__socket_debug_bool
25     { \iow_term:x { ^^J [Sockets]~ ==>~ ##1} } }
26 }

```

(End of definition for `\socket_debug_on:` and others. These functions are documented on page 343.)

3.2 The L3 layer commands

`\socket_new:nn` Declaring a socket creates a str to hold the name (a pointer) to the code that should be used when the socket is executed, and an integer to hold the number of inputs of that socket. Initially, an “empty” code chunk is created and assigned so the socket does nothing by default other than swallowing its inputs (if any).

```

27 \cs_new_protected:Npn \socket_new:nn #1 #2 {
28   \socket_if_exist:nTF {#1}
29   {
30     \msg_error:nnn { socket } { already-declared } {#1}
31   }
32   {

```

We only support declarations on top-level.

```

33   \int_if_zero:nTF \tex_currentgrouplevel:D
34   {
35     \str_new:c { l__socket_#1_plug_str }
36     \seq_new:c { l__socket_#1_plugs_seq }
37     \int_const:cn { c__socket_#1_args_int } {#2}
38     \socket_new_plug:nnn {#1} { noop } {}
39     \int_compare:nNnTF {#2} = 1
40     {
41       \socket_new_plug:nnn {#1} { identity } {##1}
42       \socket_assign_plug:nm {#1} { identity }
43     }
44     { \socket_assign_plug:nn {#1} { noop } }
45     \__socket_debug_term:n
46     { Socket~ '#1'~ declared~ with~ #2~ input(s) }
47   }
48   {
49     \msg_error:nn { socket } { not-top-level }
50   }
51 }

```

```
52 }
```

(End of definition for `\socket_new:nn`. This function is documented on page 341.)

`\socket_if_exist_p:n` Conditional testing the existence of a socket. The argument is fully expanded as part of
`\socket_if_exist:nTF` the csname generation.

```
53 \prg_new_conditional:Npnn \socket_if_exist:n #1 { p , T , F , TF }
54   { \str_if_exist:cTF { l__socket_#1_plug_str }
55     \prg_return_true:
56     \prg_return_false:
57   }
```

(End of definition for `\socket_if_exist:nTF`. This function is documented on page 342.)

`\socket_log:n` Show the current state of the socket — for now this is just a quick draft and should be
`\socket_show:n` redone and extended.

```
58 \cs_new_protected:Npn \socket_log:n #1 {
59   \typeout{ Socket~ #1:}
60   \socket_if_exist:nTF {#1}
61   {
62     \typeout{ \@spaces number~ of~ inputs~ =~
63               \int_use:c { c__socket_#1_args_int } }
64     \typeout{ \@spaces available~plugs~ =~
65               \seq_use:cnnn { l__socket_#1_plugs_seq }{,~}{,~}{,~} }
66     \typeout{ \@spaces current~ plug~ =~
67               \str_use:c { l__socket_#1_plug_str } }
68     \typeout{ \@spaces definition~ =~
69               \cs_meaning:c
70               { __socket_#1_plug_ \str_use:c { l__socket_#1_plug_str } :w } }
71     \typeout{}
72   }
73   {
```

If we are showing a socket it is not an error if it doesn't exist.

```
74     \typeout { Socket~ is~ not~ declared! }
75   }
76 }
```

And here the version that stops:

```
77 \cs_new_protected:Npn \socket_show:n #1 {\socket_log:n {#1} \errmessage{}}
```

(End of definition for `\socket_log:n` and `\socket_show:n`. These functions are documented on page 342.)

`\socket_new_plug:nnn` Declaring a code for a socket is just making a definition, taking the number of arguments
`\socket_set_plug:nnn` from the saved int.

```
78 \cs_new_protected:Npn \socket_new_plug:nnn #1#2#3 {
79   \socket_if_exist:nTF {#1}
80   {
81     \socket_if_plug_exist:nnTF {#1} {#2}
82     {
83       \msg_error:nnnn { socket } { plug-already-declared } {#1} {#2}
84     }
85     {
86       \cs_generate_from_arg_count:cNnn
```

```

87         { __socket_#1_plug_#2:w }
88         \cs_new:Npn
89         { \int_use:c { c__socket_#1_args_int } }
90         {#3}

```

This is a new declaration so we add the name to a seq for the debugging info.

```

91         \seq_put_right:cn { l__socket_#1_plugs_seq } {#2}
92         \__socket_debug_term:n
93         { Plug~ '#2'~ for~ socket~ '#1'~ declared. }
94     }
95 }
96 { \msg_error:nnn { socket } { undeclared } {#1} }
97 }

```

Changing the plug of an existing socket is rather similar, except that we don't have to deal with adding it to the debugging sequence.

```

98 \cs_new_protected:Npn \socket_set_plug:nnn #1#2#3 {
99     \socket_if_exist:nTF {#1}
100     {
101         \socket_if_plug_exist:nnTF {#1} {#2}
102         {
103             \cs_generate_from_arg_count:cNnn
104             { __socket_#1_plug_#2:w }
105             \cs_set:Npn
106             { \int_use:c { c__socket_#1_args_int } }
107             {#3}
108             \__socket_debug_term:n
109             { Plug~ '#2'~ for~ socket~ '#1'~ changed. }
110         }
111         {
112             \msg_error:nnnn { socket } { plug-undeclared } {#1} {#2}
113         }
114     }
115     { \msg_error:nnn { socket } { undeclared } {#1} }
116 }

```

(End of definition for \socket_new_plug:nnn and \socket_set_plug:nnn. These functions are documented on page 341.)

`\socket_if_plug_exist_p:nn` Conditional testing the existence of a plug. Both arguments are fully expanded as part of the csname generation.

`\socket_if_plug_exist:nnTF`

```

117 \prg_new_conditional:Npnn \socket_if_plug_exist:nn #1#2 { p , T , F , TF }
118 { \cs_if_exist:cTF { __socket_#1_plug_#2:w }
119     \prg_return_true:
120     \prg_return_false:
121 }

```

(End of definition for \socket_if_plug_exist:nnTF. This function is documented on page 342.)

`\socket_assign_plug:nn` Assigning a plug to a socket just changes the name in the socket string. The assignment is local to the current group.

```

122 \cs_new_protected:Npn \socket_assign_plug:nn #1 #2 {
123     \socket_if_exist:nTF {#1}
124     {
125         \socket_if_plug_exist:nnTF {#1} {#2}

```

```

126     {
127         \__socket_debug_term:n
128         { Replacing~ plug~ '\str_use:c { l__socket_#1_plug_str }'~
129           with~ '#2'~ in~ socket~ '#1'. }
130         \str_set:cn { l__socket_#1_plug_str } {#2}
131     }
132     {
133         \msg_error:nnnn { socket } { plug-undeclared } {#1} {#2}
134     }
135 }
136 { \msg_error:nnn { socket } { undeclared } {#1} }
137 }

```

(End of definition for `\socket_assign_plug:nn`. This function is documented on page 341.)

```

\socket_if_plug_assigned_p:nn Conditional testing the assignment of a plug. Both arguments are fully expanded.
\socket_if_plug_assigned:nnTF 138 \prg_new_conditional:Npnn \socket_if_plug_assigned:nn #1#2 { p , T , F , TF }
139   { \exp_args:Ne
140     \str_if_eq:nvTF {#2} { l__socket_#1_plug_str }
141     \prg_return_true:
142     \prg_return_false:
143   }

```

(End of definition for `\socket_if_plug_assigned:nnTF`. This function is documented on page 342.)

`\socket_use:nw` And using it is more or less a `\use:c` so very lightweight. We do not add a runtime check for speed reasons!

`\socket_use:n` This command is named `\socket_use:nw` because we don't know how many inputs the socket has until we have looked at the socket name (in argument #1). But, of course,

`\socket_use:nn` the developer knows so we also offer a few aliases `\socket_use:nn`, etc. so that one can indicate the correct number of arguments (socket inputs plus one) in the L3 layer code.

`\socket_use:nnn`

`\socket_use:nnnn`

```

144 \cs_new_protected:Npn \socket_use:nw #1 {
145   \__socket_debug_term:n
146   { Socket~ '#1'~ containing~ plug~
147     '\str_use:c { l__socket_#1_plug_str }'~ used. }
148   \use:c { __socket_#1_plug_ \str_use:c { l__socket_#1_plug_str } :w }
149 }

```

To make code a bit more readable we also define functions that indicate how many arguments are picked up. However, this is just for code documentation: internally they all do the same and the number of arguments isn't checked by default.

```

150 \cs_new_eq:NN \socket_use:n \socket_use:nw % socket with no inputs
151 \cs_new_eq:NN \socket_use:nn \socket_use:nw % socket with one input
152 \cs_new_eq:NN \socket_use:nnn \socket_use:nw % socket with two inputs
153 \cs_new_eq:NN \socket_use:nnnn \socket_use:nw % socket with three inputs

```

The above commands could be changed to check how many inputs the socket is declared with (for example, when checking is in force).

TODO: *Implement?*

(End of definition for `\socket_use:nw` and others. These functions are documented on page 342.)

```

\socket_use_expandable:nw The same as the non-expandable code, except for the missing debug output.
\socket_use_expandable:n
154 \cs_new:Npn \socket_use_expandable:nw #1 {
155   \use:c { __socket_#1_plug_ \str_use:c { l__socket_#1_plug_str } :w }
156 }
157 \cs_new_eq:NN \socket_use_expandable:n \socket_use_expandable:nw % socket with no inputs

```

(End of definition for `\socket_use_expandable:nw` and `\socket_use_expandable:n`. These functions are documented on page 342.)

3.3 Error messages

```

158 \msg_new:nmmm { socket } { already-declared }
159   { Socket~ '#1'~ already~ declared! }
160   { A~ socket~ can~ only~ be~ declared~ once.~ The~ name~ '#1'~ is~
161     already~ taken.~ Use~ \ShowSocket{#1}~ to~ see~ its~ definition. }
162
163 \msg_new:nmmm { socket } { undeclared }
164   { Socket~ '#1'~ undeclared! }
165   { You~ tried~ to~ use~ a~ socket~ that~ was~ not~ declared~ before. }
166
167 \msg_new:nmmm { socket } { not-top-level }
168   { Sockets~ can~ only~ be~ declared~ at~ top-level! }
169   { It~ is~ not~ allowed~ to~ declare~ sockets~ inside~ a~
170     group.~ Move~ the~ declaration~ to~ the~ top-level. }
171
172 \msg_new:nmmm { socket } { plug-already-declared }
173   { Plug~ '#2'~ for~ socket~ '#1'~ already~ declared! }
174   { You~ can't~ change~ an~ existing~ plug~ with~ \NewSocketPlug~ and~ it~
175     is~ normally~ not~ sensible~ to~ do~ so.~ Use~ the~ L3~ programming~
176     layer~ function~ \socket_set_plug:nmm~ if~ you~ really~ have~ to. }
177
178 \msg_new:nmmm { socket } { plug-undeclared }
179   { Plug~ '#2'~ for~ socket~ '#1'~ undeclared! }
180   { The~ plug~ name~ is~ unknown.~ Is~ the~ name~ misspelled~ or~ did~ you~
181     intend~ to~ assign~ it~ to~ a~ different~ socket? }

```

181 \prop_gput:Nnn \g_msg_module_type_prop { socket } { LaTeX }

3.4 The L^AT_EX 2_ε interface commands

As we expect that there are existing L^AT_EX 2_ε packages that may want to make use of the socket mechanism, we provide 2e names for most of the commands.

<code>\NewSocket</code>	<code>\cs_new_eq:NN \NewSocket</code>	<code>\socket_new:nn</code>
<code>\NewSocketPlug</code>	<code>\cs_new_eq:NN \NewSocketPlug</code>	<code>\socket_new_plug:nmm</code>
<code>\ShowSocket</code>	<code>\cs_new_eq:NN \ShowSocket</code>	<code>\socket_show:n</code>
<code>\LogSocket</code>	<code>\cs_new_eq:NN \LogSocket</code>	<code>\socket_log:n</code>
<code>\AssignSocketPlug</code>	<code>\cs_new_eq:NN \AssignSocketPlug</code>	<code>\socket_assign_plug:nn</code>
<code>\UseSocket</code>	<code>\cs_new_eq:NN \UseSocket</code>	<code>\socket_use:nw</code>
<code>\DebugSocketsOn</code>	<code>\cs_new_eq:NN \DebugSocketsOn</code>	<code>\socket_debug_on:</code>
<code>\DebugSocketsOff</code>	<code>\cs_new_eq:NN \DebugSocketsOff</code>	<code>\socket_debug_off:</code>

(End of definition for `\NewSocket` and others. These functions are documented on page 341.)

```

\IfSocketExistsTF A bunch of conditionals:
\IfSocketExistsT 190 \cs_new_eq:NN \IfSocketExistsTF \socket_if_exist:nTF
\IfSocketExistsF 191 \cs_new_eq:NN \IfSocketExistsT \socket_if_exist:nT
\IfSocketPlugExistsTF 192 \cs_new_eq:NN \IfSocketExistsF \socket_if_exist:nF
\IfSocketPlugExistsT 193 \cs_new_eq:NN \IfSocketPlugExistsTF \socket_if_plug_exist:nnTF
\IfSocketPlugExistsF 194 \cs_new_eq:NN \IfSocketPlugExistsT \socket_if_plug_exist:nnT
\IfSocketPlugAssignedTF 195 \cs_new_eq:NN \IfSocketPlugExistsF \socket_if_plug_exist:nnF
\IfSocketPlugAssignedT 196 \cs_new_eq:NN \IfSocketPlugAssignedTF \socket_if_plug_assigned:nnTF
\IfSocketPlugAssignedF 197 \cs_new_eq:NN \IfSocketPlugAssignedT \socket_if_plug_assigned:nnT
198 \cs_new_eq:NN \IfSocketPlugAssignedF \socket_if_plug_assigned:nnF

```

(End of definition for \IfSocketExistsTF and others. These functions are documented on page 342.)

```

199 %
200 <latexrelease> \IncludeInRelease{0000/00/00}{ltsockets}
201 <latexrelease> {The~socket~management~(undo)}%
202 <latexrelease>
203 <latexrelease> \let \NewSocket \@undefined
204 <latexrelease> \let \ShowSocket \@undefined
205 <latexrelease> \let \LogSocket \@undefined
206 <latexrelease>
207 <latexrelease> \let \NewSocketPlug \@undefined
208 <latexrelease> \let \AssignSocketPlug \@undefined
209 <latexrelease> \let \UseSocket \@undefined
210 <latexrelease>
211 <latexrelease> \let \DebugSocketsOn \@undefined
212 <latexrelease> \let \DebugSocketsOff \@undefined
213 <latexrelease>
214 <latexrelease> \let \IfSocketExistsTF \@undefined
215 <latexrelease> \let \IfSocketExistsT \@undefined
216 <latexrelease> \let \IfSocketExistsF \@undefined
217 <latexrelease> \let \IfSocketPlugExistsTF \@undefined
218 <latexrelease> \let \IfSocketPlugExistsT \@undefined
219 <latexrelease> \let \IfSocketPlugExistsF \@undefined
220 <latexrelease> \let \IfSocketPlugAssignedTF \@undefined
221 <latexrelease> \let \IfSocketPlugAssignedT \@undefined
222 <latexrelease> \let \IfSocketPlugAssignedF \@undefined
223 <latexrelease>
224 <latexrelease> \EndModuleRelease
225 \ExplSyntaxOff
226 </2ekernel | latexrelease>
Reset module prefix:
227 <@@=>

```

File 11

ltemplates.dtx

1 Introduction

There are three broad “layers” between putting down ideas into a source file and ending up with a typeset document. These layers of document writing are

1. authoring of the text with mark-up;
2. document layout design;
3. implementation (with T_EX programming) of the design.

We write the text as an author, and we see the visual output of the design after the document is generated; the T_EX implementation in the middle is the glue between the two.

L^AT_EX’s greatest success has been to standardise a system of mark-up that balances the trade-off between ease of reading and ease of writing to suit almost all forms of technical writing. It’s other original strength was a good background in typographical design; while the standard L^AT_EX 2_ε classes look somewhat dated now in terms of their visual design, their typography is generally sound (barring the occasional minor faults).

However, L^AT_EX 2_ε has always lacked a standard approach to customising the visual design of a document. Changing the looks of the standard classes involved either:

- Creating a new version of the implementation code of the class and editing it.
- Loading one of the many packages to customise certain elements of the standard classes.
- Loading a completely different document class, such as KOMA-Script or memoir, that allows easy customization.

All three of these approaches have their drawbacks and learning curves.

The idea behind ltemplates is to cleanly separate the three layers introduced at the beginning of this section, so that document authors who are not programmers can easily change the design of their documents. ltemplates also makes it easier for L^AT_EX programmers to provide their own customizations on top of a pre-existing class.

2 What is a document?

Besides the textual content of the words themselves, the source file of a document contains mark-up elements that add structure to the document. These elements include sectional divisions, figure/table captions, lists of various sorts, theorems/proofs, and so on. The list will be different for every document that can be written.

Each element can be represented logically without worrying about the formatting, with mark-up such as `\section`, `\caption`, `\begin{enumerate}` and so on. The output of each one of these document elements will be a typeset representation of the information marked up, and the visual arrangement and design of these elements can vary widely in producing a variety of desired outcomes.

For each type of document element, there may be design variations that contain the same sort of information but present it in slightly different ways. For example, the difference between a numbered and an unnumbered section, `\section` and `\section*`, or the difference between an itemized list or an enumerated list.

There are three distinct layers in the definition of “a document” at this level

1. semantic elements such as the ideas of sections and lists;
2. a set of design solutions for representing these elements visually;
3. specific variations for these designs that represent the elements in the document.

In the parlance of the template system, these are called types, templates, and instances, and they are discussed below in sections 4, 5, and 7, respectively.

3 Types, templates, and instances

By formally declaring documents to be composed of mark-up elements grouped into types, which are interpreted and typeset with a set of templates, each of which has one or more instances with which to compose each and every semantic unit of the text, we can cleanly separate the components of document construction.

All of the structures provided by the template system are global, and do not respect \TeX grouping.

4 Template types

An *template type* (sometimes just “type”) is an abstract idea of a document element that takes a fixed number of arguments corresponding to the information from the document author that it is representing. A sectioning type, for example, might take three inputs: “title”, “short title”, and “label”.

Any given document class will define which types are to be used in the document, and any template of a given type can be used to generate an instance for the type. (Of course, different templates will produce different typeset representations, but the underlying content will be the same.)

```
\NewTemplateType <template type> <no. of args>
```

This function defines an `<template type>` taking `<number of arguments>`, where the `<type>` is an abstraction as discussed above. For example,

```
\NewTemplateType{sectioning}{3}
```

creates a type “sectioning”, where each use of that type will need three arguments.

5 Templates

A *template* is a generalized design solution for representing the information of a specified type. Templates that do the same thing, but in different ways, are grouped together by their type and given separate names. There are two important parts to a template:

- the parameters it takes to vary the design it is producing;

Key-type	Description of input
<code>boolean</code>	<code>true</code> or <code>false</code>
<code>choice{⟨choices⟩}</code>	A list of pre-defined <code>⟨choices⟩</code>
<code>commalist</code>	A comma-separated list
<code>function{⟨N⟩}</code>	A function definition with N arguments (N from 0 to 9)
<code>instance{⟨name⟩}</code>	An instance of type <code>⟨name⟩</code>
<code>integer</code>	An integer or integer expression
<code>length</code>	A fixed length
<code>muskip</code>	A math length with shrink and stretch components
<code>real</code>	A real (floating point) value
<code>skip</code>	A length with shrink and stretch components
<code>tokenlist</code>	A token list: any text or commands

Table 1: Key-types for defining template interfaces with `\DeclareTemplateInterface`.

- the implementation of the design.

As a document author or designer does not care about the implementation but rather only the interface to the template, these two aspects of the template definition are split into two independent declarations, `\DeclareTemplateInterface` and `\DeclareTemplateCode`.

```

\DeclareTemplateInterface \DeclareTemplateInterface
  {⟨type⟩} {⟨template⟩} {⟨no. of args⟩}
  {⟨key list⟩}

```

A `⟨template⟩` interface is declared for a particular `⟨type⟩`, where the `⟨number of arguments⟩` must agree with the type declaration. The interface itself is defined by the `⟨key list⟩`, which is itself a key–value list taking a specialized format:

```

  ⟨key1⟩ ":" ⟨key type1⟩ ","
  ⟨key2⟩ ":" ⟨key type2⟩ ","
  ⟨key3⟩ ":" ⟨key type3⟩ "=" ⟨default3⟩ ","
  ⟨key4⟩ ":" ⟨key type4⟩ "=" ⟨default4⟩ ","
  ...

```

Each `⟨key⟩` name should consist of ASCII characters, with the exception of `,`, `=` and `␣`. The recommended form for key names is to use lower case letters, with dashes to separate out different parts. Spaces are ignored in key names, so they can be included or missed out at will. Each `⟨key⟩` must have a `⟨key type⟩`, which defined the type of input that the `⟨key⟩` requires. A full list of key types is given in Table 1. Each key may have a `⟨default⟩` value, which will be used in by the template if the `⟨key⟩` is not set explicitly. The `⟨default⟩` should be of the correct form to be accepted by the `⟨key type⟩` of the `⟨key⟩`: this is not checked by the code. Expressions for numerical values are evaluated when the template is used, thus for example values given in terms of `em` or `ex` will be set respecting the prevailing font.

`\KeyValue` `\KeyValue {<key name>}`

There are occasions where the default (or value) for one key should be taken from another. The `\KeyValue` function can be used to transfer this information without needing to know the internal implementation of the key:

```
\DeclareTemplateInterface { type } { template } { no. of args }
{
  key-name-1 : key-type = value ,
  key-name-2 : key-type = \KeyValue { key-name-1 },
  ...
}
```

`\DeclareTemplateCode` `\DeclareTemplateCode`
`{<type>} {<template>} {<no. of args>}`
`{<key bindings>} {<code>}`

The relationship between a templates keys and the internal implementation is created using the `\DeclareTemplateCode` function. As with `\DeclareTemplateInterface`, the `<template>` name is given along with the `<type>` and `<number of arguments>` required. The `<key bindings>` argument is a key-value list which specifies the relationship between each `<key>` of the template interface with an underlying `<variable>`.

```
<key1> "=" <variable1>,
<key2> "=" <variable2>,
<key3> "=" global <variable3>,
<key4> "=" global <variable4>,
...
```

With the exception of the choice, code and function key types, the `<variable>` here should be the name of an existing L^AT_EX3 register. As illustrated, the key word “global” may be included in the listing to indicate that the `<variable>` should be assigned globally. A full list of variable bindings is given in Table 2.

The `<code>` argument of `\DeclareTemplateCode` is used as the replacement text for the template when it is used, either directly or as an instance. This may therefore accept arguments #1, #2, etc. as detailed by the `<number of arguments>` taken by the type.

`\AssignTemplateKeys` `\AssignTemplateKeys`

In the final argument of `\DeclareTemplateCode` the assignment of keys defined by the template may be delayed by including the command `\AssignTemplateKeys`. If this is *not* present, keys are assigned immediately before the template code. If an `\AssignTemplateKeys` command is present, assignment is delayed until this point. Note that the command must be *directly* present in the code, not placed within a nested command/macro.

Key-type	Description of binding
<code>boolean</code>	Boolean variable, <i>e.g.</i> <code>\l_tmpa_bool</code>
<code>choice</code>	List of choice implementations (see Section 6)
<code>commalist</code>	Comma list, <i>e.g.</i> <code>\l_tmpa_clist</code>
<code>function</code>	Function taking N arguments, <i>e.g.</i> <code>\use_i:nn</code>
<code>instance</code>	
<code>integer</code>	Integer variable, <i>e.g.</i> <code>\l_tmpa_int</code>
<code>length</code>	Dimension variable, <i>e.g.</i> <code>\l_tmpa_dim</code>
<code>muskip</code>	Muskip variable, <i>e.g.</i> <code>\l_tmpa_muskip</code>
<code>real</code>	Floating-point variable, <i>e.g.</i> <code>\l_tmpa_fp</code>
<code>skip</code>	Skip variable, <i>e.g.</i> <code>\l_tmpa_skip</code>
<code>tokenlist</code>	Token list variable, <i>e.g.</i> <code>\l_tmpa_tl</code>

Table 2: Bindings required for different key types when defining template implementations with `\DeclareTemplateCode`. Apart from `code`, `choice` and `function` all of these accept the key word `global` to carry out a global assignment.

<code>\SetKnownTemplateKeys</code>	<code>\SetKnownTemplateKeys {<type>} {<template>} {<keyvals>}</code>
<code>\SetTemplateKeys</code>	<code>\SetTemplateKeys {<type>} {<template>} {<keyvals>}</code>
<code>\UnusedTemplateKeys</code>	<code>\UnusedTemplateKeys % all <keyvals> unused by previous \SetKnownTemplateKeys</code>

In the final argument of `\DeclareTemplateCode` one can also overwrite (some of) the current template key value settings by using the command `\SetKnownTemplateKeys` or `\SetTemplateKeys`, i.e., they can overwrite the template default values and the values assigned by the instance.

The `\SetKnownTemplateKeys` and `\SetTemplateKeys` commands are only supported within the code of a template; using them elsewhere has unpredictable results. If they are used together with `\AssignTemplateKeys` then the latter command should come first in the template code.

The main use case for these commands is the situation where there is an argument (normally #1) to the template in which a key/value list can be specified that overwrites the normal settings. In that case one could use

```
\SetKnownTemplateKeys{<type>}{<template>}{#1}
```

to process this key/value list inside the template.

If `\SetKnownTemplateKeys` is executed and the `<keyvals>` argument contains keys not known to the `<template>` they are simply ignored and stored in the tokenlist `\UnusedTemplateKeys` without generating an error. This way it is possible to apply the same key/val list specified by the user on a document-level command or environment to several templates, which is useful, if the command or environment is implemented by calling several different template instances.

As a variation of that, you can use this key/val list the first time, and for the next template instance use what remains in `\UnusedTemplateKeys` (i.e., the key/val list with only the keys that have not been processed previously). The final processing step could then be `\SetTemplateKeys`, which unconditionally attempts to set the `<keyvals>` received in its third argument. This command complains if any of them are unknown keys. Alternatively, you could use `\SetKnownTemplateKeys` and afterwards

check whether `\UnusedTemplateKeys` is empty.²³

For example, a list, such as `enumerate`, is made up from a `blockenv`, `block`, `list`, and a `para` template and in the single user-supplied optional argument of `enumerate` key/values for any of these templates might be specified.

In fact, in the particular example of list environments, the supplied key/value list is also saved and then applied to each `\item` which is implemented through an `item` template. This way, one can specify one-off settings for all the items of a single list (on the environment level), as well as to individual items within that list (by specifying them in the optional argument of an `\item`). With `\SetKnownTemplateKeys` and `\SetTemplateKeys` working together, it is possible to provide this flexibility and still alert the user when one of their keys is misspelled.

On the other hand you may want to allow for “misspellings” without generating an error or a warning. For example, if you define a template that accepts only a few keys, you might just want to ignore anything specified in the source when you use this template in place of a different one, without the need to alter the document source. Or you might just generate a warning message, which is easy, given that the unused key/values are available in the `\UnusedTemplateKeys` variable.

```
\DeclareTemplateCopy \DeclareTemplateCopy
  {<type>} {<template2>} {<template1>}
```

Copies `<template1>` of `<type>` to a new name `<template2>`: the copy can then be edited independent of the original.

6 Multiple choices

The `choice` key type implements multiple choice input. At the interface level, only the list of valid choices is needed:

```
\DeclareTemplateInterface { foo } { bar } { 0 }
  { key-name : choice { A, B, C } }
```

where the choices are given as a comma-list (which must therefore be wrapped in braces). A default value can also be given:

```
\DeclareTemplateInterface { foo } { bar } { 0 }
  { key-name : choice { A, B, C } = A }
```

At the implementation level, each choice is associated with code, using a nested key-value list.

```
\DeclareTemplateCode { foo } { bar } { 0 }
  {
    key-name =
    {
      A = Code-A ,
      B = Code-B ,
      C = Code-C
```

²³Using `\SetTemplateKeys` exposes the inner structure of the template keys when generating an error. This is something one may want to avoid as it can be confusing to the user, especially if several templates are involved. In that case use `\SetKnownTemplateKeys` and afterwards check whether `\UnusedTemplateKeys` is empty; if it is not empty then generate your own error message.

```

    }
  }
  { ... }

```

The two choice lists should match, but in the implementation a special **unknown** choice is also available. This can be used to ignore values and implement an “else” branch:

```

\DeclareTemplateCode { foo } { bar } { 0 }
{
  key-name =
  {
    A      = Code-A ,
    B      = Code-B ,
    C      = Code-C ,
    unknown = Else-code
  }
}
{ ... }

```

The **unknown** entry must be the last one given, and should *not* be listed in the interface part of the template.

For keys which accept the values **true** and **false** both the boolean and choice key types can be used. As template interfaces are intended to prompt clarity at the design level, the boolean key type should be favored, with the choice type reserved for keys which take arbitrary values.

7 Instances

After a template is defined it still needs to be put to use. The parameters that it expects need to be defined before it can be used in a document. Every time a template has parameters given to it, an *instance* is created, and this is the code that ends up in the document to perform the typesetting of whatever pieces of information are input into it.

For example, a template might say “here is a section with or without a number that might be centered or left aligned and print its contents in a certain font of a certain size, with a bit of a gap before and after it” whereas an instance declares “this is a section with a number, which is centered and set in 12pt italic with a 10pt skip before and a 12pt skip after it”. Therefore, an instance is just a frozen version of a template with specific settings as chosen by the designer.

`\DeclareInstance` `\DeclareInstance`
`{⟨type⟩} {⟨instance⟩} {⟨template⟩} {⟨parameters⟩}`

This function uses a `⟨template⟩` for an `⟨type⟩` to create an `⟨instance⟩`. The `⟨instance⟩` will be set up using the `⟨parameters⟩`, which will set some of the `⟨keys⟩` in the `⟨template⟩`.

As a practical example, consider a type for document sections (which might include chapters, parts, sections, *etc.*), which is called `sectioning`. One possible template for this type might be called `basic`, and one instance of this template would be a numbered section. The instance declaration might read:

```
\DeclareInstance { sectioning } { section-num } { basic }
{
  numbered      = true ,
  justification = center ,
  font          =\normalsize\itshape ,
  before-skip  = 10pt ,
  after-skip   = 12pt ,
}
```

Of course, the key names here are entirely imaginary, but illustrate the general idea of fixing some settings.

`\IfInstanceExistsT` `\IfInstanceExistsTF` `{⟨type⟩} {⟨instance⟩} {⟨true code⟩} {⟨false code⟩}`

`\IfInstanceExistsF`
`\IfInstanceExistsTF` Tests if the named `⟨instance⟩` of a `⟨type⟩` exists, and then inserts the appropriate code into the input stream.

`\DeclareInstanceCopy` `\DeclareInstanceCopy`
`{⟨type⟩} {⟨instance2⟩} {⟨instance1⟩}`

Copies the `⟨values⟩` for `⟨instance1⟩` for an `⟨type⟩` to `⟨instance2⟩`.

8 Document interface

After the instances have been chosen, document commands must be declared to use those instances in the document. `\UseInstance` calls instances directly, and this command should be used internally in document-level mark-up.

`\UseInstance` `\UseInstance`
`{⟨type⟩} {⟨instance⟩} ⟨arguments⟩`

Uses an `⟨instance⟩` of the `⟨type⟩`, which will require `⟨arguments⟩` as determined by the number specified for the `⟨type⟩`. The `⟨instance⟩` must have been declared before it can be used, otherwise an error is raised.

`\UseTemplate` `\UseTemplate {<type>} {<template>}`
`{<settings>} <arguments>`

Uses the `<template>` of the specified `<type>`, applying the `<settings>` and absorbing `<arguments>` as detailed by the `<type>` declaration. This in effect is the same as creating an instance using `\DeclareInstance` and immediately using it with `\UseInstance`, but without the instance having any further existence. This command is therefore useful when a template needs to be used only once.

This function can also be used as the argument to instance key types:

```
\DeclareInstance { type } { template } { instance }
{
  instance-key =
    \UseTemplate { type2 } { template2 } { <settings> }
}
```

9 Changing existing definitions

Template parameters may be assigned specific defaults for instances to use if the instance declaration doesn't explicit set those parameters. In some cases, the document designer will wish to edit these defaults to allow them to “cascade” to the instances. The alternative would be to set each parameter identically for each instance declaration, a tedious and error-prone process.

`\EditTemplateDefaults` `\EditTemplateDefaults`
`{<type>} {<template>} {<new defaults>}`

Edits the `<defaults>` for a `<template>` for an `<type>`. The `<new defaults>`, given as a key–value list, replace the existing defaults for the `<template>`. This means that the change will apply to instances declared after the editing, but that instances which have already been created are unaffected.

`\EditInstance` `\EditInstance`
`{<type>} {<instance>} {<new values>}`

Edits the `<values>` for an `<instance>` for an `<type>`. The `<new values>`, given as a key–value list, replace the existing values for the `<instance>`. This function is complementary to `\EditTemplateDefaults`: `\EditInstance` changes a single instance while leaving the template untouched.

10 Getting information about templates and instances

`\ShowInstanceValues` `\ShowInstanceValues {<type>} {<instance>}`

Shows the `<values>` for an `<instance>` of the given `<type>` at the terminal.

`\ShowTemplateCode` `\ShowTemplateCode {<type>} {<template>}`

Shows the `<code>` of a `<template>` for an `<type>` in the terminal.

<code>\ShowTemplateDefaults</code>	<code>\ShowTemplateDefaults {<type>} {<template>}</code> Shows the <i><default></i> values of a <i><template></i> for an <i><type></i> in the terminal.
<code>\ShowTemplateInterface</code>	<code>\ShowTemplateInterface {<type>} {<template>}</code> Shows the <i><keys></i> and associated <i><key types></i> of a <i><template></i> for an <i><type></i> in the terminal.
<code>\ShowTemplateVariables</code>	<code>\ShowTemplateVariables {<type>} {<template>}</code> Shows the <i><variables></i> and associated <i><keys></i> of a <i><template></i> for an <i><type></i> in the terminal. Note that <code>code</code> and <code>choice</code> keys do not map directly to variables but to arbitrary code. For <code>choice</code> keys, each valid choice is shown as a separate entry in the list, with the key name and choice separated by a space, for example

```

Template 'example' of type 'example' has variable mapping:
> demo unknown => \def \demo {?}
> demo c => \def \demo {c}
> demo b => \def \demo {b}
> demo a => \def \demo {a}.

```

would be shown for a choice key `demo` with valid choices `a`, `b` and `c`, plus code for an `unknown` branch.

11 The implementation

```

1 <@@=template>
2 <*2kernel>
3 \message{templates,}
4 </2kernel>
5 <*2kernel | latexrelease>
6 \ExplSyntaxOn
7 <latexrelease> \NewModuleRelease{2024/06/01}{lttemplates}
8 <latexrelease> \PrototypeDocumentCommands%

```

11.1 Variables and constants

```
\c__template_code_root_tl  
\c__template_defaults_root_tl  
\c__template_instances_root_tl  
\c__template_keytypes_root_tl  
\c__template_key_order_root_tl  
\c__template_restrict_root_tl  
\c__template_values_root_tl  
\c__template_vars_root_tl
```

So that literal values are kept to a minimum.

```
9 \tl_const:Nn \c__template_code_root_tl { template~code~>~ }  
10 \tl_const:Nn \c__template_defaults_root_tl { template~defaults~>~ }  
11 \tl_const:Nn \c__template_instances_root_tl { template~instance~>~ }  
12 \tl_const:Nn \c__template_keytypes_root_tl { template~key~types~>~ }  
13 \tl_const:Nn \c__template_key_order_root_tl { template~key~order~>~ }  
14 \tl_const:Nn \c__template_values_root_tl { template~values~>~ }  
15 \tl_const:Nn \c__template_vars_root_tl { template~vars~>~ }
```

```
\c__template_keytypes_arg_seq
```

A list of keytypes which also need additional data (an argument), used to parse the keytypes correctly.

```
16 \seq_const_from_clist:Nn \c__template_keytypes_arg_seq  
17 { choice , function , instance }
```

```
\g__template_type_prop
```

 For storing types and the associated number of arguments.

```
18 \prop_new:N \g__template_type_prop
```

```
\l__template_assignments_tl
```

When creating an instance, the assigned values are collected here.

```
19 \tl_new:N \l__template_assignments_tl
```

```
\l__template_default_tl
```

 The default value for a key is recovered here from the property list in which it is stored.

```
20 \tl_new:N \l__template_default_tl
```

```
\l__template_error_bool
```

 A flag for errors to be carried forward.

```
21 \bool_new:N \l__template_error_bool
```

```
\l__template_global_bool
```

 Used to indicate that assignments should be global.

```
22 \bool_new:N \l__template_global_bool
```

```

\l__template_key_name_tl
\l__template_keytype_tl
\l__template_keytype_arg_tl
\l__template_value_tl
\l__template_var_tl

```

When defining each key in a template, the name and type of the key need to be separated and stored. Any argument needed by the keytype is also stored separately.

```

23 \tl_new:N \l__template_key_name_tl
24 \tl_new:N \l__template_keytype_tl
25 \tl_new:N \l__template_keytype_arg_tl
26 \tl_new:N \l__template_value_tl
27 \tl_new:N \l__template_var_tl

```

```

\l__template_keytypes_prop
\l__template_key_order_seq
\l__template_values_prop
\l__template_vars_prop

```

To avoid needing too many difficult-to-follow csname assignments, various scratch token registers are used to build up data, which is then transferred

```

28 \prop_new:N \l__template_keytypes_prop
29 \seq_new:N \l__template_key_order_seq
30 \prop_new:N \l__template_values_prop
31 \prop_new:N \l__template_vars_prop

```

```

\l__template_tmp_clist
\l__template_tmp_dim
\l__template_tmp_int
\l__template_tmp_muskip
\l__template_tmp_skip
\l__template_tmp_tl

```

Scratch space.

```

32 \clist_new:N \l__template_tmp_clist
33 \dim_new:N \l__template_tmp_dim
34 \int_new:N \l__template_tmp_int
35 \muskip_new:N \l__template_tmp_muskip
36 \skip_new:N \l__template_tmp_skip
37 \tl_new:N \l__template_tmp_tl

```

```

\s__template_mark
\s__template_stop

```

Internal scan marks.

```

38 \scan_new:N \s__template_mark
39 \scan_new:N \s__template_stop

```

```

\q__template_nil

```

Internal quarks.

```

40 \quark_new:N \q__template_nil

```

```

\__template_quark_if_nil_p:n
\__template_quark_if_nil:nTF

```

Branching quark conditional.

```

41 \__kernel_quark_new_conditional:Nn \__template_quark_if_nil:N { F }

```

(End of definition for __template_quark_if_nil:nTF.)

11.2 Testing existence and validity

There are a number of checks needed for either the existence of a type, template or instance. There are also some for the validity of a particular call. All of these are collected up here.

`_template_execute_if_arg_agree:nnT`

A test agreement between the number of arguments for the template type and that specified when creating a template. This is not done as a separate conditional for efficiency and better error message

```
42 \cs_new_protected:Npn \_template_execute_if_arg_agree:nnT #1#2#3
43 {
44   \prop_get:NnN \g__template_type_prop {#1} \l__template_tmp_tl
45   \int_compare:nNnTF {#2} = \l__template_tmp_tl
46     {#3}
47   {
48     \msg_error:nneee { template } { argument-number-mismatch }
49     {#1} { \l__template_tmp_tl } {#2}
50   }
51 }
```

(End of definition for _template_execute_if_arg_agree:nnT.)

`_template_execute_if_code_exist:nnT`

A template is only fully declared if the code has been set up, which can be checked by looking for the template function itself.

```
52 \cs_new_protected:Npn \_template_execute_if_code_exist:nnT #1#2#3
53 {
54   \cs_if_exist:cTF { \c__template_code_root_tl #1 / #2 }
55     {#3}
56     { \msg_error:nnnn { template } { no-template-code } {#1} {#2} }
57 }
```

(End of definition for _template_execute_if_code_exist:nnT.)

`_template_execute_if_keytype_exist:nT`

`_template_execute_if_keytype_exist:VT`

The test for valid keytypes looks for a function to set up the key, which is part of the “code” side of the template definition. This avoids having different lists for the two parts of the process.

```
58 \cs_new_protected:Npn \_template_execute_if_keytype_exist:nT #1#2
59 {
60   \cs_if_exist:cTF { __template_store_value_ #1 :n }
61     {#2}
62     { \msg_error:nnn { template } { unknown-keytype } {#1} }
63 }
64 \cs_generate_variant:Nn \_template_execute_if_keytype_exist:nT { V }
```

(End of definition for _template_execute_if_keytype_exist:nT.)

`_template_execute_if_type_exist:nT`

To check that a particular type is valid.

```
65 \cs_new_protected:Npn \_template_execute_if_type_exist:nT #1#2
66 {
67   \prop_if_in:NnTF \g__template_type_prop {#1}
68     {#2}
69     { \msg_error:nnn { template } { unknown-type } {#1} }
70 }
```

(End of definition for _template_execute_if_type_exist:nT.)

`_template_execute_if_keys_exist:nnT` To check that the keys for a template have been set up before trying to create any code, a simple check for the correctly-named keytype property list.

```

71 \cs_new_protected:Npn \_template_if_keys_exist:nnT #1#2#3
72   {
73     \cs_if_exist:cTF { \c__template_keytypes_root_tl #1 / #2 }
74       {#3}
75     { \msg_error:nnnn { template } { unknown-template } {#1} {#2} }
76   }

```

(End of definition for _template_execute_if_keys_exist:nnT.)

`_template_if_key_value:nTF` Tests for the first token in a string being `\KeyValue`.

`_template_if_key_value:VTF`

```

77 \prg_new_conditional:Npnn \_template_if_key_value:n #1 { T , F , TF }
78   {
79     \str_if_eq:noTF { \KeyValue } { \tl_head:w #1 \q_nil \q_stop }
80     \prg_return_true:
81     \prg_return_false:
82   }
83 \prg_generate_conditional_variant:Nnn \_template_if_key_value:n { V } { T , F , TF }

```

(End of definition for _template_if_key_value:nTF.)

`_template_if_instance_exist:nnTF` Testing for an instance

```

84 \prg_new_conditional:Npnn \_template_if_instance_exist:nn #1#2 { T, F, TF }
85   {
86     \cs_if_exist:cTF { \c__template_instances_root_tl #1 / #2 }
87     \prg_return_true:
88     \prg_return_false:
89   }

```

(End of definition for _template_if_instance_exist:nnTF.)

`_template_if_use_template:nTF` Tests for the first token in a string being `\UseTemplate`.

```

90 \prg_new_conditional:Npnn \_template_if_use_template:n #1 { TF }
91   {
92     \str_if_eq:noTF { \UseTemplate } { \tl_head:w #1 \q_nil \q_stop }
93     \prg_return_true:
94     \prg_return_false:
95   }

```

(End of definition for _template_if_use_template:nTF.)

11.3 Saving and recovering property lists

The various property lists for templates have to be shuffled in and out of storage.

`_template_store_defaults:nn`

`_template_store_keytypes:nn`

The defaults and keytypes are transferred from the scratch property lists to the “proper” lists for the template being created.

`_template_store_values:nn`

`_template_store_vars:nn`

```

96 \cs_new_protected:Npn \_template_store_defaults:nn #1#2
97   {
98     \debug_suspend:
99     \prop_gclear_new:c { \c__template_defaults_root_tl #1 / #2 }
100    \prop_gset_eq:cN { \c__template_defaults_root_tl #1 / #2 }
101    \l__template_values_prop

```

```

102     \debug_resume:
103   }
104 \cs_new_protected:Npn \__template_store_keytypes:nn #1#2
105   {
106     \debug_suspend:
107     \prop_if_exist:cTF { \c__template_keytypes_root_tl #1 / #2 }
108       {
109         \msg_info:nnnn { template } { declare-template-interface } {#1} {#2}
110         \prop_gclear:c { \c__template_keytypes_root_tl #1 / #2 }
111       }
112     { \prop_new:c { \c__template_keytypes_root_tl #1 / #2 } }
113     \prop_gset_eq:cN { \c__template_keytypes_root_tl #1 / #2 }
114       \l__template_keytypes_prop
115     \seq_gclear_new:c { \c__template_key_order_root_tl #1 / #2 }
116     \seq_gset_eq:cN { \c__template_key_order_root_tl #1 / #2 }
117       \l__template_key_order_seq
118     \debug_resume:
119   }
120 \cs_new_protected:Npn \__template_store_values:nn #1#2
121   {
122     \debug_suspend:
123     \prop_clear_new:c { \c__template_values_root_tl #1 / #2 }
124     \prop_set_eq:cN { \c__template_values_root_tl #1 / #2 }
125       \l__template_values_prop
126     \debug_resume:
127   }
128 \cs_new_protected:Npn \__template_store_vars:nn #1#2
129   {
130     \debug_suspend:
131     \prop_gclear_new:c { \c__template_vars_root_tl #1 / #2 }
132     \prop_gset_eq:cN { \c__template_vars_root_tl #1 / #2 }
133       \l__template_vars_prop
134     \debug_resume:
135   }

```

(End of definition for __template_store_defaults:nn and others.)

```

\__template_recover_defaults:nn
\__template_recover_keytypes:nn
\__template_recover_values:nn
\__template_recover_vars:nn

```

Recovering the stored data for a template is rather less complex than storing it. All that happens is the data is transferred from the permanent to the scratch storage. However, we need to check the scratch storage does exist.

```

136 \cs_new_protected:Npn \__template_recover_defaults:nn #1#2
137   {
138     \prop_if_exist:cTF
139       { \c__template_defaults_root_tl #1 / #2 }
140       {
141         \prop_set_eq:Nc \l__template_values_prop
142           { \c__template_defaults_root_tl #1 / #2 }
143       }
144     { \prop_clear:N \l__template_values_prop }
145   }
146 \cs_new_protected:Npn \__template_recover_keytypes:nn #1#2
147   {
148     \prop_if_exist:cTF
149       { \c__template_keytypes_root_tl #1 / #2 }

```

```

150     {
151       \prop_set_eq:Nc \l__template_keytypes_prop
152       { \c__template_keytypes_root_tl #1 / #2 }
153     }
154     { \prop_clear:N \l__template_keytypes_prop }
155   \seq_if_exist:cTF { \c__template_key_order_root_tl #1 / #2 }
156     {
157       \seq_set_eq:Nc \l__template_key_order_seq
158       { \c__template_key_order_root_tl #1 / #2 }
159     }
160     { \seq_clear:N \l__template_key_order_seq }
161   }
162   \cs_new_protected:Npn \__template_recover_values:nn #1#2
163     {
164       \prop_if_exist:cTF
165       { \c__template_values_root_tl #1 / #2 }
166       {
167         \prop_set_eq:Nc \l__template_values_prop
168         { \c__template_values_root_tl #1 / #2 }
169       }
170       { \prop_clear:N \l__template_values_prop }
171     }
172   \cs_new_protected:Npn \__template_recover_vars:nn #1#2
173     {
174       \prop_if_exist:cTF
175       { \c__template_vars_root_tl #1 / #2 }
176       {
177         \prop_set_eq:Nc \l__template_vars_prop
178         { \c__template_vars_root_tl #1 / #2 }
179       }
180       { \prop_clear:N \l__template_vars_prop }
181     }

```

(End of definition for `__template_recover_defaults:nn` and others.)

11.4 Creating new template types

`__template_define_type:nn` Although the type is the “top level” of the template system, it is actually very easy to implement. All that happens is that the number of arguments required is recorded, indexed by the name of the type.

```

182   \cs_new_protected:Npn \__template_define_type:nn #1#2
183     {
184       \prop_if_in:NnTF \g__template_type_prop {#1}
185       { \msg_error:nnn { template } { type-already-defined } {#1} }
186       { \__template_declare_type:nn {#1} {#2} }
187     }
188   \cs_new_protected:Npn \__template_declare_type:nn #1#2
189     {
190       \int_set:Nn \l__template_tmp_int {#2}
191       \int_compare:nTF { 0 <= \l__template_tmp_int <= 9 }
192       {
193         \msg_info:nnnV { template } { declare-type }
194         {#1} \l__template_tmp_int
195         \prop_gput:NnV \g__template_type_prop {#1}

```

```

196     \l__template_tmp_int
197   }
198   {
199     \msg_error:nnnV { template } { bad-number-of-arguments }
200     {#1} \l__template_tmp_int
201   }
202 }

```

(End of definition for `__template_define_type:nn` and `__template_declare_type:nn`.)

11.5 Design part of template declaration

The “design” part of a template declaration defines the general behaviour of each key, and possibly a default value. However, it does not include the implementation. This means that what happens here is the two properties are saved to appropriate lists, which can then be used later to recover the information when implementing the keys.

`__template_declare_template_keys:nnnn`

The main function for the “design” part of creating a template starts by checking that the type exists and that the number of arguments required agree. If that is all fine, then the two storage areas for defaults and keytypes are initialised. The mechanism is then set up for the `l3keys` module to actually parse the keys. Finally, the code hands off to the storage routine to save the parsed information properly.

```

203 \cs_new_protected:Npn \__template_declare_template_keys:nnnn #1#2#3#4
204   {
205     \__template_execute_if_type_exist:nT {#1}
206     {
207       \__template_execute_if_arg_agree:nnT {#1} {#3}
208       {
209         \prop_clear:N \l__template_values_prop
210         \prop_clear:N \l__template_keytypes_prop
211         \seq_clear:N \l__template_key_order_seq
212         \keyval_parse:NNn
213         \__template_parse_keys_elt:n \__template_parse_keys_elt:nn {#4}
214         \__template_store_defaults:nn {#1} {#2}
215         \__template_store_keytypes:nn {#1} {#2}
216       }
217     }
218 }

```

(End of definition for `__template_declare_template_keys:nnnn`.)

`__template_parse_keys_elt:n`

`__template_parse_keys_elt_aux:n`

`__template_parse_keys_elt_aux:`

Processing the key part of the key–value pair is always carried out using this function, even if a value was found. First, the key name is separated from the keytype, and if necessary the keytype is separated into two parts. This information is then used to check that the keytype is valid, before storing the keytype (plus argument if necessary) as a property of the key name. The key name is also stored (in braces) in the token list to record the order the keys are defined in.

```

219 \cs_new_protected:Npn \__template_parse_keys_elt:n #1
220   {
221     \__template_split_keytype:n {#1}
222     \bool_if:NF \l__template_error_bool
223     {
224       \__template_execute_if_keytype_exist:VT \l__template_keytype_tl

```



```

225     {
226       \seq_map_function:NN \c__template_keytypes_arg_seq
227       \__template_parse_keys_elt_aux:n
228       \bool_if:NF \l__template_error_bool
229       {
230         \seq_if_in:NoTF \l__template_key_order_seq
231         \l__template_key_name_tl
232         {
233           \msg_error:nnV { template } { duplicate-key-interface }
234           \l__template_key_name_tl
235         }
236         { \__template_parse_keys_elt_aux: }
237       }
238     }
239   }
240 }
241 \cs_new_protected:Npn \__template_parse_keys_elt_aux:n #1
242 {
243   \str_if_eq:VnT \l__template_keytype_tl {#1}
244   {
245     \tl_if_empty:NT \l__template_keytype_arg_tl
246     {
247       \msg_error:nnn { template } { keytype-requires-argument } {#1}
248       \bool_set_true:N \l__template_error_bool
249       \seq_map_break:
250     }
251   }
252 }
253 \cs_new_protected:Npn \__template_parse_keys_elt_aux:
254 {
255   \tl_set:Ne \l__template_tmp_tl
256   {
257     \l__template_keytype_tl
258     \tl_if_empty:NF \l__template_keytype_arg_tl
259     { { \l__template_keytype_arg_tl } }
260   }
261   \prop_put:NVV \l__template_keytypes_prop \l__template_key_name_tl
262   \l__template_tmp_tl
263   \seq_put_right:NV \l__template_key_order_seq \l__template_key_name_tl
264   \str_if_eq:VnT \l__template_keytype_tl { choice }
265   {
266     \clist_if_in:NnT \l__template_keytype_arg_tl { unknown }
267     { \msg_error:nn { template } { choice-unknown-reserved } }
268   }
269 }

```

(End of definition for __template_parse_keys_elt:n, __template_parse_keys_elt_aux:n, and __template_parse_keys_elt_aux:.)

__template_parse_keys_elt:nn For keys which have a default, the keytype and key name are first separated out by the __template_parse_keys_elt:n routine, before storing the default value in the scratch property list.

```

270 \cs_new_protected:Npn \__template_parse_keys_elt:nn #1#2
271 {

```

```

272   \__template_parse_keys_elt:n {#1}
273   \use:c { __template_store_value_ \l__template_keytype_tl :n } {#2}
274 }

```

(End of definition for __template_parse_keys_elt:nn.)

__template_split_keytype:n The keytype and key name should be separated by :. As the definition might be given inside or outside of a code block, the category code of colons is standardised. After that, the standard delimited argument method is used to separate the two parts.

```

275 \cs_new_protected:Npe \__template_split_keytype:n #1
276 {
277   \exp_not:N \bool_set_false:N \exp_not:N \l__template_error_bool
278   \tl_set:Nn \exp_not:N \l__template_tmp_tl {#1}
279   \tl_replace_all:Nnn \exp_not:N \l__template_tmp_tl { : } { \token_to_str:N : }
280   \tl_if_in:VnTF \exp_not:N \l__template_tmp_tl { \token_to_str:N : }
281   {
282     \exp_not:n
283     {
284       \tl_clear:N \l__template_key_name_tl
285       \exp_after:wN \__template_split_keytype_aux:w
286       \l__template_tmp_tl \s__template_stop
287     }
288   }
289   {
290     \exp_not:N \bool_set_true:N \exp_not:N \l__template_error_bool
291     \msg_error:nnn { template } { missing-keytype } {#1}
292   }
293 }
294 \use:e
295 {
296   \cs_new_protected:Npn \exp_not:N \__template_split_keytype_aux:w
297   #1 \token_to_str:N : #2 \s__template_stop
298   {
299     \tl_put_right:Ne \exp_not:N \l__template_key_name_tl
300     {
301       \exp_not:N \tl_trim_spaces:e
302       { \exp_not:N \tl_to_str:n {#1} }
303     }
304     \tl_if_in:nnTF {#2} { \token_to_str:N : }
305     {
306       \tl_put_right:Nn \exp_not:N \l__template_key_name_tl
307       { \token_to_str:N : }
308       \exp_not:N \__template_split_keytype_aux:w #2 \s__template_stop
309     }
310     {
311       \exp_not:N \tl_if_empty:NTF \exp_not:N \l__template_key_name_tl
312       {
313         \msg_error:nnn { template } { empty-key-name }
314         { \token_to_str:N : #2 }
315       }
316       { \exp_not:N \__template_split_keytype_arg:n {#2} }
317     }
318   }
319 }

```

(End of definition for `__template_split_keytype:n` and `__template_split_keytype_aux:w`.)

`__template_split_keytype_arg:n`
`__template_split_keytype_arg:V`
`__template_split_keytype_arg_aux:n`
`__template_split_keytype_arg_aux:w`

The second stage of sorting out the keytype is to check for an argument. As there is no convenient delimiting token to look for, a check is made instead for each possible text value for the keytype. To keep things faster, this only involves the keytypes that need an argument. If a match is made, then a check is also needed to see that it is at the start of the keytype information. All being well, the split can then be applied. Any non-matching keytypes are assumed to be “correct” as given, and are left alone (this is checked by other code).

```

320 \cs_new_protected:Npn \__template_split_keytype_arg:n #1
321   {
322   \tl_set:Ne \l__template_keytype_tl { \tl_trim_spaces:n {#1} }
323   \tl_clear:N \l__template_keytype_arg_tl
324   \cs_set_protected:Npn \__template_split_keytype_arg_aux:n ##1
325     {
326     \tl_if_in:nnT {#1} {##1}
327       {
328       \cs_set:Npn \__template_split_keytype_arg_aux:w
329         #####1 ##1 #####2 \s__template_stop
330         {
331         \tl_if_blank:nT {#####1}
332           {
333           \tl_set:Ne \l__template_keytype_tl
334             { \tl_trim_spaces:n {##1} }
335           \tl_if_blank:nF {#####2}
336             {
337             \tl_set:Ne \l__template_keytype_arg_tl
338               { \use:n {#####2} }
339             }
340           \seq_map_break:
341           }
342         }
343         \__template_split_keytype_arg_aux:w #1 \s__template_stop
344       }
345     }
346   \seq_map_function:NN \c__template_keytypes_arg_seq
347     \__template_split_keytype_arg_aux:n
348   }
349 \cs_generate_variant:Nn \__template_split_keytype_arg:n { V }
350 \cs_new:Npn \__template_split_keytype_arg_aux:n #1 { }
351 \cs_new:Npn \__template_split_keytype_arg_aux:w #1 \s__template_stop { }

```

(End of definition for `__template_split_keytype_arg:n`, `__template_split_keytype_arg_aux:n`, and `__template_split_keytype_arg_aux:w`.)

11.5.1 Storing values

As `ltemplates` pre-processes key values for efficiency reasons, there is a need to convert the values given as defaults into “ready to use” data. The same general idea is true when an instance is declared. However, assignments are not made until an instance is used, and so there has to be some intermediate storage. Furthermore, the ability to delay evaluation of results is needed. To achieve these aims, a series of “process and store” functions are defined here.

All of the information about the key (the key name and the keytype) is already stored as variables. The same property list is always used to store the data, meaning that the only argument required is the value to be processed and potentially stored.

`_template_store_value_boolean:n`

```
352 \cs_new_protected:Npn \_template_store_value_boolean:n #1
353   { \prop_put:Non \l__template_values_prop \l__template_key_name_tl {#1} }
```

(End of definition for _template_store_value_boolean:n.)

`_template_store_value:n`

With no need to worry about delayed evaluation, these keytypes all just store the input directly.

`_template_store_value_choice:n`

`_template_store_value_function:n`

`_template_store_value_instance:n`

```
354 \cs_new_protected:Npn \_template_store_value:n #1
355   { \prop_put:Non \l__template_values_prop \l__template_key_name_tl {#1} }
356 \cs_new_eq:NN \_template_store_value_choice:n \_template_store_value:n
357 \cs_new_eq:NN \_template_store_value_function:n \_template_store_value:n
358 \cs_new_eq:NN \_template_store_value_instance:n \_template_store_value:n
```

(End of definition for _template_store_value:n and others.)

`_template_store_value_aux:Nn`

Storing values in `\l__template_values_prop` is in most cases the same.

`_template_store_value_integer:n`

`_template_store_value_length:n`

`_template_store_value_muskip:n`

`_template_store_value_real:n`

`_template_store_value_skip:n`

`_template_store_value_tokenlist:n`

`_template_store_value_commalist:n`

```
359 \cs_new_protected:Npn \_template_store_value_aux:Nn #1#2
360   { \prop_put:Non \l__template_values_prop \l__template_key_name_tl {#2} }
361 \cs_new_protected:Npn \_template_store_value_integer:n
362   { \_template_store_value_aux:Nn \int_eval:n }
363 \cs_new_protected:Npn \_template_store_value_length:n
364   { \_template_store_value_aux:Nn \dim_eval:n }
365 \cs_new_protected:Npn \_template_store_value_muskip:n
366   { \_template_store_value_aux:Nn \muskip_eval:n }
367 \cs_new_protected:Npn \_template_store_value_real:n
368   { \_template_store_value_aux:Nn \fp_eval:n }
369 \cs_new_protected:Npn \_template_store_value_skip:n
370   { \_template_store_value_aux:Nn \skip_eval:n }
371 \cs_new_protected:Npn \_template_store_value_tokenlist:n
372   { \_template_store_value_aux:Nn \use:n }
373 \cs_new_eq:NN \_template_store_value_commalist:n \_template_store_value_tokenlist:n
```

(End of definition for _template_store_value_aux:Nn and others.)

11.6 Implementation part of template declaration

`_template_declare_template_code:nmmnn`

The main function for implementing a template starts with a couple of simple checks to make sure that there are no obvious mistakes: the number of arguments must agree and the template keys must have been declared.

`_template_declare_template_code:nmmnn`

```
374 \cs_new_protected:Npn \_template_declare_template_code:nmmnn #1#2#3#4#5
375   {
376     \_template_execute_if_type_exist:nT {#1}
377     {
378       \_template_execute_if_arg_agree:nnT {#1} {#3}
379       {
380         \_template_if_keys_exist:nnT {#1} {#2}
381         {
382           \_template_store_key_implementation:nmm {#1} {#2} {#4}
383           \str_if_in:nnTF {#5} { AssignTemplateKeys }

```

```

384         {
385             \regex_match:nnTF { \c { AssignTemplateKeys } } {#5}
386             { \_template_declare_template_code:nnnn {#1} {#2} {#3} {#5} }
387             {
388                 \_template_declare_template_code:nnnn
389                 {#1} {#2} {#3} { \AssignTemplateKeys #5 }
390             }
391         }
392     {
393         \_template_declare_template_code:nnnn
394         {#1} {#2} {#3} { \AssignTemplateKeys #5 }
395     }
396 }
397 }
398 }
399 }
400 \cs_new_protected:Npn \_template_declare_template_code:nnnn #1#2#3#4
401 {
402     \cs_if_exist:cT { \c__template_code_root_tl #1 / #2 }
403     { \msg_info:nnnn { template } { declare-template-code } {#1} {#2} }
404     \cs_generate_from_arg_count:cNnn
405     { \c__template_code_root_tl #1 / #2 }
406     \cs_gset_protected:Npn {#3} {#4}
407 }

```

(End of definition for `_template_declare_template_code:nnnnn` and `_template_declare_template_code:nnnn`.)

`_template_store_key_implementation:nnn`

Actually storing the implementation part of a template is quite easy as it only requires the list of keys given to be turned into a property list. There is also some error-checking to do, hence the need to have the list of defined keytypes available. In certain cases (when choices are involved) parsing the key results in changes to the default values. That is why they are loaded and then saved again.

```

408 \cs_new_protected:Npn \_template_store_key_implementation:nnn #1#2#3
409 {
410     \_template_recover_defaults:nn {#1} {#2}
411     \_template_recover_keytypes:nn {#1} {#2}
412     \prop_clear:N \l__template_vars_prop
413     \keyval_parse:nnn
414     { \_template_parse_vars_elt:n } { \_template_parse_vars_elt:nnn { #1 / #2 } } {#3}
415     \_template_store_vars:nn {#1} {#2}
416     \prop_map_inline:Nn \l__template_keytypes_prop
417     { \msg_error:nnnnn { template } { key-not-implemented } {##1} {#2} {#1} }
418 }

```

(End of definition for `_template_store_key_implementation:nnn`.)

`_template_parse_vars_elt:n`

At the implementation stage, every key must have a value given. So this is an error function.

```

419 \cs_new_protected:Npn \_template_parse_vars_elt:n #1
420 { \msg_error:nnn { template } { key-no-variable } {#1} }

```

(End of definition for `_template_parse_vars_elt:n`.)

`_template_parse_vars_elt:nnn
 _template_parse_vars_elt_aux:nn
 _template_parse_vars_elt_aux:nw
 _template_parse_vars_elt_aux:nnn
 _template_parse_vars_elt_aux:nne
 _template_parse_vars_elt_key:nn`

The actual storage part here is very simple: the storage bin name is placed into the property list. At the same time, a comparison is made with the keytypes defined earlier: if there is a mismatch then an error is raised.

```

421 \cs_new_protected:Npn \_template_parse_vars_elt:nnn #1#2#3
422 {
423   \tl_set:Ne \l__template_key_name_tl
424     { \tl_trim_spaces:e { \tl_to_str:n {#2} } }
425   \prop_get:NVNTF \l__template_keytypes_prop
426     \l__template_key_name_tl
427     \l__template_keytype_tl
428     {
429       \_template_split_keytype_arg:V \l__template_keytype_tl
430       \_template_parse_vars_elt_aux:nn {#1} {#3}
431       \prop_remove:NV \l__template_keytypes_prop \l__template_key_name_tl
432     }
433     { \msg_error:nnn { template } { unknown-key } {#2} }
434 }
  
```

Split off any leading global and they look for the way to implement.

```

435 \cs_new_protected:Npn \_template_parse_vars_elt_aux:nn #1#2
436 {
437   \_template_parse_vars_elt_aux:nw {#1} #2 global global \s__template_stop
438 }
439 \cs_new_protected:Npn \_template_parse_vars_elt_aux:nw
440   #1#2 global #3 global #4 \s__template_stop
441 {
442   \tl_if_blank:nTF {#4}
443     { \_template_parse_vars_elt_aux:nnn {#1} { } {#2} }
444     {
445       \tl_if_blank:nTF {#2}
446         {
447           \_template_parse_vars_elt_aux:nne
448             {#1} { global } { \tl_trim_spaces:n {#3} }
449         }
450         { \msg_error:nnn { template } { bad-variable } { #2 global #3 } }
451     }
452 }
453 \cs_new_protected:Npn \_template_parse_vars_elt_aux:nnn #1#2#3
454 {
455   \str_case:VnF \l__template_keytype_tl
456   {
457     { choice } { \_template_implement_choices:nn {#1} {#3} }
458     { function }
459     {
460       \cs_if_exist:NF #3
461         { \cs_new:Npn #3 { } }
462       \_template_parse_vars_elt_key:nn {#1}
463       {
464         .code:n =
465         {
466           \cs_generate_from_arg_count:NNnn
467             \exp_not:N #3
468             \exp_not:c
469             { cs_ \str_if_eq:nnT {#1} { global } { g } set:Npn }
  
```

```

470             { \exp_not:V \l__template_keytype_arg_tl }
471             {##1}
472         }
473     }
474     \prop_put:NVn \l__template_vars_prop
475     \l__template_key_name_tl {#2#3}
476 }
477 { instance }
478 {
479     \__template_parse_vars_elt_key:nn {#1}
480     {
481         .code:n =
482         {
483             \exp_not:c
484             { cs_ \str_if_eq:nnT {#1} { global } { g } set:Npn }
485             \exp_not:N #3 { \UseInstance {##1} }
486         }
487     }
488     \prop_put:NVn \l__template_vars_prop
489     \l__template_key_name_tl {#2#3}
490 }
491 }
492 {
493     \tl_if_single:nTF {#3}
494     {
495         \cs_if_exist:NF #3
496         { \use:c { \__template_map_var_type: _new:N } #3 }
497         \__template_parse_vars_elt_key:nn {#1}
498         {
499             . \__template_map_var_type:
500             _ \str_if_eq:nnT {#1} { global } { g } set:N
501             = \exp_not:N #3
502         }
503         \prop_put:NVn \l__template_vars_prop
504         \l__template_key_name_tl {#2#3}
505     }
506     { \msg_error:nnn { template } { bad-variable } {#2#3} }
507 }
508 }
509 \cs_generate_variant:Nn \__template_parse_vars_elt_aux:nnn { nne }
510 \cs_new_protected:Npn \__template_parse_vars_elt_key:nn #1#2
511 {
512     \keys_define:ne { template / #1 }
513     { \l__template_key_name_tl #2 }
514 }

```

(End of definition for `__template_parse_vars_elt:nnn` and others.)

`__template_map_var_type:` Turn a “friendly” variable type into an `expl3` one.

```

515 \cs_new:Npn \__template_map_var_type:
516 {
517     \str_case:Vn \l__template_keytype_tl
518     {
519         { boolean } { bool }

```

```

520     { commalist } { clist }
521     { integer }  { int }
522     { length }   { dim }
523     { muskip }   { muskip }
524     { real }     { fp }
525     { skip }     { skip }
526     { tokenlist } { tl }
527   }
528 }

```

(End of definition for `__template_map_var_type:`.)

```

\__template_implement_choices:nn
\__template_implement_choices_default:

```

Implementing choices requires a second key–value loop. So after a little set-up, the standard parser is called.

```

529 \cs_new_protected:Npn \__template_implement_choices:nn #1#2
530 {
531   \clist_set:NV \l__template_tmp_clist \l__template_keytype_arg_tl
532   \prop_put:NVn \l__template_vars_prop \l__template_key_name_tl { }
533   \keys_define:ne { template / #1 } { \l__template_key_name_tl .choice: }
534   \keyval_parse:nnn
535     { \__template_implement_choice_elt:n }
536     { \__template_implement_choice_elt:nnn {#1} }
537     {#2}
538   \prop_get:NVNT \l__template_values_prop \l__template_key_name_tl
539     \l__template_tmp_tl
540     { \__template_implement_choices_default: }
541   \clist_if_empty:NF \l__template_tmp_clist
542     {
543       \clist_map_inline:Nn \l__template_tmp_clist
544         { \msg_error:nnn { template } { choice-not-implemented } {##1} }
545     }
546 }

```

A sanity check for the default value, so that an error is raised now and not when converting to assignments.

```

547 \cs_new_protected:Npn \__template_implement_choices_default:
548 {
549   \tl_set:Ne \l__template_tmp_tl
550     { \l__template_key_name_tl \c_space_tl \l__template_tmp_tl }
551   \prop_if_in:NVF \l__template_vars_prop \l__template_tmp_tl
552     {
553       \tl_set:Ne \l__template_tmp_tl
554         { \l__template_key_name_tl \c_space_tl \l__template_tmp_tl }
555       \prop_if_in:NVF \l__template_vars_prop \l__template_tmp_tl
556         {
557           \prop_get:NVN \l__template_keytypes_prop \l__template_key_name_tl
558             \l__template_tmp_tl
559           \__template_split_keytype_arg:V \l__template_tmp_tl
560           \prop_get:NVN \l__template_values_prop \l__template_key_name_tl
561             \l__template_tmp_tl
562           \msg_error:nnVV { template } { unknown-default-choice }
563             \l__template_key_name_tl
564             \l__template_key_name_tl
565         }
566     }

```



```
567 }
```

(End of definition for `__template_implement_choices:nn` and
`__template_implement_choices_default:.`)

```
\__template_implement_choice_elt:nnn  
\__template_implement_choice_elt_aux:nnn  
\__template_implement_choice_elt_aux:n  
\__template_implement_choice_elt:n
```

The actual storage of the implementation of a choice is mainly about error checking. The code here ensures that all choices have to have been declared, apart from the special **unknown** choice, which must come last. The code for each choice is stored along with the key name in the variables property list.

```
568 \cs_new_protected:Npn \__template_implement_choice_elt:nnn #1#2#3  
569 {  
570   \clist_if_empty:NTF \l__template_tmp_clist  
571   {  
572     \str_if_eq:nnTF {#2} { unknown }  
573     { \__template_implement_choice_elt_aux:nnn {#1} {#2} {#3} }  
574     { \__template_implement_choice_elt_aux:n {#2} }  
575   }  
576   {  
577     \clist_if_in:NnTF \l__template_tmp_clist {#2}  
578     {  
579       \clist_remove_all:Nn \l__template_tmp_clist {#2}  
580       \__template_implement_choice_elt_aux:nnn {#1} {#2} {#3}  
581     }  
582     { \__template_implement_choice_elt_aux:n {#2} }  
583   }  
584 }  
585 \cs_new_protected:Npn \__template_implement_choice_elt_aux:n #1  
586 {  
587   \prop_get:NVN \l__template_keytypes_prop \l__template_key_name_tl  
588   \l__template_tmp_tl  
589   \__template_split_keytype_arg:V \l__template_tmp_tl  
590   \msg_error:nnVn { template } { unknown-choice } \l__template_key_name_tl {#1}  
591 }  
592 \cs_new_protected:Npn \__template_implement_choice_elt_aux:nnn #1#2#3  
593 {  
594   \keys_define:ne { template / #1 }  
595   { \l__template_key_name_tl / #2 .code:n = { \exp_not:n {#3} } }  
596   \tl_set:Ne \l__template_tmp_tl  
597   { \l__template_key_name_tl \c_space_tl #2 }  
598   \prop_put:NVN \l__template_vars_prop \l__template_tmp_tl {#3}  
599 }  
600 \cs_new_protected:Npn \__template_implement_choice_elt:n #1  
601 {  
602   \msg_error:nnVn { template } { choice-requires-code }  
603   \l__template_key_name_tl {#1}  
604 }
```

(End of definition for `__template_implement_choice_elt:nnn` and others.)

11.7 Editing template defaults

```
\__template_edit_defaults:nnn
```

Editing the template defaults means getting the values back out of the store, then parsing the list of new values before putting the updated list back into storage.

```
605 \cs_new_protected:Npn \__template_edit_defaults:nnn #1#2#3
```

```

606 {
607   \__template_if_keys_exist:nnT {#1} {#2}
608   {
609     \__template_recover_defaults:nn {#1} {#2}
610     \__template_parse_values:nnn {#1} {#2} {#3}
611     \__template_store_defaults:nn {#1} {#2}
612   }
613 }

```

(End of definition for __template_edit_defaults:nnn.)

__template_parse_values:nnn The routine to parse values is the same for both editing a template and setting up an instance. So the code here does only the minimum necessary for reading the values.

```

614 \cs_new_protected:Npn \__template_parse_values:nnn #1#2#3
615 {
616   \__template_recover_keytypes:nn {#1} {#2}
617   \keyval_parse:NNn
618   \__template_parse_values_elt:n \__template_parse_values_elt:n {#3}
619 }

```

(End of definition for __template_parse_values:nnn.)

__template_parse_values_elt:n Every key needs a value, so this is just an error routine.

```

620 \cs_new_protected:Npn \__template_parse_values_elt:n #1
621 {
622   \bool_set_true:N \l__template_error_bool
623   \msg_error:nnn { template } { key-no-value } {#1}
624 }

```

(End of definition for __template_parse_values_elt:n.)

__template_parse_values_elt:n To store the value, find the keytype then call the saving function. These need the current key name saved as \l__template_key_name_tl.

__template_parse_values_elt_aux:n

```

625 \cs_new_protected:Npn \__template_parse_values_elt:n #1#2
626 {
627   \tl_set:Ne \l__template_key_name_tl
628   { \tl_trim_spaces:e { \tl_to_str:n {#1} } }
629   \prop_get:NVNTF \l__template_keytypes_prop \l__template_key_name_tl
630   \l__template_tmp_tl
631   { \__template_parse_values_elt_aux:n {#2} }
632   { \msg_error:nnV { template } { unknown-key } \l__template_key_name_tl }
633 }
634 \cs_new_protected:Npn \__template_parse_values_elt_aux:n #1
635 {
636   \__template_split_keytype_arg:V \l__template_tmp_tl
637   \use:c { __template_store_value_ \l__template_keytype_tl :n } {#1}
638 }

```

(End of definition for __template_parse_values_elt:n and __template_parse_values_elt_aux:n.)

__template_template_set_eq:nnn To copy a template, each of the lists plus the code has to be copied across. To keep this independent of the list storage system, it is all done with two-part shuffles.

```

639 \cs_new_protected:Npn \__template_template_set_eq:nnn #1#2#3
640 {
641   \__template_recover_defaults:nn {#1} {#3}

```

```

642     \__template_store_defaults:nn {#1} {#2}
643     \__template_recover_keytypes:nn {#1} {#3}
644     \__template_store_keytypes:nn {#1} {#2}
645     \__template_recover_vars:nn {#1} {#3}
646     \__template_store_vars:nn {#1} {#2}
647     \cs_if_exist:cT { \c__template_code_root_tl #1 / #2 }
648       { \msg_info:nxxx { template } { declare-template-code } {#1} {#2} }
649     \cs_gset_eq:cc { \c__template_code_root_tl #1 / #2 }
650       { \c__template_code_root_tl #1 / #3 }
651   }

```

(End of definition for `__template_template_set_eq:nnn`.)

11.8 Creating instances of templates

```

\__template_declare_instance:nnnn
\__template_declare_instance_aux:nnnn

```

Making an instance has two distinct parts. First, the keys given are parsed to transfer the values into the structured data format used internally. This allows the default and given values to be combined with no repetition. In the second step, the structured data is converted to pre-defined variable assignments, and these are stored in the function for the instance.

```

652 \cs_new_protected:Npn \__template_declare_instance:nnnn #1#2#3#4
653   {
654     \__template_execute_if_code_exist:nnT {#1} {#2}
655     {
656       \__template_recover_defaults:nn {#1} {#2}
657       \__template_recover_vars:nn {#1} {#2}
658       \__template_declare_instance_aux:nnnn {#1} {#2} {#3} {#4}
659     }
660   }
661 \cs_new_protected:Npn \__template_declare_instance_aux:nnnn #1#2#3#4
662   {
663     \bool_set_false:N \l__template_error_bool
664     \__template_parse_values:nnn {#1} {#2} {#4}
665     \bool_if:NF \l__template_error_bool
666     {
667       \prop_put:Nnn \l__template_values_prop { from-template } {#2}
668       \__template_store_values:nn {#1} {#3}
669       \__template_convert_to_assignments:
670       \cs_if_exist:cT { \c__template_instances_root_tl #1 / #3 }
671         { \msg_info:nxxx { template } { declare-instance } {#3} {#1} }
672       \cs_set_protected:cpe { \c__template_instances_root_tl #1 / #3 }
673         {
674           \exp_not:N \__template_assignments_push:n
675             { \exp_not:V \l__template_assignments_tl }
676           \exp_not:c { \c__template_code_root_tl #1 / #2 }
677         }
678     }
679   }

```

(End of definition for `__template_declare_instance:nnnn` and `__template_declare_instance_aux:nnnn`.)

```

\__template_instance_set_eq:nnn

```

Copy-paste an instance.

```

680 \cs_new_protected:Npn \__template_instance_set_eq:nnn #1#2#3

```

```

681 {
682   \__template_if_instance_exist:nnTF {#1} {#3}
683   {
684     \__template_recover_values:nn {#1} {#3}
685     \__template_store_values:nn {#1} {#2}
686     \cs_if_exist:cT { \c__template_instances_root_tl #1 / #2 }
687     { \msg_info:nnnn { template } { declare-instance } {#2} {#1} }
688     \cs_set_eq:cc { \c__template_instances_root_tl #1 / #2 }
689     { \c__template_instances_root_tl #1 / #3 }
690   }
691   { \msg_error:nnnn { template } { unknown-instance } {#1} {#3} }
692 }

```

(End of definition for __template_instance_set_eq:nnn.)

__template_edit_instance:nnn Editing an instance is almost identical to declaring one. The only variation is the source of the values to use. When editing, they are recovered from the previous instance run.

```

\__template_edit_instance_aux:nnnn
\__template_edit_instance_aux:nVnnn
693 \cs_new_protected:Npn \__template_edit_instance:nnn #1#2#3
694 {
695   \__template_if_instance_exist:nnTF {#1} {#2}
696   {
697     \__template_recover_values:nn {#1} {#2}
698     \prop_get:NnN \l__template_values_prop { from-template }
699     \l__template_tmp_tl
700     \__template_edit_instance_aux:nVnn
701     {#1} \l__template_tmp_tl {#2} {#3}
702   }
703   { \msg_error:nnnn { template } { unknown-instance } {#1} {#2} }
704 }
705 \cs_new_protected:Npn \__template_edit_instance_aux:nnnn #1#2#3#4
706 {
707   \__template_recover_vars:nn {#1} {#2}
708   \__template_declare_instance_aux:nnnn {#1} {#2} {#3} {#4}
709 }
710 \cs_generate_variant:Nn \__template_edit_instance_aux:nnnn { nV }

```

(End of definition for __template_edit_instance:nnn and __template_edit_instance_aux:nnnn.)

__template_convert_to_assignments: The idea on converting to a set of assignments is to loop over each key, so that the loop order follows the declaration order of the keys. This is done using a sequence as property lists are not “ordered”.

```

\__template_convert_to_assignments_aux:n
\__template_convert_to_assignments_aux:nn
\__template_convert_to_assignments_aux:nV
711 \cs_new_protected:Npn \__template_convert_to_assignments:
712 {
713   \tl_clear:N \l__template_assignments_tl
714   \seq_map_function:NN \l__template_key_order_seq
715   \__template_convert_to_assignments_aux:n
716 }
717 \cs_new_protected:Npn \__template_convert_to_assignments_aux:n #1
718 {
719   \prop_get:NnN \l__template_keytypes_prop {#1} \l__template_tmp_tl
720   \__template_convert_to_assignments_aux:nV {#1} \l__template_tmp_tl
721 }

```

The second auxiliary function actually does the work. The arguments here are the key name (#1) and the keytype (#2). From those, the value to assign and the name of the appropriate variable are recovered. A bit of work is then needed to sort out keytypes with arguments (for example instances), and to look for global assignments. Once that is done, a hand-off can be made to the handler for the relevant keytype.

```

722 \cs_new_protected:Npn \__template_convert_to_assignments_aux:nn #1#2
723 {
724   \prop_get:NnNT \l__template_values_prop {#1} \l__template_value_tl
725   {
726     \prop_get:NnNTF \l__template_vars_prop {#1} \l__template_var_tl
727     {
728       \__template_split_keytype_arg:n {#2}
729       \str_if_eq:VnF \l__template_keytype_tl { choice }
730       {
731         \str_if_eq:VnF \l__template_keytype_tl { code }
732         { \__template_find_global: }
733       }
734       \tl_set:Nn \l__template_key_name_tl {#1}
735       \cs_if_exist_use:cF { __template_assign_ \l__template_keytype_tl : }
736       { \__template_assign_variable: }
737     }
738     { \msg_error:nnn { template } { unknown-attribute } {#1} }
739   }
740 }
741 \cs_generate_variant:Nn \__template_convert_to_assignments_aux:nn { nV }

```

(End of definition for `__template_convert_to_assignments:`,
`__template_convert_to_assignments_aux:n`, and `__template_convert_to_assignments_aux:nn`.)

`__template_find_global:` Global assignments should have the phrase `global` at the front. This is pretty easy to find: no other error checking, though.

`__template_find_global_aux:w`

```

742 \cs_new_protected:Npn \__template_find_global:
743 {
744   \bool_set_false:N \l__template_global_bool
745   \tl_if_in:onT \l__template_var_tl { global }
746   {
747     \exp_after:wN \__template_find_global_aux:w \l__template_var_tl \s__template_stop
748   }
749 }
750 \cs_new_protected:Npn \__template_find_global_aux:w #1 global #2 \s__template_stop
751 {
752   \tl_set:Nn \l__template_var_tl {#2}
753   \bool_set_true:N \l__template_global_bool
754 }

```

(End of definition for `__template_find_global:` and `__template_find_global_aux:w`.)

11.9 Using templates directly

`__template_use_template:nnn` Directly use a template with a particular parameter setting. This is also picked up if used in a nested fashion inside a parameter list. The idea is essentially the same as creating an instance, just with no saving of the result.

```

755 \cs_new_protected:Npn \__template_use_template:nnn #1#2#3

```

```

756 {
757   \__template_execute_if_code_exist:nnT {#1} {#2}
758   {
759     \__template_recover_defaults:nn {#1} {#2}
760     \__template_recover_vars:nn {#1} {#2}
761     \__template_parse_values:nnn {#1} {#2} {#3}
762     \__template_convert_to_assignments:
763     \use:c { \c__template_code_root_tl #1 / #2 }
764   }
765 }

```

(End of definition for __template_use_template:nnn.)

11.10 Assigning values to variables

__template_assign_boolean: Setting a Boolean value is slightly different to everything else as the value can be used to work out which `set` function to call. As long as there is no need to recover things from another variable, everything is pretty easy. If there is, then we need to allow for the fact that the recovered value here will *not* be expandable, so needs to be converted to something that is.

```

766 \cs_new_protected:Npn \__template_assign_boolean:
767 {
768   \bool_if:NTF \l__template_global_bool
769     { \__template_assign_boolean_aux:n { bool_gset } }
770     { \__template_assign_boolean_aux:n { bool_set } }
771 }
772 \cs_new_protected:Npn \__template_assign_boolean_aux:n #1
773 {
774   \__template_if_key_value:VTF \l__template_value_tl
775   {
776     \__template_key_to_value:
777     \tl_put_right:Ne \l__template_assignments_tl
778     {
779       \exp_not:c { #1 _eq:NN }
780       \exp_not:V \l__template_var_tl
781       \exp_not:V \l__template_value_tl
782     }
783   }
784   {
785     \tl_put_right:Ne \l__template_assignments_tl
786     {
787       \exp_not:c { #1 _ \l__template_value_tl :N }
788       \exp_not:V \l__template_var_tl
789     }
790   }
791 }

```

(End of definition for __template_assign_boolean: and __template_assign_boolean_aux:n.)

__template_assign_choice: The idea here is to find either the choice as-given or else the special unknown choice, and to copy the appropriate code across.

```

\__template_assign_choice_aux:nF
\__template_assign_choice_aux:eF
792 \cs_new_protected:Npn \__template_assign_choice:
793 {
794   \__template_assign_choice_aux:eF

```

```

795     { \l__template_key_name_tl \c_space_tl \l__template_value_tl }
796     {
797         \__template_assign_choice_aux:eF
798         { \l__template_key_name_tl \c_space_tl unknown }
799         {
800             \prop_get:NVN \l__template_keytypes_prop \l__template_key_name_tl
801             \l__template_tmp_tl
802             \__template_split_keytype_arg:V \l__template_tmp_tl
803             \msg_error:nnVV { template } { unknown-choice }
804             \l__template_key_name_tl
805             \l__template_value_tl
806         }
807     }
808 }
809 \cs_new_protected:Npn \__template_assign_choice_aux:nF #1
810 {
811     \prop_get:NnNTF \l__template_vars_prop {#1} \l__template_tmp_tl
812     { \tl_put_right:NV \l__template_assignments_tl \l__template_tmp_tl }
813 }
814 \cs_generate_variant:Nn \__template_assign_choice_aux:nF { e }

```

(End of definition for __template_assign_choice: and __template_assign_choice_aux:nF.)

__template_assign_function: This looks a bit messy but is only actually one function.

```

\__template_assign_function_aux:N
815 \cs_new_protected:Npn \__template_assign_function:
816 {
817     \bool_if:NTF \l__template_global_bool
818     { \__template_assign_function_aux:N \cs_gset:Npn }
819     { \__template_assign_function_aux:N \cs_set:Npn }
820 }
821 \cs_new_protected:Npn \__template_assign_function_aux:N #1
822 {
823     \tl_put_right:Ne \l__template_assignments_tl
824     {
825         \cs_generate_from_arg_count:NNnn
826         \exp_not:V \l__template_var_tl
827         \exp_not:N #1
828         { \exp_not:V \l__template_keytype_arg_tl }
829         { \exp_not:V \l__template_value_tl }
830     }
831 }

```

(End of definition for __template_assign_function: and __template_assign_function_aux:N.)

__template_assign_instance: Using an instance means adding the appropriate function creation to the tl. No checks are made at this stage, so if the instance is not valid then errors will arise later.

```

\__template_assign_instance_aux:N
832 \cs_new_protected:Npn \__template_assign_instance:
833 {
834     \bool_if:NTF \l__template_global_bool
835     { \__template_assign_instance_aux:N \cs_gset_protected:Npn }
836     { \__template_assign_instance_aux:N \cs_set_protected:Npn }
837 }
838 \cs_new_protected:Npn \__template_assign_instance_aux:N #1
839 {

```

```

840 \tl_put_right:Ne \l__template_assignments_tl
841 {
842   \exp_not:N #1 \exp_not:V \l__template_var_tl
843   {
844     \__template_use_instance:nn
845     { \exp_not:V \l__template_keytype_arg_tl }
846     { \exp_not:V \l__template_value_tl }
847   }
848 }
849 }

```

(End of definition for `__template_assign_instance:` and `__template_assign_instance_aux:N`.)

`__template_assign_variable:` A general-purpose function for all of the other assignments. As long as the value is not coming from another variable, the stored value is simply transferred for output. We use V-type expansion for the `\KeyValue` case: for token lists this is essential, whilst for register-based variables, it does no harm and avoids needing a low-level test.

`__template_assign_variable:n`

```

850 \cs_new_protected:Npn \__template_assign_variable:
851 {
852   \exp_args:Ne \__template_assign_variable:n
853   {
854     \__template_map_var_type:
855     -
856     \bool_if:NT \l__template_global_bool { g }
857     set:N
858   }
859 }

```

Notice we need a V-type variant for each (g)set operation here: these need to be provided by `expl3`.

```

860 \cs_new_protected:Npn \__template_assign_variable:n #1
861 {
862   \__template_if_key_value:VTF \l__template_value_tl
863   {
864     \__template_key_to_value:
865     \tl_put_right:Ne \l__template_assignments_tl
866     {
867       \exp_not:c { #1 V } \exp_not:V \l__template_var_tl
868       \exp_not:V \l__template_value_tl
869     }
870   }
871   {
872     \tl_put_right:Ne \l__template_assignments_tl
873     {
874       \exp_not:c { #1 n } \exp_not:V \l__template_var_tl
875       { \exp_not:V \l__template_value_tl }
876     }
877   }
878 }

```

(End of definition for `__template_assign_variable:` and `__template_assign_variable:n`.)

`__template_key_to_value:` The idea here is to recover the attribute value of another key. To do that, the marker is removed and a look up takes place. If this is successful, then the name of the variable of the attribute is returned. This assumes that the value will be used in context where it

`__template_key_to_value_auxi:w`
`__template_key_to_value_auxii:w`

will be converted to a value, for example when setting a number. There is also a need to check in case the copied value happens to be `global`.

```

879 \cs_new_protected:Npn \__template_key_to_value:
880   { \exp_after:wN \__template_key_to_value_auxi:w \l__template_value_tl }
881 \cs_new_protected:Npn \__template_key_to_value_auxi:w \KeyValue #1
882   {
883     \tl_set:Ne \l__template_tmp_tl { \tl_trim_spaces:e { \tl_to_str:n {#1} } }
884     \prop_get:NVNTF \l__template_vars_prop \l__template_tmp_tl
885       \l__template_value_tl
886     {
887       \exp_after:wN \__template_key_to_value_auxii:w \l__template_value_tl
888       \s__template_mark global \q__template_nil \s__template_stop
889     }
890     { \msg_error:nnV { template } { unknown-attribute } \l__template_tmp_tl }
891   }
892 \cs_new_protected:Npn \__template_key_to_value_auxii:w #1 global #2#3 \s__template_stop
893   {
894     \__template_quark_if_nil:NF #2
895     { \tl_set:Nn \l__template_value_tl {#2} }
896   }

```

(End of definition for `__template_key_to_value:`, `__template_key_to_value_auxi:w`, and `__template_key_to_value_auxii:w`.)

11.11 Using instances

`__template_use_instance:nn` Using an instance is just a question of finding the appropriate function. If nothing is found, an error is raised. One complication is that if the first token of argument #2 is `\UseTemplate` then that is also valid. There is an error-test to make sure that the types agree, and if so the template is used directly.

```

897 \cs_new_protected:Npn \__template_use_instance:nn #1#2
898   {
899     \__template_if_use_template:nTF {#2}
900     { \__template_use_instance_aux:nNnnn {#1} #2 }
901     { \__template_use_instance_aux:nn {#1} {#2} }
902   }
903 \cs_new_protected:Npn \__template_use_instance_aux:nNnnn #1#2#3#4#5
904   {
905     \str_if_eq:nnTF {#1} {#3}
906     { \__template_use_template:nnn {#3} {#4} {#5} }
907     { \msg_error:nnnn { template } { type-mismatch } {#1} {#3} }
908   }
909 \cs_new_protected:Npn \__template_use_instance_aux:nn #1#2
910   {
911     \__template_if_instance_exist:nnTF {#1} {#2}
912     { \use:c { \c__template_instances_root_tl #1 / #2 } }
913     { \msg_error:nnnn { template } { unknown-instance } {#1} {#2} }
914   }

```

(End of definition for `__template_use_instance:nn`, `__template_use_instance_aux:nNnnn`, and `__template_use_instance_aux:nn`.)

11.12 Assignment manipulation

A few functions to transfer assignments about, as this is needed by `\AssignTemplateKeys`.

```
\_template_assignments_pop: To actually use the assignments.
915 \cs_new:Npn \_template_assignments_pop: { \l__template_assignments_tl }
(End of definition for \_template_assignments_pop:.)

\_template_assignments_push:n Here, the assignments are stored for later use.
916 \cs_new_protected:Npn \_template_assignments_push:n #1
917 { \tl_set:Nn \l__template_assignments_tl {#1} }
(End of definition for \_template_assignments_push:n.)
```

11.13 Showing templates and instances

```
\_template_show_code:nn Showing the code for a template is just a translation of \cs_show:c.
918 \cs_new_protected:Npn \_template_show_code:nn #1#2
919 { \cs_show:c { \c__template_code_root_tl #1 / #2 } }
(End of definition for \_template_show_code:nn.)

\_template_show_defaults:nn A modified version of the property-list printing code, such that the output refers to
\_template_show_keytypes:nn templates and instances rather than to the underlying structures.
\_template_show_vars:nn
\_template_show:Nnnn
920 \cs_new_protected:Npn \_template_show_defaults:nn #1#2
921 {
922   \_template_if_keys_exist:nnT {#1} {#2}
923   {
924     \_template_recover_defaults:nn {#1} {#2}
925     \_template_show:Nnnn \l__template_values_prop
926     {#1} {#2} { default-values }
927   }
928 }
929 \cs_new_protected:Npn \_template_show_keytypes:nn #1#2
930 {
931   \_template_if_keys_exist:nnT {#1} {#2}
932   {
933     \_template_recover_keytypes:nn {#1} {#2}
934     \_template_show:Nnnn \l__template_keytypes_prop
935     {#1} {#2} { interface }
936   }
937 }
938 \cs_new_protected:Npn \_template_show_vars:nn #1#2
939 {
940   \_template_execute_if_code_exist:nnT {#1} {#2}
941   {
942     \_template_recover_vars:nn {#1} {#2}
943     \_template_show:Nnnn \l__template_vars_prop
944     {#1} {#2} { variable-mapping }
945   }
946 }
947 \cs_new_protected:Npn \_template_show:Nnnn #1#2#3#4
948 {
```

```

949 \msg_show:nneeee { template } { show-attribute }
950   { \tl_to_str:n {#2} }
951   { \tl_to_str:n {#3} }
952   { \tl_to_str:n {#4} }
953   { \prop_map_function:NN #1 \msg_show_item_unbraced:nn }
954 }

```

(End of definition for `__template_show_defaults:nn` and others.)

`__template_show_values:nn` Instance values are a little more complex, as is the template to consider.

```

955 \cs_new_protected:Npn \__template_show_values:nn #1#2
956   {
957     \__template_if_instance_exist:nnT {#1} {#2}
958     {
959       \__template_recover_values:nn {#1} {#2}
960       \msg_show:nneee { template } { show-values }
961       { \tl_to_str:n {#1} }
962       { \tl_to_str:n {#2} }
963       {
964         \prop_map_function:NN \l__template_values_prop
965         \msg_show_item_unbraced:nn
966       }
967     }
968 }

```

(End of definition for `__template_show_values:nn`.)

11.14 Messages

The text for error messages: short and long text for all of them.

```

969 \msg_new:nnnn { template } { argument-number-mismatch }
970   { Template-type~'#1'~takes~#2~argument(s). }
971   {
972     Templates~of~type~'#1'~require~#2~argument(s).\
973     You~have~tried~to~make~a~template~for~'#1'~
974     with~#3~argument(s),~which~is~not~possible:~
975     the~number~of~arguments~must~agree.
976   }
977 \msg_new:nnnn { template } { bad-number-of-arguments }
978   { Bad~number~of~arguments~for~template~type~'#1'. }
979   {
980     A~template~may~accept~between~0~and~9~arguments.\
981     You~asked~to~use~#2~arguments:~this~is~not~supported.
982   }
983 \msg_new:nnnn { template } { bad-variable }
984   { Incorrect~variable~description~'#1'. }
985   {
986     The~argument~'#1'~is~not~of~the~form \
987     ~'<variable>' \
988     ~or~ \
989     ~'global~<variable>' \
990     It~must~be~given~in~one~of~these~formats~to~be~used~in~a~template.
991   }
992 \msg_new:nnnn { template } { choice-not-implemented }

```

```

993 { The~choice~'#1'~has~no~implementation. }
994 {
995   Each~choice~listed~in~the~interface~for~a~template~must~
996   have~an~implementation.
997 }
998 \msg_new:nnon { template } { choice-no-code }
999 { The~choice~'#1'~requires~implementation~details. }
1000 {
1001   When~creating~template~code~using~\DeclareTemplateCode,~
1002   each~choice~name~must~have~an~associated~implementation.\\
1003   This~should~be~given~after~a~'= '~sign:~LaTeX~did~not~find~one.
1004 }
1005 \msg_new:nnon { template } { choice-requires-code }
1006 { The~choice~'#2'~for~key~'#1'~requires~an~implementation. }
1007 {
1008   You~should~have~put:\\
1009   \ \ #1~:~choice~{~#2 = <code> ~} \\
1010   but~LaTeX~did~not~find~any~<code>.
1011 }
1012 \msg_new:nnon { template } { duplicate-key-interface }
1013 { Key~'#1'~appears~twice~in~interface~definition~\msg_line_context:. }
1014 {
1015   Each~key~can~only~have~one~interface~declared~in~a~template.\\
1016   LaTeX~found~two~interfaces~for~'#1'.
1017 }
1018 \msg_new:nnon { template } { keytype-requires-argument }
1019 { The~key~type~'#1'~requires~an~argument~\msg_line_context:. }
1020 {
1021   You~should~have~put:\\
1022   \ \ <key-name>~:~#1~{~<argument>~} \\
1023   but~LaTeX~did~not~find~an~<argument>.
1024 }
1025 \msg_new:nnon { template } { invalid-keytype }
1026 { The~key~'#1'~is~missing~a~key~type~\msg_line_context:. }
1027 {
1028   Each~key~in~a~template~requires~a~key~type,~given~in~the~form:\\
1029   \ \ <key>~:~<key-type>\\
1030   LaTeX~could~not~find~a~<key-type>~in~your~input.
1031 }
1032 \msg_new:nnon { template } { key-no-value }
1033 { The~key~'#1'~has~no~value~\msg_line_context:. }
1034 {
1035   When~creating~an~instance~of~a~template~
1036   every~key~listed~must~include~a~value:\\
1037   \ \ <key>~::~~<value>
1038 }
1039 \msg_new:nnon { template } { key-no-variable }
1040 { The~key~'#1'~requires~implementation~details~\msg_line_context:. }
1041 {
1042   When~creating~template~code~using~\DeclareTemplateCode,~
1043   each~key~name~must~have~an~associated~implementation.\\
1044   This~should~be~given~after~a~'= '~sign:~LaTeX~did~not~find~one.
1045 }
1046 \msg_new:nnon { template } { key-not-implemented }

```

```

1047 { Key~'#1'~has-no-implementation~\msg_line_context:. }
1048 {
1049   The~definition~of~key~implementations~for~template~'#2'~
1050   of~template~type~'#3'~does~not~include~any~details~for~key~'#1'.\\
1051   The~key~was~declared~in~the~interface~definition,~
1052   and~so~an~implementation~is~required.
1053 }
1054 \msg_new:nnnn { template } { missing-keytype }
1055 { The~key~'#1'~is~missing~a~key~type~\msg_line_context:. }
1056 {
1057   Key~interface~definitions~should~be~of~the~form\\
1058   \ \ #1~::~~<key-type>\\
1059   but~LaTeX~could~not~find~a~<key-type>.
1060 }
1061 \msg_new:nnnn { template } { no-template-code }
1062 {
1063   The~template~'#2'~of~type~'#1'~is~unknown~
1064   or~has~no~implementation.
1065 }
1066 {
1067   There~is~no~code~available~for~the~template~name~given.\\
1068   This~should~be~given~using~\DeclareTemplateCode.
1069 }
1070 \msg_new:nnnn { template } { type-already-defined }
1071 { Template~type~'#1'~already~defined. }
1072 {
1073   You~have~used~\NewTemplateType~
1074   with~a~template~type~that~has~already~been~defined.
1075 }
1076 \msg_new:nnnn { template } { type-mismatch }
1077 { Template~types~'#1'~and~'#2'~do~not~agree. }
1078 {
1079   You~are~trying~to~use~a~template~directly~with~\UseInstance
1080   (or~a~similar~function),~but~the~template~types~do~not~match.
1081 }
1082 \msg_new:nnnn { template } { unknown-attribute }
1083 { The~template~attribute~'#1'~is~unknown. }
1084 {
1085   There~is~a~definition~in~the~current~template~reading\\
1086   \ \ \token_to_str:N \KeyValue {~#1~} \ \
1087   but~there~is~no~key~called~'#1'.
1088 }
1089 \msg_new:nnnn { template } { unknown-choice }
1090 { The~choice~'#2'~was~not~declared~for~key~'#1'. }
1091 {
1092   The~key~'#1'~takes~a~fixed~list~of~choices~
1093   and~this~list~does~not~include~'#2'.
1094 }
1095 \msg_new:nnnn { template } { unknown-default-choice }
1096 { The~default~choice~'#2'~was~not~declared~for~key~'#1'. }
1097 {
1098   The~key~'#1'~takes~a~fixed~list~of~choices~
1099   and~this~list~does~not~include~'#2'.
1100 }

```

```

1101 \msg_new:nnnn { template } { unknown-instance }
1102   { The~instance~'#2'~of~type~'#1'~is~unknown. }
1103   {
1104     You~have~asked~to~use~an~instance~'#2',~
1105     but~this~has~not~been~created.
1106   }
1107 \msg_new:nnnn { template } { unknown-key }
1108   { Unknown~template~key~'#1'. }
1109   {
1110     The~key~'#1'~was~not~declared~in~the~interface~
1111     for~the~current~template.
1112   }
1113 \msg_new:nnnn { template } { unknown-keytype }
1114   { The~key~type~'#1'~is~unknown. }
1115   {
1116     Valid~key~types~are:\\
1117     --boolean;\\
1118     --choice;\\
1119     --commalist;\\
1120     --function;\\
1121     --instance;\\
1122     --integer;\\
1123     --length;\\
1124     --muskip;\\
1125     --real;\\
1126     --skip;\\
1127     --tokenlist.
1128   }
1129 \msg_new:nnnn { template } { unknown-type }
1130   { The~template~type~'#1'~is~unknown. }
1131   {
1132     A~template~type~needs~to~be~defined~with~\NewTemplateType
1133     prior~to~using~it.
1134   }
1135 \msg_new:nnnn { template } { unknown-template }
1136   { The~template~'#2'~of~type~'#1'~is~unknown. }
1137   {
1138     No~interface~has~been~declared~for~a~template~
1139     '#2'~of~template~type~'#1'.
1140   }
1141   Information messages only have text: more text should not be needed.
1142 \msg_new:nnn { template } { declare-instance }
1143   { Declaring~instance~'#1'~of~type~'#2'~\msg_line_context:. }
1144 \msg_new:nnn { template } { declare-template-code }
1145   { Declaring~code~for~template~'#2'~of~template~type~'#1'~\msg_line_context:. }
1146 \msg_new:nnn { template } { declare-template-interface }
1147   {
1148     Declaring~interface~for~template~'#2'~of~template~type~'#1'~
1149     \msg_line_context:.
1150   }
1151 \msg_new:nnn { template } { declare-type }
1152   { Declaring~template~type~'#1'~taking~#2~argument(s)~\msg_line_context:. }
1153 \msg_new:nnn { template } { show-attribute }
1154   {

```

```

1154   The~template~'#2'~of~type~'#1'~has~
1155   \tl_if_empty:nTF {#4} { no~#3. } { #3 : #4 }
1156   }
1157 \msg_new:nnn { template } { show-values }
1158   {
1159   The~instance~'#2'~of~type~'#1'~has~
1160   \tl_if_empty:nTF {#3} { no~values. } { values: #3 }
1161   }

```

Also add template to the LaTeX messages.

```

1162 \prop_gput:Nnn \g_msg_module_type_prop { template } { LaTeX }

```

11.15 User functions

All simple translations.

```

\NewTemplateType \DeclareTemplateInterface \DeclareTemplateCode
\DeclareTemplateCopy \EditTemplateDefaults \UseTemplate
\DeclareInstance \DeclareInstanceCopy \EditInstance
\UseInstance
1163 \cs_new_protected:Npn \NewTemplateType #1#2
1164   { \__template_define_type:nn {#1} {#2} }
1165 \cs_new_protected:Npn \DeclareTemplateInterface #1#2#3#4
1166   { \__template_declare_template_keys:nnnn {#1} {#2} {#3} {#4} }
1167 \cs_new_protected:Npn \DeclareTemplateCode #1#2#3#4#5
1168   { \__template_declare_template_code:nnnnn {#1} {#2} {#3} {#4} {#5} }
1169 \cs_new_protected:Npn \DeclareTemplateCopy #1#2#3
1170   { \__template_template_set_eq:nnn {#1} {#2} {#3} }
1171 \cs_new_protected:Npn \EditTemplateDefaults #1#2#3
1172   { \__template_edit_defaults:nnn {#1} {#2} {#3} }
1173 \cs_new_protected:Npn \UseTemplate #1#2#3
1174   { \__template_use_template:nnn {#1} {#2} {#3} }
1175 \cs_new_protected:Npn \DeclareInstance #1#2#3#4
1176   { \__template_declare_instance:nnnn {#1} {#3} {#2} {#4} }
1177 \cs_new_protected:Npn \DeclareInstanceCopy #1#2#3
1178   { \__template_instance_set_eq:nnn {#1} {#2} {#3} }
1179 \cs_new_protected:Npn \EditInstance #1#2#3
1180   { \__template_edit_instance:nnn {#1} {#2} {#3} }
1181 \cs_new_protected:Npn \UseInstance #1#2
1182   { \__template_use_instance:nn {#1} {#2} }

```

(End of definition for \NewTemplateType and others. These functions are documented on page 351.)

The show functions are again just translation.

```

\ShowTemplateCode \ShowTemplateDefaults \ShowTemplateInterface
\ShowTemplateVariables \ShowInstanceValues
1183 \cs_new_protected:Npn \ShowTemplateCode #1#2
1184   { \__template_show_code:nn {#1} {#2} }
1185 \cs_new_protected:Npn \ShowTemplateDefaults #1#2
1186   { \__template_show_defaults:nn {#1} {#2} }
1187 \cs_new_protected:Npn \ShowTemplateInterface #1#2
1188   { \__template_show_keytypes:nn {#1} {#2} }
1189 \cs_new_protected:Npn \ShowTemplateVariables #1#2
1190   { \__template_show_vars:nn {#1} {#2} }
1191 \cs_new_protected:Npn \ShowInstanceValues #1#2
1192   { \__template_show_values:nn {#1} {#2} }

```

(End of definition for \ShowTemplateCode and others. These functions are documented on page 358.)

More direct translation.

```

\IfInstanceExistsT \IfInstanceExistsF \IfInstanceExistsTF
1193 \cs_new:Npn \IfInstanceExistsTF #1#2

```

```

1194 { \_template_if_instance_exist:nnTF {#1} {#2} }
1195 \cs_new:Npn \IfInstanceExistsT #1#2
1196 { \_template_if_instance_exist:nnT {#1} {#2} }
1197 \cs_new:Npn \IfInstanceExistsF #1#2
1198 { \_template_if_instance_exist:nnF {#1} {#2} }

```

(End of definition for `\IfInstanceExistsT`, `\IfInstanceExistsF`, and `\IfInstanceExistsTF`. These functions are documented on page 357.)

\KeyValue Simply dump the argument when executed: this should not happen.

```

1199 \cs_new_protected:Npn \KeyValue #1 {#1}

```

(End of definition for `\KeyValue`. This function is documented on page 353.)

\AssignTemplateKeys A short call to use a token register by proxy.

```

1200 \cs_new_protected:Npn \AssignTemplateKeys { \_template_assignments_pop: }

```

(End of definition for `\AssignTemplateKeys`. This function is documented on page 353.)

\SetKnownTemplateKeys A friendly wrapper, with some speed up for the common case of the third argument being empty.

\SetTemplateKeys

```

1201 \cs_new_protected:Npn \SetKnownTemplateKeys #1#2#3
1202 {
1203   \tl_if_empty:oTF {#3}
1204   {
1205     \tl_set_eq:NN \UnusedTemplateKeys \c_empty_tl
1206   }
1207   {
1208     \keys_set_known:noN { template / #1 / #2 } {#3} \UnusedTemplateKeys
1209   }
1210 }
1211 \cs_new_protected:Npn \SetTemplateKeys #1#2#3
1212 {
1213   \tl_if_empty:oF {#3}
1214   {
1215     \keys_set:no { template / #1 / #2 } {#3}
1216   }
1217 }
1218 \tl_new:N \UnusedTemplateKeys

```

(End of definition for `\SetKnownTemplateKeys` and `\SetTemplateKeys`. These functions are documented on page 354.)

```

1219 <latexrelease> \IncludeInRelease{0000/00/00}{lttemplates}%
1220 <latexrelease> {Prototype-document~commands}%
1221 <latexrelease>
1222 <latexrelease> \EndModuleRelease
1223 \ExplSyntaxOff
1224 </2ekernel | latexrelease>

```

We need to stop DocStrip treating @@ in a special way at this point.

```

1225 <@@=>

```


File 12

lalloc.dtx

1 Counters

This section deals with counter and other variable allocation.

`1 (*2kernel)`

The following are from plain T_EX:

`\z@` A zero dimen or number. It's more efficient to write `\parindent\z@` than `\parindent 0pt`.

`\@ne` The number 1.

`\m@ne` The number -1 .

`\tw@` The number 2.

`\sixt@@n` The number 16.

`\@m` The number 1000.

`\@MM` The number 20000.

`\@xxxii` The constant 32.

`2 \chardef\@xxxii=32`

(End of definition for \@xxxii.)

`\@Mi` Constants 10001–10004.

`\@Mii` `3 \mathchardef\@Mi=10001`

`\@Miii` `4 \mathchardef\@Mii=10002`

`\@Miv` `5 \mathchardef\@Miii=10003`

`6 \mathchardef\@Miv=10004`

(End of definition for \@Mi and others.)

`\@tempcnta` Scratch count registers used by L^AT_EX kernel commands.

`\@tempcntb` `7 \newcount\@tempcnta`

`8 \newcount\@tempcntb`

(End of definition for \@tempcnta and \@tempcntb.)

`\if@tempswa` General boolean switch used by L^AT_EX kernel commands.

`9 \newif\if@tempswa`

(End of definition for \if@tempswa.)

`\@tempdima` Scratch dimen registers used by L^AT_EX kernel commands.

`\@tempdimb` `10 \newdimen\@tempdima`

`\@tempdimc` `11 \newdimen\@tempdimb`

`12 \newdimen\@tempdimc`

(End of definition for \@tempdima, \@tempdimb, and \@tempdimc.)

`\@tempboxa` Scratch box register used by L^AT_EX kernel commands.
¹³ `\newbox\@tempboxa`
(End of definition for \@tempboxa.)

`\@tempskipa` Scratch skip registers used by L^AT_EX kernel commands.
`\@tempskipb` ¹⁴ `\newskip\@tempskipa`
¹⁵ `\newskip\@tempskipb`
(End of definition for \@tempskipa and \@tempskipb.)

`\@temptokena` Scratch token register used by L^AT_EX kernel commands.
¹⁶ `\newtoks\@temptokena`
(End of definition for \@temptokena.)

`\@flushglue` Glue used for `\right-` & `\leftskip = 0pt plus 1fil`
¹⁷ `\newskip\@flushglue \@flushglue = 0pt plus 1fil`
(End of definition for \@flushglue.)
¹⁸ `\</2ekernel`

File 13

ltnctrl.dtx

1 Program control structure

This section defines a number of control structure macros, such as while-loops and for-loops.

Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

```
1 <*2kernel>
2 \message{control,}
```

```
\@whilenum TEST \do {BODY}
\@whiledim TEST \do {BODY} : These implement the loop
    while TEST do BODY od
    where TEST is a TeX \ifnum or \ifdim test, respectively.
    They are optimized for the normal case of TEST initially false.
```

```
\@whilesw SWITCH \fi {BODY} : Implements the loop
    while SWITCH do BODY od
    Optimized for normal case of SWITCH initially false.
```

```
\@for NAME := LIST \do {BODY} : Assumes that LIST expands to A1,A2,
    ... ,An .
    Executes BODY n times, with NAME = Ai on the i-th iteration.
    Optimized for the normal case of n = 1. Works for n=0.
```

```
\@tfor NAME := LIST \do {BODY}
    if, before expansion, LIST = T1 ... Tn where each Ti is a
    token or {...}, then executes BODY n times, with NAME = Ti
    on the i-th iteration. Works for n=0.
```

NOTES: 1. These macros use no \@temp sequences.
2. These macros do not work if the body contains anything that looks syntactically to TeX like an improperly balanced \if \else \fi.

```
\@whilenum TEST \do {BODY} ==
BEGIN
  if TEST
  then BODY
    \@iwhilenum{TEST \relax BODY}
END
```

```
\@iwhilenum {TEST BODY} ==
BEGIN
  if TEST
  then BODY
```

```

        \@nextwhile = def(\@iwhilenum)
    else \@nextwhile = def(\@whilenoop)
    fi
    \@nextwhile {TEST BODY}
END

```

```

\@whilesw SWITCH \fi {BODY} ==
BEGIN
    if SWITCH
    then BODY
        \@iwhilesw {SWITCH BODY}\fi
    fi
END

```

```

\@iwhilesw {SWITCH BODY} \fi ==
BEGIN
    if SWITCH
    then BODY
        \@nextwhile = def(\@iwhilesw)
    else \@nextwhile = def(\@whileswnoop)
    fi
    \@nextwhile {SWITCH BODY} \fi
END

```

End of historical L^AT_EX 2.09 comments.

```

\@whilenoop
\@whilenum 3 \long\def\@whilenum#1\do #2{\ifnum #1\relax #2\relax\@iwhilenum{#1\relax
\@iwhilenum 4 #2\relax}\fi}
5 \long\def\@iwhilenum#1{\ifnum #1\expandafter\@iwhilenum
6 \else\expandafter\@gobble\fi{#1}}

```

(End of definition for \@whilenoop, \@whilenum, and \@iwhilenum.)

```

\@whiledim
\@iwhiledim 7 \long\def\@whiledim#1\do #2{\ifdim #1\relax#2\@iwhiledim{#1\relax#2}\fi}
8 \long\def\@iwhiledim#1{\ifdim #1\expandafter\@iwhiledim
9 \else\expandafter\@gobble\fi{#1}}

```

(End of definition for \@whiledim and \@iwhiledim.)

```

\@whileswnoop
\@whilesw 10 \long\def\@whilesw#1\fi#2{#1#2\@iwhilesw{#1#2}\fi\fi}
\@iwhilesw 11 \long\def\@iwhilesw#1\fi{#1\expandafter\@iwhilesw
12 \else\@gobbletwo\fi{#1}\fi}

```

(End of definition for `\@whilesnoop`, `\@whilesw`, and `\@iwhilesw`.)
 Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

```
\@for NAME := LIST \do {BODY} ==
  BEGIN \@forloop expand(LIST),\@nil,\@nil \@@ NAME {BODY} END
```

```
\@forloop CAR, CARCDR, CDRCDR \@@ NAME {BODY} ==
  BEGIN
    NAME = CAR
    if def(NAME) = def(\@nnil)
      else BODY;
      NAME = CARCDR
      if def(NAME) = def(\@nnil)
        else BODY
          \@iforloop CDRCDR \@@ NAME \do {BODY}
        fi
      fi
  END
```

```
\@iforloop CAR, CDR \@@ NAME {BODY} =
  NAME = CAR
  if def(NAME) = def(\@nnil)
    then \@nextwhile = def(\@fornoop)
    else BODY ;
      \@nextwhile = def(\@iforloop)
    fi
  \@nextwhile name cdr {body}
```

```
\@tfor NAME := LIST \do {BODY}
  = \@tforloop LIST \@nil \@@ NAME {BODY}
```

```
\@tforloop car cdr \@@ name {body} =
  name = car
  if def(name) = def(\@nnil)
    then \@nextwhile == \@fornoop
    else body ;
      \@nextwhile == \@forloop
    fi
  \@nextwhile name cdr {body}
```

End of historical L^AT_EX 2.09 comments.

`\@nnil`

```
13 \def\@nnil{\@nil}
```

(End of definition for `\@nnil`.)

`\@empty`

```
14 \def\@empty{}
```

(End of definition for `\@empty`.)

```

\@fornoop
15 \long\def\@fornoop#1\@@#2#3{}
(End of definition for \@fornoop.)

\@for
16 \long\def\@for#1:=#2\do#3{%
17   \expandafter\def\expandafter\@fortmp\expandafter{#2}%
18   \ifx\@fortmp\@empty \else
19     \expandafter\@forloop#2,\@nil,\@nil\@@#1{#3}\fi}
(End of definition for \@for.)

\@forloop
20 \long\def\@forloop#1,#2,#3\@@#4#5{\def#4{#1}\ifx #4\@nnil \else
21   #5\def#4{#2}\ifx #4\@nnil \else#5\@iforloop #3\@@#4{#5}\fi\fi}
(End of definition for \@forloop.)

\@iforloop
22 \long\def\@iforloop#1,#2\@@#3#4{\def#3{#1}\ifx #3\@nnil
23   \expandafter\@fornoop \else
24   #4\relax\expandafter\@iforloop\fi#2\@@#3{#4}}
(End of definition for \@iforloop.)

\@tfor
25 \def\@tfor#1:={\tf@r#1 }
26 \long\def\@tfor#1#2\do#3{\def\@fortmp{#2}\ifx\@fortmp\space\else
27   \@tforloop#2\@nil\@nil\@@#1{#3}\fi}
28 \long\def\@tforloop#1#2\@@#3#4{\def#3{#1}\ifx #3\@nnil
29   \expandafter\@fornoop \else
30   #4\relax\expandafter\@tforloop\fi#2\@@#3{#4}}
(End of definition for \@tfor.)

\@break@tfor Break out of a \@tfor loop. This should be called inside the scope of an \if. See
\@iffileonpath for an example.
31 \long\def\@break@tfor#1\@@#2#3{\fi\fi}
(End of definition for \@break@tfor.)

\@removeelement Removes an element from a comma-separated list and puts it into a control sequence,
called as \@removeelement{<element>}{<list>}{<cs>}. Due to the implementation
method the <element> is not allowed to contain braces.
32 \def\@removeelement#1#2#3{%
33   \def\reserved@a##1,#1,#2\reserved@a{##1,##2\reserved@b}%
34   \def\reserved@b##1,\reserved@b##2\reserved@b{%
35     \ifx,##1\@empty\else##1\fi}%
36   \edef#3{%
37     \expandafter\reserved@b\reserved@a,#2,\reserved@b,#1,\reserved@a}}
(End of definition for \@removeelement.)
38 </2kernel>

```

File 14

lterror.dtx

1 Error handling and tracing

This section defines L^AT_EX's error commands.

```
1 <*2kernel)
```

The ‘2kernel’ code ensures that a `\usepackage{autoerr}` is essentially ignored if a ‘full’ format is being used that has the error messages already in the format.

These days we don't support autoloading approach any longer, but this part bit is kept in case it is used in old documents.

```
2 \expandafter\let\csname ver@autoerr.sty\endcsname\fmtversion
```

1.1 General commands

\MessageBreak This command prints a new-line inside a message, followed by a continuation line begun with `\@msg@continuation`. Normally it is defined to be `\relax`, but inside messages, it is let to `\@message@break`.

```
3 \let\MessageBreak\relax
```

(End of definition for \MessageBreak.)

\GenericInfo This takes two arguments: a continuation and a message, and sends the result to the log file.

```
4 \DeclareRobustCommand{\GenericInfo}[2]{%
5   \begingroup
6     \def\MessageBreak{^^J#1}%
7     \set@display@protect
8     \immediate\write\m@ne{#2\on@line.}%
9   \endgroup
10 }
```

(End of definition for \GenericInfo.)

\GenericWarning This takes two arguments: a continuation and a message, and sends the result to the screen.

```
11 \DeclareRobustCommand{\GenericWarning}[2]{%
12   \begingroup
13     \def\MessageBreak{^^J#1}%
14     \set@display@protect
15     \immediate\write\@unused{^^J#2\on@line.^^J}%
16   \endgroup
17 }
```

(End of definition for \GenericWarning.)

`\GenericError` This macro takes four arguments: a continuation, an error message, where to go for further information, and the help information. It displays the error message, and sets the error help (the result of typing `h` to the prompt), and does a horrible hack to turn the last context line (which by default is the only context line) into just three dots. This could be made more efficient.

```

18 \bgroup
19 \lccode'\@=' \ %
20 \lccode'\~=' \ %
21 \lccode'\}= \ %
22 \lccode'\{=' \ %
23 \lccode'\T=' \T%
24 \lccode'\H=' \H%
25 \catcode'\ =11\relax%
26 \lowercase{%
27 \egroup%
```

Unfortunately \TeX versions older than 3.141 have a bug which means that `^^J` does not force a linebreak in `\message` and `\errmessage` commands. So for these old \TeX 's we use `\typeout` to produce the message, and then have an empty `\errmessage` command. This causes an extra line of the form

.

To appear on the terminal, but if you do not like it, you can always upgrade your \TeX ! In order for your format to use this version, you must define the macro `\@TeXversion` to be the version number, e.g., 3.14 of the underlying \TeX . See the comments in `ltdircheck.dtx`.

```

28 \dimen@ifx\@TeXversion\@undefined4\else\@TeXversion\fi\p@%
29 \ifdim\dimen@>3.14\p@%
    First the 'standard case'.
30 \DeclareRobustCommand{\GenericError}[4]{%
31 \begingroup%
32 \immediate\write\@unused{}%
33 \def\MessageBreak{^^J}%
34 \set@display@protect%
35 \edef%
36 % %<-----do not delete this space!----->%
37 \@err@
38 {{#4}}%
39 \errhelp
40 % %<-----do not delete this space!----->%
41 \@err@
42 \let
43 % %<-----do not delete this space!----->%
44 \@err@
45 \@empty
46 \def\MessageBreak{^^J#1}%
47 \def~{\errmessage{%
48 #2.^^J^^J%
49 #3^^J%
50 Type H <return> for immediate help%
51 % %<-----do not delete this space!----->%
52 \@err@
%
```



```

53 }}%
54 ~%
55 \endgroup}%
56 \else%
    Secondly the version for old TEX's.
57 \DeclareRobustCommand{\GenericError}[4]{%
58 \begingroup%
59 \immediate\write\@unused{}%
60 \def\MessageBreak{^^J}%
61 \set@display@protect%
62 \edef%
63 % %<-----do not delete this space!----->%
64 \@err@ %
65 {#{4}}%
66 \errhelp
67 % %<-----do not delete this space!----->%
68 \@err@ %
69 \let
70 % %<-----do not delete this space!----->%
71 \@err@ %
72 \errmessage
73 \def\MessageBreak{^^J#1}%
74 \def~{\typeout{! %
75 #2.^^J^^J%
76 #3^^J%
77 Type H <return> for immediate help.}%
78 % %<-----do not delete this space!----->%
79 \@err@ %
80 {}}%
81 ~%
82 \endgroup}%
83 \fi}%

```

(End of definition for \GenericError.)

<pre> \PackageError \PackageWarning \PackageWarningNoLine \PackageInfo \ClassError \ClassWarning \ClassWarningNoLine \ClassInfo </pre>	<p>These commands are intended for use by package and class writers, to give information to authors. The syntax is:</p> <pre> \PackageError{<package>}{<error>}{<help>} \PackageWarning{<package>}{<warning>} \PackageWarningNoLine{<package>}{<warning>} \PackageInfo{<package>}{<info>} </pre>
--	--

and similarly for classes. The **Error** commands print the `<error>` message, and present the interactive prompt; if the author types `h`, then the `<help>` information is displayed. The **Warning** commands produce a warning but do not present the interactive prompt. The **WarningNoLine** commands do the same, but don't print the input line number. The **Info** commands write the message to the log file. Within the messages, the command `\MessageBreak` can be used to break a line, `\protect` can be used to protect command names, and `\space` is a space, for example:

```

\newcommand{\foo}{F00}
\PackageWarning{ethel}{%
  Your hovercraft is full of eels,\MessageBreak
  and \protect\foo\space is \foo}

```

produces:

```

Package ethel warning: Your hovercraft is full of eels,
(ethel)                and \foo is F00 on input line 54.

```

```

84 \gdef\PackageError#1#2#3{%
85   \GenericError{%
86     (#1)\@spaces\@spaces\@spaces\@spaces
87   }{%
88     Package #1 Error: #2%
89   }{%
90     See the #1 package documentation for explanation.%
91   }{#3}%
92 }
93 \def\PackageWarning#1#2{%
94   \GenericWarning{%
95     (#1)\@spaces\@spaces\@spaces\@spaces
96   }{%
97     Package #1 Warning: #2%
98   }%
99 }
100 \def\PackageWarningNoLine#1#2{%
101   \PackageWarning{#1}{#2\@gobble}%
102 }
103 \def\PackageInfo#1#2{%
104   \GenericInfo{%
105     (#1) \@spaces\@spaces\@spaces
106   }{%
107     Package #1 Info: #2%
108   }%
109 }
110 \gdef\ClassError#1#2#3{%
111   \GenericError{%
112     (#1) \space\@spaces\@spaces\@spaces
113   }{%
114     Class #1 Error: #2%
115   }{%
116     See the #1 class documentation for explanation.%
117   }{#3}%
118 }
119 \def\ClassWarning#1#2{%
120   \GenericWarning{%
121     (#1) \space\@spaces\@spaces\@spaces
122   }{%
123     Class #1 Warning: #2%
124   }%
125 }
126 \def\ClassWarningNoLine#1#2{%

```

```

127 \ClassWarning{#1}{#2@gobble}%
128 }
129 \def\ClassInfo#1#2{%
130 \GenericInfo{%
131 (#1) \space\space\@spaces\@spaces
132 }{%
133 Class #1 Info: #2%
134 }%
135 }

```

(End of definition for \PackageError and others.)

```

\ClassNote
\ClassNoteNoLine 136 </2ekernel>
\PackageNote 137 <*2ekernel | latexrelease>
\PackageNoteNoLine 138 <latexrelease> \IncludeInRelease{2021/11/15}%
139 <latexrelease> {\ClassNote}{Notes for classes/packages}%
140 \def\ClassNote#1#2{%
141 \GenericWarning{%
142 (#1) \space\space\@spaces\@spaces
143 }{%
144 Class #1 Info: #2%
145 }%
146 }
147 \def\ClassNoteNoLine#1#2{\ClassNote{#1}{#2@gobble}}
148 \def\PackageNote#1#2{%
149 \GenericWarning{%
150 (#1) \@spaces\@spaces\@spaces
151 }{%
152 Package #1 Info: #2%
153 }%
154 }
155 \def\PackageNoteNoLine#1#2{\PackageNote{#1}{#2@gobble}}
156 </2ekernel | latexrelease>
157 <latexrelease> \EndIncludeInRelease

```

We don't roll back, because if this code is used by packages then most often they will not have rollback code implemented, so they would immediately break even if they otherwise would work fine.

```

158 <latexrelease> \IncludeInRelease{0000/00/00}%
159 <latexrelease> {\ClassNote}{Notes for classes/packages}%
160 <latexrelease>
161 <latexrelease> \EndIncludeInRelease
162 <*2ekernel>

```

(End of definition for \ClassNote and others.)

```

\@latex@error Errors and other info, for use in the LATEX core.
\@latex@warning 163 \gdef\@latex@error#1#2{%
\@latex@warning@no@line 164 \GenericError{%
\@latex@info 165 \space\space\space\@spaces\@spaces\@spaces
\@latex@info@no@line 166 }{%
167 LaTeX Error: #1%
168 }%

```

```

169     See the LaTeX manual or LaTeX Companion for explanation.%
170   }{#2}%
171 }
172 \def\@latex@warning#1{%
173   \GenericWarning{%
174     \space\space\space\@spaces\@spaces\@spaces
175   }{%
176     LaTeX Warning: #1%
177   }%
178 }
179 \def\@latex@warning@no@line#1{%
180   \@latex@warning{#1\@gobble}}
181 \def\@latex@info#1{%
182   \GenericInfo{%
183     \@spaces\@spaces\@spaces
184   }{%
185     LaTeX Info: #1%
186   }%
187 }
188 \def\@latex@info@no@line#1{%
189   \@latex@info{#1\@gobble}}

```

\@font@warning and \@font@info are defined later since they have to be redefined by the tracefnt package.

```

def\@font@warning#1{%
  \GenericWarning{%
    {(font)\@spaces\@spaces}%
    {Font Warning: #1}%
  }
def\@font@info#1{%
  \GenericInfo{%
    (font)\space\@spaces
  }{%
    Font Info: #1%
  }%
}

```

(End of definition for \@latex@error and others.)

```

\@latex@note These are “info” messages that display on the terminal not just in the transcript.
\@latex@note@no@line
190 </2ekernel>
191 <*2ekernel | latexrelease>
192 <latexrelease> \IncludeInRelease{2021/11/15}%
193 <latexrelease>           {\@latex@note}{Display notes}%
194 \def\@latex@note#1{%
195   \GenericWarning{%
196     \@spaces\@spaces\@spaces
197   }{%
198     LaTeX Info: #1%
199   }%
200 }

```

```

201 \def\@latex@note@no@line#1{%
202   \@latex@note{#1\@gobble}}

```

We don't make them undefined but rather point to `\@latex@info` because that's what they replace. This way we can change `\@latex@info` elsewhere without the need to further rollback sections.

```

203 </2ekernel | latexrelease>
204 <latexrelease>\EndIncludeInRelease
205 <latexrelease>\IncludeInRelease{0000/00/00}%
206 <latexrelease>           {\@latex@note}{Display notes}%
207 <latexrelease>
208 <latexrelease>\let\@latex@note\@latex@info
209 <latexrelease>\let\@latex@note@no@line\@latex@info@no@line
210 <latexrelease>\EndIncludeInRelease
211 <*2ekernel>

```

(End of definition for \@latex@note and \@latex@note@no@line.)

`\c@errorcontextlines` `\errorcontextlines` as a L^AT_EX counter, so that it may be manipulated with `\setcounter` (once it is defined :-)

```

212 \let\c@errorcontextlines\errorcontextlines
213 \c@errorcontextlines=-1

```

(End of definition for \c@errorcontextlines.)

`\on@line` The message ‘ on input line *n*’.

```

214 \def\on@line{ on input line \the\inputlineno}

```

(End of definition for \on@line.)

`\@warning` Older L^AT_EX messages. For the moment, these `\let` to the new message commands. They may be changed later, once only obsolete packages and classes contain them.

```

\@warning
\@warning
\@latexerr
215 \let\@warning\@latex@warning
216 \let\@@warning\@latex@warning@no@line
217 \global\let\@latexerr\@latex@error

```

(End of definition for \@warning, @@warning, and \@latexerr.)

`\@spaces` Four spaces.

```

218 \def\@spaces{\space\space\space\space}

```

(End of definition for \@spaces.)

1.2 Specific errors

`\@eha` The more common error help messages.

```

\@ehb
219 \gdef\@eha{%
\@ehc
220   Your command was ignored.\MessageBreak
\@ehd
221   Type \space I <command> <return> \space to replace it %
222   with another command,\MessageBreak
223   or \space <return> \space to continue without it.}
224 \gdef\@ehb{%
225   You've lost some text. \space \@ehc}
226 \gdef\@ehc{%
227   Try typing \space <return> %

```

```

228 \space to proceed.\MessageBreak
229 If that doesn't work, type \space X <return> \space to quit.}
230 \gdef\@ehd{%
231 You're in trouble here. \space\@ehc}

```

(End of definition for \@eha and others.)

\@notdefinable Error message generated in \@ifdefinable from calls to one of the commands \newcommand, \newlength or \newtheorem specifying an already-defined command name or one that begins \end....

```

232 \gdef\@notdefinable{%
233 \@latex@error{%
234 Command \@backslashchar\reserved@a\space
235 already defined.\MessageBreak
236 Or name \@backslashchar\@qend... illegal,
237 see p.192 of the manual}\@eha}

```

(End of definition for \@notdefinable.)

\@nolnerr Generated by \newline and \\ when called in vertical mode.

```

238 \gdef\@nolnerr{%
239 \@latex@error{There's no line here to end}\@eha}

```

(End of definition for \@nolnerr.)

\@nocounterr Generated by \setcounter, \addtocounter or \newcounter if applied to an undefined counter <cnt>.

\@nocnterr Obsolete error message generated in L^AT_EX 2.09 by \setcounter, \addtocounter or \newcounter for undefined counter. DO NOT use for L^AT_EX 2_ε it MIGHT vanish! Use \@nocounterr{<cnt>} instead.

```

240 \gdef\@nocounterr#1{%
241 \@latex@error{No counter '#1' defined}\@eha}
242 \gdef\@nocnterr{\@nocounterr?}

```

(End of definition for \@nocounterr and \@nocnterr.)

\@ctrerr Called when trying to print the value of a counter numbered by letters that's greater than 26.

```

243 \gdef\@ctrerr{%
244 \@latex@error{Counter too large}\@ehb}

```

(End of definition for \@ctrerr.)

\@nodocument Error produced if paragraphs are typeset in the preamble.

```

245 \gdef\@nodocument{%
246 \@latex@error{Missing \protect\begin{document}}\@ehd}

```

(End of definition for \@nodocument.)

`\@badend` Called by `\end` that doesn't match its `\begin`. RmS 1992/08/24: added code to `\@badend` to display position of non-matching `\begin`. FMi 1993/01/14: missing space added.

The environment name has to literally match, i.e., what is stored in `\@currenvir` (after one expansion) must match what is passed to `\end` (without expansion). If not we complain. Not the absolute best solution but at least it avoids getting `\begin{foo}` ended by `\end{foo}` which was possible in the past.

```
247 \gdef\@badend#1{%
248   \latex@error{\protect\begin
249     {\detokenize\expandafter{\@currenvir}}\@currenvline
250     \space ended by \protect\end{\detokenize{#1}}}\@eha}
```

(End of definition for \@badend.)

`\@badmath` Called by `\[`, `\]`, `\(` or `\)` when used in wrong mode.

```
251 \gdef\@badmath{%
252   \latex@error{Bad math environment delimiter}\@eha}
```

(End of definition for \@badmath.)

`\@toodeep` Called by a list environment nested more than six levels deep, or an `enumerate` or `itemize` nested more than four levels.

```
253 \gdef\@toodeep{%
254   \latex@error{Too deeply nested}\@ehd}
```

(End of definition for \@toodeep.)

`\@badpoptabs` Called by `\endtabbing` when not enough `\poptabs` have occurred, or by `\poptabs` when too many have occurred.

```
255 \gdef\@badpoptabs{%
256   \latex@error{\protect\pushtabs\space and \protect\poptabs
257     \space don't match}\@ehd}
```

(End of definition for \@badpoptabs.)

`\@badtab` Called by `\>`, `\+`, `\-` or `\<` when stepping to an undefined tab.

```
258 \gdef\@badtab{%
259   \latex@error{Undefined tab position}\@ehd}
```

(End of definition for \@badtab.)

`\@preamerr` This error is special: it appears in places where we normally have to `\protect` expansions. However, to prevent a protection of the error message itself (which would result in the message getting printed not issued on the terminal) we need to locally reset `\protect` to `\relax`.

```
260 \gdef\@preamerr#1{%
261   \begingroup
262     \let\protect\relax
263     \latex@error{\ifcase #1 Illegal character\or
264       Missing @-exp\or Missing p-arg\fi\space
265       in array arg}\@ehd
266   \endgroup}
```

(End of definition for \@preamerr.)

`\@badlinearg` Occurs in `\line` and `\vector` command when a bad slope argument is encountered.

```

267 \gdef\@badlinearg{%
268   \latex@error{%
269     Bad \protect\line\space or \protect\vector
270     \space argument}\@ehb}

```

(End of definition for \@badlinearg.)

`\@LRmoderr` A command is not allowed in restricted horizontal mode, i.e., in LR-mode in L^AT_EX terminology.

```

271 \gdef\@LRmoderr{%
272   \latex@error{Not allowed in LR mode}\@ehb}

```

(End of definition for \@LRmoderr.)

`\@parmoderr` Occurs in a float environment or a `\marginpar` when encountered in inner vertical mode.

```

273 \gdef\@parmoderr{%
274   \latex@error{Not in outer par mode}\@ehb}

```

(End of definition for \@parmoderr.)

`\@fltovf` Occurs in float environment or `\marginpar` when there are no more free boxes for storing floats.

```

275 \gdef\@fltovf{%
276   \latex@error{Too many unprocessed floats}\@ehb}

```

(End of definition for \@fltovf.)

`\@latexbug` Occurs in output routine. This is bad news.

```

277 \gdef\@latexbug{%
278   \latex@error{This may be a LaTeX bug}{Call for help}}

```

(End of definition for \@latexbug.)

`\@badcrerr` This error was removed and replaced by `\@nolnerr`.

```

279 %\def\@badcrerr {\latex@error{Bad use of \protect\\}\@ehc}

```

(End of definition for \@badcrerr.)

`\@noitemerr` `\addvspace` or `\addpenalty` was called when not in vmode. Probably caused by a missing `\item`.

```

280 \gdef\@noitemerr{%
281   \latex@error{Something's wrong--perhaps a missing %
282     \protect\item}\@ehc}

```

(End of definition for \@noitemerr.)

`\@notprerr` A command that can be used only in the preamble appears after the command `\begin{document}`.

```

283 \gdef\@notprerr{%
284   \latex@error{Can be used only in preamble}\@eha}

```

(End of definition for \@notprerr.)

`\@inmatherr` Issued by commands that don't work correctly within math (like `\item`). There is no real error recovery happening, e.g., the user might get additional errors afterwards.

```
285 \gdef\@inmatherr#1{%  
286   \relax  
287   \ifmmode  
288     \@latex@error{Command \protect#1 invalid in math mode}\@ehc  
289   \fi}
```

(End of definition for \@inmatherr.)

`\@invalidchar` An error for use with invalid characters. This is commented out, since we decided to use catcode 15 instead.

```
290 %\def\@invalidchar{\@latex@error{Invalid character in input}\@ehc}
```

(End of definition for \@invalidchar.)

As well as the above error commands some error messages are directly coded to save space. The messages already present in L^AT_EX2.09 include:

Environment --- undefined

Issued by `\begin` for undefined environment.

Tab overflow

Occurs in `\=` when maximum number of tabs exceeded.

\< in mid line

Occurs in `\<` when it appears in middle of line.

Float(s) lost

In output routine, caused by a float environment or `\marginpar` occurring in inner vertical mode.

1.3 Tracing

The `trace` package implements the commands `\traceon` and `\traceoff` that work similar to `\tracingall` but skip certain code blocks that produce a lot of tracing output being of no interest during debugging (for example loading a font). Code blocks that should be hidden during tracing need to be surrounded by the macros `\conditionally@traceoff` and `\contionally@traceon`.

For the kernel code the `trace` package then redefines a number of macros to include this tracing support.

However, in order to allow any macro package to react to `\traceon` we also provide dummy definitions for the two commands in the kernel so that they can be used by external packages without the need to distinguish between `trace` being loaded or not.

`\conditionally@traceon` These are only dummy definitions. For details see the `trace` package.

```
\conditionally@traceoff 291 \let\conditionally@traceon\@empty  
292 \let\conditionally@traceoff\@empty
```

(End of definition for \conditionally@traceon and \conditionally@traceoff.)

```
293 </2kernel>
```

File 15

ltpar.dtx

1 Paragraphs

This section of the kernel declares the commands used to set `\par` and `\everypar` whenever their function needs to be changed for a long time.

This file here describes the interfaces that have been in the kernel forever, used to implement the scenarios described below. They remain valid but are now augmented in the next file (`ltpara.dtx`) to add hooks to paragraphs. At some point we will consolidate the two files further.

There are two situations in which `\par` may be changed:

- Long-term changes, in which the new value is to remain in effect until the current environment is left. The environments that change `\par` in this way are the following:
 - All list environments (`itemize`, `quote`, etc.)
 - Environments that turn `\par` into a noop: `tabbing`, `array` and `tabular`.
- Temporary changes, in which `\par` is restored to its previous value the next time it is executed. The following are all such uses.
 - `\end` when preceded by `\@endparenv`, which is called by `\endtrivlist`
 - The mechanism for avoiding page breaks and getting the spacing right after section heads.

1.1 Implementation

`\@setpar` To permit the proper interaction of these two situations, long-term changes are made by the `\@setpar{<VAL>}` command. Its function is:

To set `\par`. It `\def`'s `\par` and `\@par` to `<VAL>`.

`\@restorepar` Short-term changes are made by the usual `\def\par` commands. The original values are restored after a short-term change by the `\@restorepar` commands.

`\@@par` `\@@par` always is defined to be the original \TeX `\par`.

`\everypar` `\everypar` is changed only for the short term. Whenever `\everypar` is set non-null, it should restore itself to null when executed.

The following commands change `\everypar` in this way:

- `\item`
- `\end` when preceded by `\@endparenv`, which is called by `endtrivlist`
- `\minipage`

When dealing with `\par` and `\everypar` remember the following two warnings:

1. Commands that make short-term changes to `\par` and `\everypar` must take account of the possibility that the new commands and the ones that do the restoration may be executed inside a group. In particular, `\everypar` is executed inside a group whenever a new paragraph begins with a left brace. The `\everypar` command that restores its definition should be local to the current group (in case the command is inside a minipage used inside someplace where `\everypar` has been redefined). Thus, if `\everypar` is redefined to do an `\everypar{}` it could take several executions of `\everypar` before the restoration “holds”. This usually causes no problem. However, to prevent the extra executions from doing harm, use a global switch to keep anything harmful in the new `\everypar` from being done twice.
2. Commands that change `\everypar` should remember that `\everypar` might be supposed to set the following switches false:

- `@nobreak`
- `@minipage`

they should do the setting if necessary.

```

1 <*2kernel>
2 \message{par,}

```

```

\@setpar  Initiate a long-term change to \par.
\@par    3 \def\@setpar#1{\def\par{#1}\def\@par{#1}}

```

The default definition of `\@par` will ensure that if `\@restorepar` defines `\par` to execute `\@par` it will redefine itself to the primitive `\@@par` after one iteration.

```

4 \def\@par{\let\par\@@par\par}

```

(End of definition for \@setpar and \@par.)

```

\@restorepar  Restore from a short-term change to \par.

```

```

5 \def\@restorepar{\def\par{\@par}}
6 </2kernel>

```

(End of definition for \@restorepar.)

File 16

ltpara.dtx

Abstract

This code defines four special kernel hooks to support paragraph tagging as well as four public hooks which can be occasionally useful.

1 Introduction

The building of paragraphs in the \TeX engine(s) has a number of peculiarities that makes it on one hand fairly flexible but on the other hand somewhat awkward to control or reliably to extend. Thus to better understand the code below we start with a brief introduction of the mechanism; for more details refer to the \TeX book [?, chap. 14] (for the full truth you may even have to study the program code).

1.1 The default processing done by the engine

\TeX automatically starts building a paragraph when it is currently in vertical mode and encounters anything that can only live in horizontal mode. Most often this is a character, but there are also many commands that can be used only in horizontal mode. If any of them is encountered, \TeX will immediately back up (i.e., the character or command is read later again), adds a `\parskip` glue to the current vertical list unless the list is empty, switches to horizontal mode, starts its special “start of paragraph processing” and only then rereads the character or command that caused the mode change.²⁴

This “start of paragraph processing” first adds an empty box at the start of the horizontal list of width `\parindent` (which represents the paragraph indentation) unless the paragraph was started with `\noindent` in which case no such box is added²⁵. It then reads and processes all tokens stored in the special engine token register `\everypar`. After that it reads and processes whatever has caused the paragraph to start.

Thus out of the box, \TeX offers the possibility to put some special code into `\everypar` to gain control at (more or less) the start of the paragraph. For example, in LaTeX and a number of packages, special code like the following is sometimes used:

```
\everypar{\setbox\z@\lastbox}\everypar{} ...}
```

This removes the paragraph indentation box again (that was already placed by \TeX), then resets `\everypar` so that it doesn't do anything on the next paragraph start and then does whatever it wants to do, e.g., in an `\item` of a list it will typeset the label in front of the paragraph text. However, there is only one such `\everypar` token register and if different packages and/or the kernel all attempt to add their own code here, coordination is very difficult if not impossible.

The process when the paragraph ends has different mechanisms and interfaces. A paragraph ends when the engine primitive `\par` is called while \TeX is in unrestricted horizontal mode, i.e., is building a paragraph. At other times this primitive does nothing or generates as an error depending on the mode \TeX is in, e.g., the `\par` in `\hbox{a\par b}` is ignored, but `$a\par b$` would complain.

²⁴Already not quite true: the command `\noindent` starts the paragraph but influences the special processing by suppressing the paragraph indentation box normally inserted by it.

²⁵That's a bit different from placing a zero-sized box!

If this primitive ends the paragraph it does some special “end of horizontal list” processing, then calls T_EX’s paragraph builder; this breaks the horizontal list into lines and then these lines are added as boxes to the enclosing vertical list and T_EX returns to vertical mode.

This `\par` command can be given explicitly, but there are also situations in which T_EX is generating it on the fly. Most often this happens when T_EX encounters a blank line which is automatically changed to a `\par` command which is then executed. The other possibility is that T_EX encounters a command which is incompatible with horizontal processing, e.g., `\vskip` (a request for adding vertical space). In such cases it silently backs up, and inserts a `\par` in the hope that this gets it out of horizontal mode and makes the vertical command acceptable.

The important point to note here is that T_EX really inserts the command with the name `\par`, which can be redefined. Thus, it may not have its original “primitive” meaning and therefore may not end the horizontal list and call the paragraph builder. This approach offers some flexibility but also allows you to easily produce a T_EX document that loops forever, for example, the simple line

```
A \let\par\relax \vskip
```

will start a horizontal list at A, redefines `\par`, then sees `\vskip` and inserts `\par` to end the paragraph. But this now only runs `\relax` so nothing changes and `\vskip` is read again, issues a `\par` which In short, it only takes a plain T_EX document with five tokens to run forever (since no memory is consumed and therefore eventually exhausted).

There is no way other than changing `\par` to gain control at the end of a paragraph, i.e., there is no token list like `\everypar` that is inserted. Hence the only way to change the default behavior is to modify the action that `\par` executes, with similar issues as outlined before: different processes need to ensure that they do not overwrite their modifications or worse, think that the `\par` in front of them is the engine primitive while in fact it has already been changed by other code.

To make matters slightly worse there are a few places where T_EX handles the situation differently (most likely for speed reasons back when computers were much slower). If T_EX finds itself in unrestricted horizontal mode at the end of building a vertical box (for an `\insert`, `\vadjust` or executing the output routine code), it will finish the horizontal list not by issuing a `\par` command (which would be consistent with all other places) but by simply executing the primitive meaning of `\par`, regardless of the actual definition that `\par` has at the time.

Thus, if you have carefully crafted a redefined `\par` to execute some special actions at the end of a paragraph and you write something like

```
\vbox{Some paragraph ... text.}
```

you will find that your code does not get run for the last paragraph in that box. L^AT_EX avoids this problem, by making sure that its boxes (such as `\parbox` or the `minipage` environment, etc.) all internally add an explicit `\par` at the end so that such code is run and T_EX finds itself in vertical mode already without the need to start up the paragraph builder internally. But, of course, this only works for boxes under direct control of the L^AT_EX kernel; if some package uses low-level `\vboxes` without adding this precaution the T_EX optimization kicks in and no special `\par` code is executed.

And there is another optimization that is painful: if a paragraph is interrupted by a mathematical display, e.g., `\[. . .\]` in L^AT_EX or `$$. . . $$` in plain T_EX, then T_EX will resume horizontal mode afterward, i.e., it will start to build a new horizontal list

without inserting an indentation box or `\everypar` at that point. However, if that list immediately ends with an explicit or implicit `\par` then `TEX` will simply throw away this “null” paragraph and not do its usual “end of horizontal list” processing, so this special case also needs to be accounted for when introducing any extended processing.

2 The new mechanism implemented for `LATEX`

To improve the situation (and also to support automatic tagging of PDF documents) we now offer public as well as private hooks at the start and end of the paragraph processing. The public hooks can be used by packages (or by the user in the preamble or within the document) and using the hook mechanisms it is possible to reorder or arrange code from different packages in such a way that these can safely coexist.

To make that happen we have to make use of the basic functionality that is offered by `TEX`, e.g., we install special code inside `\everypar` to provide hooks at the beginning and we redefine `\par` to do some special processing when appropriate to install hooks at the end of the paragraph.

In order to make this work, we have to ensure that package use of `\everypar` is not overwriting our code. This is done through a trick: we basically hide the real `\everypar` from the packages and offer them a new token register (with the same name). So if they install their own code it doesn’t overwrite ours. Our code then inserts the new `\everypar` at the right place inside the process so that it looks as if it was the primitive `\everypar`.²⁶

At the end of the paragraph it would be great if we could use a similar trick. However, due to the fact that `TEX` inserts the token `\par` (that doesn’t have a defined meaning) we can’t hide “the real thingTM” and offer the package an indistinguishable alternate.

Fortunately, `LATEX` has already redefined `\par` for its own purposes. As a result there aren’t many packages that attempt to change `\par`, because without a lot of extra care that would fail miserably. But the bottom line is that, if you load a package that alters `\par` then the end of paragraph hooks are most likely not executing while that redefinition is active.²⁷

²⁶Ideally, `\everypar` wouldn’t be used at all by packages and instead they would simply write their code into the hooks now offered by the kernel. However, while this is the longterm goal and clearly an improvement (because then the packages do no longer need to worry about getting their code overwritten or needing to account for already existing code in `\everypar`), this will not happen overnight. For that reason support for this legacy method is retained.

²⁷Similarly to the `\everypar` situation, the remedy is that such packages stop doing this and instead add their alterations into the paragraph hooks now provided.

2.1 The provided hooks

<code>para/before</code>	The following four public hooks are defined and executed for each paragraph:
<code>para/begin</code>	
<code>para/end</code>	para/before This hook is executed after the kernel hook <code>\@kernel@before@para@before</code> (discussed below) in vertical mode immediately after \TeX has contributed <code>\parskip</code> to the vertical list and before the actual paragraph processing in horizontal mode starts. This hook should either not produce any typeset material or add only vertical material. If it starts a paragraph an error is generated. The reason is that we are in the starting process of processing a paragraph and so this would lead to endless recursion. ²⁸
<code>para/after</code>	para/begin This hook is executed after the kernel hook <code>\@kernel@before@para@begin</code> (discussed below) in horizontal mode immediately before the indentation box is placed (if there is any, i.e., if the paragraph hasn't been started with <code>\noindent</code>). The indentation box to be typeset is available to the hook as <code>\IndentBox</code> and its automatic placement (after the hook is executed) can be prevented through <code>\OmitIndent</code> . More precisely <code>\OmitIndent</code> voids the box. The indentation box is then typeset directly after the hook execution by something equivalent to <code>\box\IndentBox</code> followed by the current content of the token register <code>\everypar</code> that it is available to the kernel or to packages (that run some legacy code). One has to be careful not to add any code to the hook that starts its own paragraph (e.g., by adding a <code>\parbox</code> or a <code>\marginpar</code> inside) because that would call the hook inside again (as a new paragraph is started there) and thus lead to an endless recursion ending only after exhausting the available memory. This can only be done by making sure that is not executed for the inner paragraphs (or at least not recursively forever). para/end This hook is executed at the end of a paragraph when \TeX is ready to return to vertical mode and after it has removed the last horizontal glue (but not any kerns) placed on the horizontal list. The code is still executed in horizontal mode so it is possible to add further horizontal material at this point, but it should not alter the mode (even a temporary exit from horizontal mode would create chaos—any attempt will cause an error message)! After the hook has ended the kernel hook <code>\@kernel@after@para@end</code> is executed and then \TeX returns to vertical mode. The hook is offered as public hook, but because of the requirement to stay within horizontal mode one needs to be careful in what is placed into the hook. ²⁹ This hook is implemented as a reversed hook. para/after This hook is executed directly after \TeX has returned to vertical mode and after any material that migrated out of the horizontal list (e.g., from a <code>\vadjust</code>) has processed.

²⁸One could allow it but only if the newly started paragraph is processed without any hooks. Further more correct spacing would be a bit of a nightmare so for now this is forbidden.

²⁹Maybe we should guard against that, but it would be rather tricky to implement as mode changes can happen across group boundaries so one would need to keep a private stack just for that. Well, something to ponder.

This hook should either not produce any typeset material or add only vertical material. However, for this hook starting a new paragraph is not a disaster so that it isn't prevented.

This hook is implemented as a reversed hook.

Once that hook code has been processed the kernel hook `\@kernel@after@para@after` is executed as the final action of the paragraph processing.

```
\@kernel@before@para@before
\@kernel@after@para@after
\@kernel@before@para@begin
\@kernel@after@para@end
```

As already mentioned above there are also four kernel hooks that are executed at the start and end of the processing.

`\@kernel@before@para@before` For future extensions, not currently used by the kernel.

`\@kernel@after@para@after` For future extensions, not currently used by the kernel.

`\@kernel@before@para@begin` Used by the kernel to implement tagging. This hook is executed at the very beginning of a paragraph after `TEX` has switched to horizontal mode but before any indentation box got added or any `\everypar` was run.

It should not generate typeset material that could alter the position. Note that it should never leave hmode, otherwise you will end with a loop! We could guard against this, but since it is an internal kernel hook that shouldn't be touched this isn't checked.

`\@kernel@after@para@end` Used by the kernel to implement tagging. It is executed directly after the public `para/end` hook. After it there is a quick check that we are still in horizontal mode, i.e., that the public hook has not mistakenly ended horizontal mode prematurely (this is an incomplete check just testing the mode and could perhaps be improved (at the cost of speed)).

2.2 Altered and newly provided commands

```
\par
\endgraf
\para_end:
```

An explicit request for ending a paragraph is provided in plain `TEX` under the name `\endgraf`, which simply uses the primitive meaning (regardless of what `\par` may have as its current definition). In `LATEX` `\endgraf` (with that behavior) was originally also available.

With the new paragraph handling in `LATEX`, ending a paragraph means a bit more than just calling the engine's paragraph builder: the process also has to add any hook code for the end of a paragraph. Thus `\endgraf` was changed to provide this additional functionality (along with `\par` remaining subject to its current meaning).

The `expl3` name for this functionality is `\para_end:`.

Note: *The next two commands are still under discussion and may slightly change their semantics (as described in the document) and/or their names between now and the 2021 Spring release!*

<code>\OmitIndent</code>	Inside the <code>para/begin</code> hook one can use this command to suppress the indentation box at the start of the paragraph. (Technically it is possible to use this command outside the hook as well, but this should not be relied upon.) The box itself remains available for use.
<code>\para_omit_indent:</code>	

The expl3 name for the function is `\para_omit_indent:`.

<code>\IndentBox</code>	The box register holding the indentation box for the paragraph is available for inspection (or changes) inside hooks. It remains available even if the <code>\OmitIndent</code> command was used; in that case it will just not be automatically placed.
<code>\g_para_indent_box</code>	

The expl3 name for the box register is `\g_para_indent_box`.

<code>\RawIndent</code>	<code>\RawIndent</code> <i>hmode material</i> <code>\RawParEnd</code>
<code>\para_raw_indent:</code>	<code>\RawNoindent</code> <i>hmode material</i> <code>\RawParEnd</code>
<code>\RawNoindent</code>	The commands <code>\RawIndent</code> and <code>\RawNoindent</code> are not meant for normal paragraph building (where the result is a textual paragraph in the traditional meaning of the word), but for special cases where TeX's low-level algorithm is used to achieve special effects, but where the result is not a "paragraph".
<code>\para_raw_noindent:</code>	
<code>\RawParEnd</code>	
<code>\para_raw_end:</code>	

They are called "raw", because they bypass L^AT_EX's hook mechanism for paragraphs and simply invoke the low-level TeX algorithm. I.e., they are like the original TeX primitives `\indent` and `\noindent` (that is they execute no hooks other than `\everypar`) except that they can only be used in vertical mode and generate an error if found elsewhere.

To avoid issues a paragraph started by them should always be ended by `\RawParEnd`³⁰ and not by `\par` (or a blank line), because the latter will execute hooks which then have no counterpart at the beginning of the paragraph. It is the responsibility of the programmer to make sure that they are properly paired. This also means that one should not put arbitrary user content between these commands if that content could contain stray `\pars`.

The expl3 names for the functions are `\para_raw_indent:`, `\para_raw_indent:` and `\para_raw_end:`.

2.3 Examples

None of the examples in this section are meant for real use as they are far too simple-minded but they should give some ideas of what could be possible if a bit more care is applied.

2.3.1 Testing the mechanism

The idea is to output for each paragraph encountered some information: a paragraph sequence number, a level number in roman numerals, the environment in which this paragraph appears, and the line number where the start or end of the paragraph is, e.g., something like

³⁰Technical note for those who know their *TeXbook*: the `\RawParEnd` command invokes the original TeX engine definition of `\par` that (solely) triggers the paragraph builder in TeX when found inside unrestricted horizontal mode and does nothing in other processing modes.

```

PARA: 1-i start (document env. on input line 38)
PARA: 1-i end   (document env. on input line 38)
PARA: 2-i start (document env. on input line 40)
PARA: 3-ii start (minipage env. on input line 40)
PARA: 3-ii end   (minipage env. on input line 40)
PARA: 2-i end   (document env. on input line 41)

```

As you can see paragraph 2 starts on line 40 and ends on 41 and inside a minipage started paragraph 3 (start and end on line 40). If you run this on some document you will find that L^AT_EX considers more things “a paragraph” than you have probably thought.

This was generated by the following hook code:

```

\newcounter{paracnt}          % sequence counter
\newcounter{paralevel}       % level counter

```

To support paragraph nesting we need to maintain a stack of the sequence numbers. This is most easily done using expl3 functions, so we switch over. This is not a very general implementation, just enough for what we need and a bit of L^AT_EX 2_ε thrown in as well. When popping, the result gets stored in `\paracntvalue` and the `\ERROR` should never happen because it means we have tried to pop from an empty stack.

```

\ExplSyntaxOn
\seq_new:N \g_para_seq
\cs_new:Npn \ParaPush
  {\seq_gpush:No \g_para_seq {\the\value{paracnt}}}
\cs_new:Npn \ParaPop  {\seq_gpop:MMF \g_para_seq \paracntvalue \ERROR }
\ExplSyntaxOff

```

At the start of the paragraph increment both sequence counter and level and also save the then current sequence number on our stack.

```

\makeatletter % because we use a few internal Te commands
\AddToHook{para/begin}{%
  \stepcounter{paracnt}\stepcounter{paralevel}%
  \ParaPush
}

```

To display the sequence number we `\typeout` the current sequence and level number. The command `\@currenenvir` gives us the current environment and `\on@line` produces a space and the current input line number.

```

\typeout{PARA: \arabic{paracnt}-\roman{paralevel} start
  (\@currenenvir\space env.\on@line)}%

```

We also typeset the sequence number as a tiny red number in a box that takes up no horizontal space. This helps us seeing where L^AT_EX sees the start and end of the paragraphs in the document.

```

\llap{\color{red}\tiny\arabic{paracnt}\ }%
}

```

At the end of the paragraph we display sequence number and level again. The level counter has the correct value but we need to retrieve the right sequence value by popping it off the stack after which it is available in `\paracntvalue` the way we have set this up above.

```

\AddToHook{para/end}{%
  \ParaPop
  \typeout{PARA: \paracntvalue-\roman{paralevel} end \space\space
    (\@currenvir\space env.\on@line)}%
}

```

We also typeset again a tiny red number with that value, this time sticking out to the right.³¹ We also decrement the level counter since our level has finished.

```

  \rlap{\color{red}\tiny\ \paracntvalue}%
  \addtocounter{paralevel}{-1}%
}
\makeatother

```

2.3.2 Mark the first paragraph of each itemize

The code for this is rather simple. We supply some code that is executed only once inside a hook at the start of each `itemize`. We explicitly change the color back and forth so that we don't introduce grouping around the paragraph.

```

\AddToHook{env/itemize/begin}{%
  \AddToHookNext{para/begin}{\color{blue}}%
  \AddToHookNext{para/end}{\color{black}}%
}

```

As a result the first paragraph of each `itemize` will appear in blue.

2.4 Some technical notes

The code tries hard to be transparent for package code, but of course any change means that there is a potential for breaking other code. So in section we collect a few cases that may be of importance if low-level code is dealing with paragraphs that are now behaving slightly differently. The notes are from issues we observed and will probably grow over time.

2.4.1 Glue items between paragraphs (found with `fancypar`)

In the past L^AT_EX placed two glue items between two consecutive paragraphs, e.g.,

```
text1 \par text2 \par
```

would show something like

```

\glue(\parskip) 0.0 plus 1.0
\glue(\baselineskip) 5.16669

```

but now there is another `\parskip` glue (that is always 0pt):

```

\glue(\parskip) 0.0 plus 1.0
\glue(\parskip) 0.0
\glue(\baselineskip) 5.16669

```

³¹Note that this can alter the document pagination, because a paragraph ending in a display (e.g., an equation) will get an extra line—in that case our tiny number has an effect even though it doesn't take up any space, because it paragraph is no longer empty and thus isn't dropped!

The reason is that we generate a “fake” paragraph to gain control and safely add the early hooks, but this generates an additional glue item. That item doesn’t contribute anything vertically but if somebody writes code that unravels a constructed list using `\lastbox`, `\unskip` and `\unpenalty` then the code has to remove one additional glue item or else it will fail.

3 The Implementation

```

1 <@@=para>
2 <*2kernel | latexrelease>
3 \ExplSyntaxOn
4 <latexrelease> \NewModuleRelease{2021/06/01}{ltpara}
5 <latexrelease> {Paragraph-handling-and-hooks}

```

3.1 Providing hooks for paragraphs

`para/before` The public hooks. They are implemented as a paired set of hooks.

```

para/after
para/begin
para/end
6 \hook_new_pair:nm{para/before}{para/after}
7 \hook_new_pair:nm{para/begin}{para/end}

```

(End of definition for para/before and others. These functions are documented on page 413.)

`\@kernel@before@para@before` The corresponding kernel hooks (for tagging and future extensions).

```

\@kernel@after@para@after
\@kernel@before@para@begin
\@kernel@after@para@end
8 \let \@kernel@before@para@before \@empty
9 \let \@kernel@before@para@begin \@empty
10 \let \@kernel@after@para@end \@empty
11 \let \@kernel@after@para@after \@empty

```

(End of definition for \@kernel@before@para@before and others. These functions are documented on page 414.)

`\g__para_standard_everypar_tl`

Whenever \TeX starts a paragraph it inserts first an indentation box and then executes the tokens stored in `\tex_everypar:D` (known to \LaTeX as `\everypar`). We alter this behavior slightly here, so that hooks are added into the right place. Otherwise the process change remains transparent to any legacy code for this space.

We keep the standard code to be used by `\tex_everypar:D` in a separate token list because we have to switch back and forth for error recovery and so altering `\tex_everypar:D` all the time should be a tiny bit faster.

```

12 <latexrelease> \IncludeInRelease{2023/06/01}
13 <latexrelease> { \g__para_standard_everypar_tl } { minipage~ fix }
14 \tl_new:N \g__para_standard_everypar_tl

```

Here is now its definition:

```

15 \tl_gset:Nn \g__para_standard_everypar_tl {

```

First we remove the indentation box and store it in `\g_para_indent_box`. If there was none because the paragraph was started by `\noindent` the box register will be void.

```

16 \box_gset_to_last:N \g_para_indent_box

```

This will make the newly started horizontal list empty, so if we stop it now and return to vertical mode it will be dropped by \TeX . We do that but inside a group so that any `\parshape` settings will not get lost as we need them for later.

```

17 \group_begin:
18 \tex_par:D
19 \group_end:

```

We then change `\tex_everypar:D` to generate an error so that we can detect and report if the `para/before` hook illegally changed out of vmode.

```

20 \tex_everypar:D { \msg_error:nnnn { hooks }{ para-mode }{before}{vertical} }
21 \@kernel@before@para@before
22 \hook_use:n {para/before}

```

Assuming the hooks have been well behaved it is time to return to horizontal mode and start the paragraph in earnest. We already have the indentation box saved away so we now have to restart the paragraph with an empty `\tex_everypar:D` and with `\tex_noindent:D`. And we need to make sure not to get another `\parskip` or rather (since we can't prevent that) that it is of zero size.

```

23 \group_begin:
24 \tex_everypar:D {}

```

There has been a long-standing problem with L^AT_EX's minipages in that invisible material at the beginning of a minipage (such as a `\color` setting) would result in `\parskip` being added in front of the first paragraph—something that is not done by T_EX if a vertical list is completely empty. As this is happening on a very low-level in the engine it wasn't really possible to find out if this `\parskip` was added or if a space we see in front of the current point is legitimate. However, with the new paragraph handling we are in a better position: while we still don't know if there is such a space or not, we do know if we have just created an empty paragraph. Thus, if we now set `\parskip` to `-\parskip` the two will cancel each other if present and if the first was ignored because the vertical list was empty, then the second will be ignored too because it is still empty. Of course, we don't want to cancel always but only at the start of a minipage and that is signaled with the `@minipage` switch.

```

25 \skip_set:Nn \tex_parskip:D
26 { \if@minipage -\tex_parskip:D \else: \c_zero_skip \fi: }
27 \tex_noindent:D
28 \group_end:

```

That brings us back to the start of the horizontal list but we need to change `\tex_everypar:D` back to its normal content in case there are nested paragraphs coming up.

```

29 \tex_everypar:D{\g__para_standard_everypar_tl}

```

This is followed by executing the kernel and the public hook. The kernel hook is there to enable tagging.

```

30 \@kernel@before@para@begin
31 \hook_use:n {para/begin}

```

If we aren't in horizontal mode any longer the hooks above misbehaved.

```

32 \if_mode_horizontal: \else:
33 \msg_error:nnnn { hooks }{ para-mode }{begin}{horizontal} \fi:

```

Finally we reinsert the indentation box (unless suppressed) and then call `\everypar` the way legacy L^AT_EX code expects it.

However, adding the public `\everypar` is a bit tricky (see below) so we add that later, and indirectly.

```

34 \__para_handle_indent:
35 % \the \everypar % <--- done differently below
36 }

```

```

37 <latexrelease> \cs_set:Npn \__para_tmp:w #1#2#3#4#5 { }
38 <latexrelease> \tl_gput_right:Nx \g__para_standard_everypar_tl {
39 <latexrelease>   \exp_not:N \the
40 <latexrelease>   \exp_not:N \toks
41 <latexrelease>   \exp_after:wN \__para_tmp:w \token_to_meaning:N \everypar
42 <latexrelease>   \c_space_tl
43 <latexrelease> }
44 <latexrelease> \EndIncludeInRelease
45 <latexrelease> \IncludeInRelease{2021/06/01}
46 <latexrelease>   {\g__para_standard_everypar_tl}{minipage~ fix}
47 <latexrelease>
48 <latexrelease> \tl_gset:Nn \g__para_standard_everypar_tl {
49 <latexrelease>   \box_gset_to_last:N \g_para_indent_box
50 <latexrelease>   \group_begin:
51 <latexrelease>     \tex_par:D
52 <latexrelease>   \group_end:
53 <latexrelease>   \tex_everypar:D { \msg_error:nnnn { hooks }{ para-mode }{before}{vertical} }
54 <latexrelease>   \@kernel@before@para@before
55 <latexrelease>   \hook_use:n {para/before}
56 <latexrelease>   \group_begin:
57 <latexrelease>     \tex_everypar:D {}
58 <latexrelease>     \skip_zero:N \tex_parskip:D
59 <latexrelease>     \tex_noindent:D
60 <latexrelease>   \group_end:
61 <latexrelease>   \tex_everypar:D{\g__para_standard_everypar_tl}
62 <latexrelease>   \@kernel@before@para@begin
63 <latexrelease>   \hook_use:n {para/begin}
64 <latexrelease>   \if_mode_horizontal: \else:
65 <latexrelease>     \msg_error:nnnn { hooks }{ para-mode }{begin}{horizontal} \fi:
66 <latexrelease>   \__para_handle_indent:
67 <latexrelease> }

```

We also have to add the `\everypar` toks register at the end. In case of rollback this is already allocated and we have to find out the correct number (hope this is correctly done)

```

68 <latexrelease> \cs_set:Npn \__para_tmp:w #1#2#3#4#5 { }
69 <latexrelease> \tl_gput_right:Nx \g__para_standard_everypar_tl {
70 <latexrelease>   \exp_not:N \the
71 <latexrelease>   \exp_not:N \toks
72 <latexrelease>   \exp_after:wN \__para_tmp:w \token_to_meaning:N \everypar
73 <latexrelease>   \c_space_tl
74 <latexrelease> }
75 <latexrelease> \EndIncludeInRelease

```

(End of definition for `\g__para_standard_everypar_tl`.)

`\tex_everypar:D` `\tex_everypar:D` then only has to execute `\g__para_standard_everypar_tl` by default.

```
76 \tex_everypar:D{\g__para_standard_everypar_tl}
```

(End of definition for `\tex_everypar:D`.)

`\everypar` Tokens inserted at the beginning of the paragraph are placed into `\everypar` inside legacy L^AT_EX code, e.g., by the list environments or by headings to handle `\clubpenalty`, etc.

Now this isn't any longer the primitive but simply a toks register used in the code above but to legacy L^AT_EX code that is transparent.

There is, however, a problem: a handful packages use exactly the same trick and replace the primitive with a token register and call the token register inside the renamed primitive. That is they assume that `\everypar` is the primitive and that it will still be called at the start of the paragraph even if renamed.

But if we have already replaced it by a token register then all they do is to give that token register a new name. Thus our code in `\tex_everypar:D` would call `\everypar` (which is now their token register) and the code that they added ends up in our token register which is then never used at all. A bit mind boggling I guess.

So what we have to do is not to call the token register `\everypar` by its name inside `\tex_everypar:D` but by using its actual register number.

```
77 \newtoks \everypar
```

After we have allocated a new toks register with the name `\everypar` the actual register number is available (briefly) inside `\allocationnumber`. So instead of `\the\everypar` we have to put `\the\toks(allocation number)` at the end of `\tex_everypar:D`.

So what remains doing is to append a few tokens to the token list `\g__para_standard_everypar_tl` which we do now. We use `x` expansion here to get the value of `\allocationnumber` in, all the other tokens should not be expanded at this point.

One important point here is to terminate the register allocation number with a real space. This space will get swallowed up when the number is read. Anything else, such as `\scan_stop:` would remain in the input and that would mean that it would interfere with `\everypar` code that attempts to scan ahead to see how the paragraph text starts.

```
78 \tl_gput_right:Nx \g__para_standard_everypar_tl {
79   \exp_not:N \the
80   \exp_not:N \toks
81   \the \allocationnumber
82   \c_space_tl
83 }
```

(End of definition for \everypar.)

`\g_para_indent_box` For managing the indentation we need to provide a public accessible box register

```
84 \box_new:N \g_para_indent_box
```

(End of definition for \g_para_indent_box. This function is documented on page 415.)

`__para_handle_indent:` Adding (typesetting) the indent box is straight forward. If it was emptied before it does nothing.

```
85 \cs_new:Npn \__para_handle_indent: {
86   \box_use_drop:N \g_para_indent_box
87 }
```

The declaration `\para_omit_indent:` (or `\OmitIndent`) changes that to do nothing.

```
88 \cs_new:Npn \para_omit_indent: {
89   \box_gclear:N \g_para_indent_box
90 }
```

(End of definition for __para_handle_indent:.)

`\IndentBox` The L^AT_EX 2_ε names for the indentation box and for suppressing it for use in the `para/begin` hook.

`\OmitIndent`

```
91 \cs_set_eq:NN \IndentBox \g_para_indent_box
92 \cs_set_eq:NN \OmitIndent \para_omit_indent:
```

(End of definition for `\IndentBox` and `\OmitIndent`. These functions are documented on page 415.)

`\para_end:` Adding hooks to the end of a paragraph is similar but here we need to alter the command that is used by T_EX to end horizontal mode and return to vertical mode, i.e., `\par`.

This is a bit more complicated as this command can appear anywhere either explicitly or implicitly added by T_EX in certain situations:

- when using `\par` in the code or the document
- when using a blank line (which is converted to `\par`)
- when T_EX finds any commands incompatible with horizontal mode it issues a `\par` and then rereads the command.

Unfortunately, T_EX has some (these days) unnecessary optimizations: if a `\vbox` ends and T_EX is still in horizontal mode it simply exercises the paragraph builder instead of issuing a `\par`. It is therefore necessary for L^AT_EX to ensure that this case doesn't happen and all boxes internally have a `\par` command at their end.

This `\par` may or may not run the “par primitive” (which is always available as `\tex_par:D` in `expl3`); it is permissible to have a changed meaning and it is in fact changed by L^AT_EX in various ways at various points inside `latex.ltx`. For this L^AT_EX 2_ε code has the following conventions: `\@@par` and `\endgraf` both refer to the default meaning (in the past this was the `initex` primitive) while `\par` is the current meaning which maybe does something else.

We are now going to change this default meaning to instead run `\para_end:`, which ultimately executes the `initex` primitive but additionally adds our hooks when appropriate. This way the change is again transparent to the legacy L^AT_EX 2_ε code.

In most cases `\para_end:` should behave exactly like the primitive and we achieve this by simply expanding it to the primitive which is available to us as `\tex_par:D`. This way we don't have to care about whether T_EX just does nothing (e.g., if in vertical mode already) or generates an error, etc.

```
93 \cs_new_protected:Npn \para_end: {
```

CCC Maybe needs more explanation. TEMP NOTE: What should happen if in outer hmode with an empty hlist?

The only case we care about is when we are in horizontal mode (i.e., doing typesetting) and not also in inner mode (i.e., making paragraphs and not building an `\hbox`).

```
\bool_lazy_and:nnT
  { \mode_if_horizontal_p: }
  { \bool_not_p:n { \mode_if_inner_p: } }
  { ...
```

Since this is executed for each and every paragraph in a document we try to stay as fast as possible, so we do not use the above construct but two conditionals instead. Using low-level `\if_mode...` conditions would be even faster but has the danger to conflict with conditionals in the user hooks.

If `\para_end:` is executed while T_EX is currently doing a low-level assignment the test for horizontal mode may get executed as part of the assignment. That is normally not an issue but we just found one case where it is:


```
\afterassignment\lst@vskip\@tempskipa \z@ \par
```

If T_EX is in hmode while that assignment happens then the `\par` is seen in hmode because in the above case the assignment may not be finished (one should have used `\z@skip`) and the `\lst@vskip` will get inserted into the middle of the conditional. The `\lst@vskip` then changes to vmode and you get a surprising error about the `para/end` hook having changed modes even if you don't have any hook code(!): it is the inserted `\lst@vskip` that is actually causing the change of mode. This is what happened when the output routines got started while a `lstlisting` environment (that redefines `\vskip` in this way) was active. This is really faulty coding, but we try to be proactive and guard the conditional so that any scanning is first stopped, thus:

```
94 \scan_stop:
95 \mode_if_horizontal:TF {
96 \mode_if_inner:F {
```

In that case the action of the primitive would be to remove the last glue (but no kerns) from the horizontal list (constructed to form a paragraph) and then to append a penalty of 10000 and the `\parfillskip`; it then passes the whole list to the paragraph builder, which breaks it into lines and T_EX then returns to vertical mode.

What we want to do is to add this hook code at the end of the horizontal list before any of the above happens. If there was a glue item at the end of the list then it should get removed before the hook code gets added so we have to arrange for this removal.

As in other similar cases, it may be best to add here a `\nobreak` in case the hook itself adds glue and thus creates a non-explicit and unwanted potential breakpoint. On the other hand (as has been argued) the code in the hook should perhaps have the responsibility for adding such a guard penalty in this case. This needs further analysis and decisions (as in emails).

In either case, good documentation of these hooks is essential, covering what the hook may or should provide and all such related considerations concerning the content.

There is not much point in checking if there was really a glue item at the end of the horizontal list, instead we simply try to remove one using `\tex_unskip:D`: if there wasn't one this will do nothing.

```
97 \tex_unskip:D
```

We then execute the public hook (which may add some final typeset material) followed by the kernel hook that we need for adding tagging support. None of this is supposed to change the mode—at the moment we make only a very simple test for this, more devious changes go unnoticed, but too bad as they will then probably backfire badly.

```
98 \hook_use:n{para/end}
99 \@kernel@after@para@end
100 \mode_if_horizontal:TF {
```

The final action (before getting to the point where `\tex_par:D` is called) is to add an extra glue item so that the primitive is prevented from removing intended glue (if there was some). If we don't do this and the horizontal list ends in several glue items we would end up removing two glue items instead of just the last one, which would be wrong. We use glue (rather than a kern) as that will be removed by the primitive.

There is however one other T_EX optimization that hurts: in a sequence like this `$$... $$ \par` (with `\par` being the primitive) T_EX will be in horizontal mode after the display, ready to receive further paragraph text, but since the `\par` follows immediately there is a “null” paragraph at the end and T_EX simply throws that away. The space between `$$` and `\par` got already dropped during the display processing so the `\par` is

not removing any space and appending `\parfillskip`, instead it simply goes silently to vmode.

Now if we would have added something (to prevent glue removal) that would look to \TeX like material after the display and so we would end up with an empty paragraph just containing a penalty and `\parfillskip`.

We therefore check if the current hlist does end in glue (`\tex_lastnodetype:D` has the value 11) and if so we add a zero-length guard skip which will be removed by the following `\tex_par:D`.

```
101     \if_int_compare:w 11 = \tex_lastnodetype:D
102     \tex_hskip:D \c_zero_dim
103     \fi:
```

To run the `para/after` hook we first end the paragraph. This means that the `\tex_par:D` at the very end is unnecessary but executing it there unnecessarily is better than having code that tests for all the different mode possibilities.

```
104     \tex_par:D
105     \hook_use:n{para/after}
106     \@kernel@after@para@after
107     }
```

If we were not horizontal mode (the F case from above) then the earlier hook `para/end` must have been at fault, so we report that.

```
108     { \msg_error:nnnn { hooks }{ para-mode }{end}{horizontal} }
```

Finally close out the nested conditionals.

```
109     }
110 }
```

And then we can use the primitive to truly end the paragraph.

```
111 \tex_par:D
112 }
```

(End of definition for `\para_end:`. This function is documented on page 414.)

`\para_raw_indent:` The commands `\para_raw_indent:` and `\para_raw_noindent:` are like the primitives `\indent` and `\noindent` except that they can only be used in vertical mode.

`\para_raw_end:` To avoid issues a paragraph started by them should always be ended by `\para_raw_end:` and not by `\para_end:` or `\par` as the latter will execute hooks which then have no counterpart at the beginning of the paragraph. It is the responsibility of the programmer to make sure that they are properly paired.

```
113 \cs_new:Npn \para_raw_indent: {
114   \mode_if_vertical:TF
115   {
116     \tex_everypar:D {
117       \box_gset_to_last:N \g_para_indent_box
118       \tex_everypar:D { \g__para_standard_everypar_tl }
119       \__para_handle_indent:
120       \the\everypar }
121   }
122   { \msg_error:nn { latex2e }{ raw-para } }
123 \tex_indent:D
124 }
```

```

125 \cs_new:Npn \para_raw_noindent: {
126   \mode_if_vertical:TF
127     {
128       \tex_everypar:D {
129         \tex_everypar:D { \g__para_standard_everypar_tl }
130         \the\everypar }
131       }
132     { \msg_error:nn { latex2e }{ raw-para } }
133   \tex_noindent:D
134 }
135 \cs_new_eq:NN \para_raw_end: \tex_par:D

```

(End of definition for `\para_raw_indent:`, `\para_raw_noindent:`, and `\para_raw_end:`. These functions are documented on page 415.)

\RawIndent The L^AT_EX 2_ε names for starting and ending a paragraph without adding any hooks.
\RawNoIndent
\RawParEnd

```

136 \cs_set_eq:NN \RawIndent \para_raw_indent:
137 \cs_set_eq:NN \RawNoindent \para_raw_noindent:
138 \cs_set_eq:NN \RawParEnd \para_raw_end:

```

(End of definition for `\RawIndent`, `\RawNoIndent`, and `\RawParEnd`. These functions are documented on page 415.)

This ends the para module code.

```

139 <@@=>

```

\par Having the new default definition for `\par` we also have to set it up so that it gets used.
\endgraf This involves three commands: `\par`, `\@@par` (to which L^AT_EX resets `\par` occasionally)
`\@@par` and `\endgraf`, which is another name for the “default” action of `\par`.

```

140 \cs_set_eq:NN \par \para_end:
141 \cs_set_eq:NN \@@par \para_end:
142 \cs_set_eq:NN \endgraf \para_end:

```

(End of definition for `\par`, `\endgraf`, and `\@@par`. These functions are documented on page 414.)

While this is not integrated properly into the format we have to redo the `\everypar` setting from the kernel, otherwise that gets lost (as it happens before that file is loaded).

```

143 \everypar{\@nodocument} %% To get an error if text appears before the \document

```

3.2 The error messages

This one is used when we detect that some hook code has changed the mode where it shouldn't, e.g., by starting or ending a paragraph. The first argument is the hook name second the mode it should have stayed in but didn't.

```

144 \msg_new:nnnn { hooks } { para-mode }
145 {
146   Illegal-mode~ change~ in~ hook~ 'para/#1'.\
147   Hook~ code~ did~ not~ remain~ in~ #2~ mode.
148 }
149 {
150   Paragraph~ hooks~ cannot~ change~ the~ TeX~ mode~ without~ causing~
151   endless~ recursion.~ The~ hook~ code~ in~ 'para/#1'~ needs~ to~ stay~
152   in~ #2~ mode,~ but~ it~ didn't.~ Examine~ the~ hook~
153   code~ with~ \iow_char:N \ShowHook~ to~ find~ the~ issue.
154 }

```

And here is one used in the “raw” commands when they are used outside of vertical mode.

```

155 \msg_new:nnnn { latex2e } { raw-para }
156 {
157   Not~ in~ vertical~ mode.
158 }
159 {
160   Starting~ a~ paragraph~ with~ \iow_char:N \RawIndent~ or~
161   \iow_char:N \RawNoindent \
162   (or~ \iow_char:N \para_raw_indent:~ or~
163   \iow_char:N \para_raw_noindent:)- is~ only~ allowed \
164   if~ LaTeX~ is~ in~ vertical~ mode.
165 }

166 %
167 <latexrelease>\IncludeInRelease{0000/00/00}%
168 <latexrelease>          {ltpara}{Undo-hooks-for-paragraphs}
169 <latexrelease>
170 <latexrelease>\let \OmitIndent \undefined
171 <latexrelease>\let \IndentBox \undefined
172 <latexrelease>\let \RawIndent \undefined
173 <latexrelease>\let \RawNoindent \undefined
174 <latexrelease>\let \RawParEnd \undefined
175 <latexrelease>
176 <latexrelease>\cs_set_eq:NN \par \tex_par:D
177 <latexrelease>\cs_set_eq:NN \@par \tex_par:D
178 <latexrelease>\cs_set_eq:NN \endgraf \tex_par:D
179 <latexrelease>

```

We also need to clean up the primitive “everypar” as that should no longer execute any code by default. And, of course, make \everypar become the primitive again.

```

180 <latexrelease>\tex_everypar:D {}
181 <latexrelease>\cs_set_eq:NN \everypar \tex_everypar:D
182 <latexrelease>
183 <latexrelease>\EndModuleRelease
184 \ExplSyntaxOff
185 </2ekernel | latexrelease>

```

File 17

ltmeta.dtx

Abstract

This code defines the `\DocumentMetadata` interface.

1 Introduction

In the past there was no dedicated location to declare settings concerning a document as a whole. Settings are placed somewhere in the preamble or with the class options or even with some package options. For some settings this can be too late, for example the pdf version can no longer be changed if a package has used code which already opened the PDF.

`\DocumentMetadata` as a new command unifies such settings in one place. It must be used before `\documentclass` but can be issued more than once there.

At the moment most of the code run by `\DocumentMetadata` is external to the format and subject to change. This includes the supported key/values.

For that reason all that happens right now in the format is to look for suitable support files and if found, to redirect the processing to them.

1.1 `\DocumentMetadata`

```
\DocumentMetadata \DocumentMetadata{<key-value list>}
```

The keys defined for `\DocumentMetadata` currently allow to set the PDF version, to set the PDF `/Lang`, to uncompress a PDF, to set the language and to declare a few PDF standards and some color profiles.

`\DocumentMetadata` is also used to activate the new PDF management code and it loads a number of required files for the PDF management code. As this forces the loading of the backend files, a backend which can't be detected automatically like `dvipdfmx`, must be set in the first `\DocumentMetadata` call (if there is more than one).

The full set of keys currently supported is documented in `documentmetadata-support.pdf` for now.

2 The Implementation

```
1 <*2kernel | latexrelease>
Not needed yet but ...
2 %\ExplSyntaxOn
3 <latexrelease> \NewModuleRelease{2022/06/01}{ltmeta}
4 <latexrelease>           {Document Metadata handling}
5 \let \IfDocumentMetadataTF \@secondoftwo
6 \protected\def\DocumentMetadata{%
7   \InputIfFileExists{documentmetadata-support.ltx}%
8   {}}%
```

The above file is changing `\DocumentMetadata` to a suitable definition (or so we hope), so that we can try again — if not tough.

If the file can't be found we say so and carry on without it.

```

9      {%
10         \latex@error{No support files for
11             \noexpand\DocumentMetadata found}
12             {Is the 'LaTeX-lab' bundle installed?%
13             \MessageBreak
14             Without it, the declaration is ignored.}%

```

No point in trying this more than once if there are several calls in the document.

```

15     \let\DocumentMetadata@gobble
16     }%
17     \let \IfDocumentMetadataTF \@firstoftwo
18     \DocumentMetadata
19 }

```

To allow package and class author to support for document links we provide also the new interface commands of the hyperref package for the creation of targets.

```

\MakeLinkTarget
\LinkTargetOn 20 <latexrelease> \IncludeInRelease{2024/11/01}%
\LinkTargetOff 21 <latexrelease>           {\MakeLinkTarget}{Record target name for tagging support}%
\NextLinkTarget 22 \ExplSyntaxOn
23 \int_new:N\g__kernel_target_int
24 \NewDocumentCommand\MakeLinkTarget{s0{m}}{%
25   \ifvmode
26     \special{}%
27   \else
28     \@savsf\spacefactor
29     \smash{}%
30     \spacefactor\@savsf
31   \fi
32   \IfBooleanTF {#1}
33   {
34     \tl_gset:Ne \@currentHref {#3}
35   }
36   {
37     \int_gincr:N\g__kernel_target_int
38     \tl_gset:Ne \@currentHref {target*.\int_use:N\g__kernel_target_int}
39   }
40   \UseTaggingSocket{recordtarget}
41 }
42 \ExplSyntaxOff
43 <latexrelease> \EndIncludeInRelease
44 <latexrelease> \IncludeInRelease{2022/06/01}%
45 <latexrelease>           {\MakeLinkTarget}{Record target name for tagging support}%
46 <latexrelease> \NewDocumentCommand\MakeLinkTarget{s0{m}}{%
47 <latexrelease>   \ifvmode
48 <latexrelease>     \special{}%
49 <latexrelease>   \else
50 <latexrelease>     \@savsf\spacefactor
51 <latexrelease>     \smash{}%
52 <latexrelease>     \spacefactor\@savsf
53 <latexrelease>   \fi}
54 <latexrelease> \EndIncludeInRelease

```

```

55 \NewDocumentCommand\LinkTargetOn{}{}
56 \NewDocumentCommand\LinkTargetOff{}{}
57 \NewDocumentCommand\NextLinkTarget{m}{}

```

(End of definition for \MakeLinkTarget and others.)

We do not undo `\MakeLinkTarget` and friends if we roll back, in case they are used in packages that themselves do not offer rollback. This way a roll forward adds them, but the dummies remain if you roll back and you don't get missing csname errors if they are used.

```

58 \latexrelease\IncludeInRelease{0000/00/00}{ltmeta}%
59 \latexrelease          {Undo Document Metadata handling}
60 \latexrelease
61 \latexrelease\let\DocumentMetadata\@undefined
62 \latexrelease
63 \latexrelease\EndModuleRelease

```

Again for the future ...

```

64 %\ExplSyntaxOff
65 \</2kernel | latexrelease>
    Restore module prefix (if any):
66 \@@@=

```

File 18

ltspace.dtx

1 Spacing

This section deals with spacing, and line- and page-breaking.

1.1 User Commands

`\nopcodebreak` [$\langle i \rangle$] : $\langle i \rangle = 0, \dots, 4$.
Default argument = 4. Puts a penalty into the vertical list output as follows:
0 : penalty = 0
1 : penalty = `\@lowpenalty`
2 : penalty = `\@medpenalty`
3 : penalty = `\@highpenalty`
4 : penalty = 10000

`\pagebreak` [$\langle i \rangle$] : same as except negatives of its penalty

`\linebreak` [$\langle i \rangle$] : analog of the above

`\nolinebreak` [$\langle i \rangle$] : analog of the above

`\samepage` : inhibits page breaking most places by setting the following penalties to 10000:
`\interlinepenalty`
`\predisplaypenalty`
`\postdisplaypenalty`
`\interdisplaylinepenalty`
`\@beginparpenalty`
`\@endparpenalty`
`\@itempenalty`
`\@secpenalty`
`\interfootnotelinepenalty`

`\` : initially defined to be `\newline`
`\` [$\langle length \rangle$] : initially defined to be `\vspace{\langle length \rangle}\newline`

Note: `\`* adds a `\vadjust{\penalty 10000}`

OBSOLETE COMMANDS (which never made it into the manual):

`\obeycr` : defines `<CR>` == `\``\relax`

`\restorecr` : restores `<CR>` to its usual meaning.

1.2 Chris' comments

There are several aspects of the handling of space in horizontal mode that are inconsistent or do not work well in some cases. These are largely concerned with ignoring the effect of space tokens that would otherwise typeset an inter-word space.

Negating the effect of such space tokens is achieved by two mechanisms:

- `\unskip` is used to remove the glue just added by a space that has already had its effect; it is sometimes invoked after an `\ifdim` test on `\lastskip` (see below);
- `\ignorespaces` is used to ignore space-tokens yet to come.

The test done on `\lastskip` is sometimes for equality with zero and sometimes for being positive. Recall also that the test is only on the natural length of the glue and that no glue cannot be distinguished from glue whose natural length is zero: to summarise, a pretty awful test. It is not clear why these tests are not all the same; I think that they should all be for equality. One place where `\unskip` is often used is just before a `\par` (which itself internally does an `\unskip`) and one bit of code (in `\@item`) even has two `\unskips` before a `\par`. These uses may be fossil code but if they are necessary, maybe `\@killglue` would be even safer.

Such removal of glue by `\unskip` may sometimes have the wrong result, removing not the glue from a space-token but other explicit glue; this is sometimes not what is intended.

A common way to prevent such removal is to add an `\hskip\z@` after the glue that should not be removed. This protects that glue against one `\unskip` with no test but not against more than one. It does work for ‘tested `\unskips`’. This is used by `\hspace*` but not by `\hspace`; this is inconsistent as the star is supposed to prevent removal only at the beginning of a line, not at the end, or in a tabular, etc.

If this reason for removing glue were the only consideration then a tested-`\unskip` and protection by `\hskip\z@` would suffice but would need to be consistently implemented.

However, the class of invisibles, commands and environments tries to be even cleverer: one of these tries to leave only one inter-word space whenever there is one before it and one after it; and it does this quite well.

But problems can arise when there is not a space-token on both sides of it; in particular, when an invisible appears at the beginning or end of a piece of text the method still leaves one space token whereas usually in these cases it should leave none.

Also, the current rules do not work well when more than one such command appears consecutively, separated by space-tokens; it leaves glue between every other invisible.

There is also a question about what these commands should do when they occur next to spaces that do not come from space tokens but, for example, from `\hspace`. Should they still produce ‘just one space’? If so, which one? It is good to note that the manual is sufficiently cautious about invisibles that we are not obliged to make anything work.

Another interesting side-road to explore is whether the space-tokens either side of an `\hspace{...}` should be ignored.

One alternative to the current algorithm that is often suggested is that all glue around the invisible should be consolidated into a space after it (usually without stating how much glue should be put there). The command `\nolinebreak` is implemented this way (and `\linebreak` should also be). This does not work correctly for the following common case:

```
... some text
\index{some-word}
some-word and more text.
```

This is optimal coding since it is normal to index a word that gets split across a page-break on its starting page. This would, on the other hand, fix another common (and documented) failure of the current system: when the invisible is the last thing in a paragraph the space before it is not removed and, worse, it is also hidden from the paragraph-ending mechanism so that an ‘empty’ line can be created at the end of the paragraph.

Another deficiency (I think) of the current system is that the following is treated as having the `\index` command between the paragraphs, which is probably not what the author intended (since there is no empty line after it).

```
\index{beginnings}  
Beginnings of paragraphs ...
```

I know of no algorithm that will handle satisfactorily even all the most common cases; note that it could be that the best algorithm may be different for different invisibles since, for example, the common uses and expected behaviour of `\index`, `\marginpar`, `\linebreak`, `\pagebreak` and `\vspace` are somewhat different. [For example, is `\vspace` ever used in the middle of a paragraph?]

One method that can (and is) used to make invisible commands produce no space when used at the beginning of text is to put in some glue that is nearly enough the same as no glue or glue of zero length in all respects except for the precise test for not being exactly equal to zero; examples of such glue are `\hskip 1sp` and, possibly better but more complex, `\hskip -1sp \hskip 1sp`. However, this only works when it is known that user-supplied text is about to start.

Some similar concerns apply to the handling of space and penalties in vertical mode; there is an extra hurdle here as `\unskip` does not work on the main vertical list. The complexity of the tests done by `\addvspace` have never been explained.

The implementation of space hacks etc for vertical mode is another major area that needs further attention; my earlier experiments did not produce much improvement over the current unsatisfactory situation.

One particular problem is what happens when the following very natural coding is used (part of the problem here is that this looks like an hmode problem, but it is not):

```
... end of text.  
  
\begin{enumerate}  
  \item \label{item:xxx} Item text.  
\end{enumerate}
```

1.3 Some immediate actions

- Fix bug in `\linebreak`.
- Fix bug in `*`.
- Reimplement `\\`, etc, removing extra `\vadjusts` and getting better error trapping (this seems to involve a lot more tokens).
- Investigate whether `\\`, etc need to be errors in vmode; I think that they could be noops (maybe with a warning).
- Make all(?) `\unskips` include test for zero skip (rather than other tests or no test).
- Consider replacing `\hskip 1sp` by something better (here called an ‘infinitesimal’ skip).
- Look at all `\hskip\z@` (or similar) to see if they should be changed to an ‘infinitesimal’ skip.

- Resolve the inconsistency between `\hspace` and `\hspace*`.
- Remove unnecessary `\unskips`.
- Investigate and rationalise the ‘newline’ code.
- Find better algorithms for all sorts of things or, easier(?), fix T_EX itself.

1.4 The code

```

1 <*2ekernel>
2 \message{spacing,}
3 </2ekernel>
4 <*2ekernel | latexrelease>
5 <latexrelease>\IncludeInRelease{2019/10/01}%
6 <latexrelease>                {\pagebreak}{Make commands robust}%

\pagebreak
\nopagebreak
7 \DeclareRobustCommand\pagebreak{\@testopt{\@no@pgbk-}4}
8 \DeclareRobustCommand\nopagebreak{\@testopt{\@no@pgbk4}

(End of definition for \pagebreak and \nopagebreak.)

\linebreak
\nolinebreak
9 \DeclareRobustCommand\linebreak{\@testopt{\@no@lnbk-}4}
10 \DeclareRobustCommand\nolinebreak{\@testopt{\@no@lnbk4}

(End of definition for \linebreak and \nolinebreak.)

\samepage
11 \DeclareRobustCommand\samepage{\interlinepenalty\@M
12   \predisplaypenalty\@M
13   \postdisplaypenalty\@M
14   \interdisplaylinepenalty\@M
15   \@beginparpenalty\@M
16   \@endparpenalty\@M
17   \@itempenalty\@M
18   \@secpenalty\@M
19   \interfootnotelinepenalty\@M}

(End of definition for \samepage.)

20 </2ekernel | latexrelease>
21 <latexrelease>\EndIncludeInRelease
22 <latexrelease>\IncludeInRelease{0000/00/00}%
23 <latexrelease>                {\pagebreak}{Make commands robust}%
24 <latexrelease>
25 <latexrelease>\kernel@make@fragile\pagebreak
26 <latexrelease>\kernel@make@fragile\nopagebreak
27 <latexrelease>\kernel@make@fragile\linebreak
28 <latexrelease>\kernel@make@fragile\nolinebreak
29 <latexrelease>\kernel@make@fragile\samepage
30 <latexrelease>
31 <latexrelease>\EndIncludeInRelease
32 <*2ekernel>

```

`\@no@pgbk`

```
33 \def\@no@pgbk #1[#2]{%
34   \ifvmode
35     \penalty #1\@getpen{#2}%
36   \else
37     \@bsphack
38     \vadjust{\penalty #1\@getpen{#2}}%
39     \@esphack
40   \fi}
```

(End of definition for \@no@pgbk.)

`\@no@lmbk`

```
41 \def\@no@lmbk #1[#2]{%
42   \ifvmode
43     \@nolnerr
44   \else
45     \@tempskipa\lastskip
46     \unskip
47     \penalty #1\@getpen{#2}%
48     \ifdim\@tempskipa>\z@
49       \hskip\@tempskipa
50       \ignorespaces
51     \fi
52   \fi}
```

(End of definition for \@no@lmbk.)

`\@` The purpose of the new code is to fix a few bugs; however, it also attempts to optimize the following, in order of priority:

1. efficient execution of plain `\@`;
2. efficient execution of `\@[...]`;
3. memory use;
4. name-space use.

The changes should make no difference to the typeset output. It appears to be safe to use `\reserved@e` and `\reserved@f` here (other reserved macros are somewhat disastrous).

These changes made `\newline` even less robust than it had been, so now it is explicitly robust, like `\@`.

The internal definition of the ‘normal’ definition of `\@`.

`\@normalcr`

```
53 </2kernel>
54 < *2kernel | latexrelease >
55 < latexrelease > \IncludeInRelease{2020/02/02}%
56 < latexrelease >           {\@normalcr}{Make robust}%
57 \protected\def\@normalcr{%
58   \let \reserved@e \relax
59   \let \reserved@f \relax
60   \@ifstar{\let \reserved@e \vadjust \let \reserved@f \nobreak
61             \@xnewline}%
62   \@xnewline}
```

```

63 \let\\@normalcr
64 </2ekernel | latexrelease>
65 <latexrelease>\EndIncludeInRelease
66 <latexrelease>\IncludeInRelease{0000/00/00}%
67 <latexrelease>          {\@normalcr}{Make robust}%
68 <latexrelease>
69 <latexrelease>\DeclareRobustCommand\\{%
70 <latexrelease>  \let \reserved@e \relax
71 <latexrelease>  \let \reserved@f \relax
72 <latexrelease>  \@ifstar{\let \reserved@e \vadjust \let \reserved@f \nobreak
73 <latexrelease>          \@xnewline}%
74 <latexrelease>  \@xnewline}
75 <latexrelease>\expandafter\let\expandafter\@normalcr
76 <latexrelease>  \csname\expandafter@gobble\string\\ \endcsname
77 <latexrelease>
78 <latexrelease>\EndIncludeInRelease
79 <*2ekernel>

```

(End of definition for \\ and \@normalcr.)

\@vspace@calcify Helper command to produce a \vskip that is first run through \setlength. This way the calc package can operate on the argument value.

```

80 </2ekernel>
81 <*2ekernel | latexrelease>
82 <latexrelease>\IncludeInRelease{2020/10/01}%
83 <latexrelease>          {\@vspace@calcify}{Add calc support}%
84 \def\@vspace@calcify#1{\setlength\sp@ce@skip{#1}\vskip\sp@ce@skip}
85 </2ekernel | latexrelease>
86 <latexrelease>\EndIncludeInRelease
87 <latexrelease>\IncludeInRelease{0000/00/00}%
88 <latexrelease>          {\@vspace@calcify}{Add calc support}%
89 <latexrelease>
90 <latexrelease>\let\@vspace@calcify\@undefined
91 <latexrelease>\EndIncludeInRelease
92 <*2ekernel>

```

(End of definition for \@vspace@calcify.)

\newline A simple form of the ‘normal’ definition of \\.

```

93 \DeclareRobustCommand\newline{\@normalcr\relax}

```

(End of definition for \newline.)

\@xnewline

```

94 \def\@xnewline{\@ifnextchar[% ] bracket matching
95                \@newline
96                {\@gnewline\relax}}

```

(End of definition for \@xnewline.)

\@newline

```

97 </2ekernel>
98 <*2ekernel | latexrelease>
99 <latexrelease>\IncludeInRelease{2020/10/01}%

```

```

100 <latexrelease>                {\@newline}{\newline calc support}%
101 \def\@newline[#1]{\let \reserved@e \vadjust
102                \@gnewline {\@vspace@calcfy{#1}}}
103 </2ekernel | latexrelease>
104 <latexrelease>\EndIncludeInRelease

105 <latexrelease>\IncludeInRelease{0000/00/00}%
106 <latexrelease>                {\@newline}{\newline calc support}%
107 <latexrelease>
108 <latexrelease>\def\@newline[#1]{\let \reserved@e \vadjust
109 <latexrelease>                \@gnewline {\vskip #1}}
110 <latexrelease>\EndIncludeInRelease
111 <*2ekernel>

```

(End of definition for \@newline.)

\@gnewline The `\nobreak` added to prevent null lines when `\\` ends an overfull line. Change made 24 May 89 as suggested by Frank Mittelbach and Rainer Schöpf

```

112 \def\@gnewline #1{%
113     \ifvmode
114         \@nolnerr
115     \else
116         \unskip \reserved@e {\reserved@f#1}\nobreak \hfil \break
117     \fi}

```

(End of definition for \@gnewline.)

\@getpen

```

118 \def\@getpen#1{\ifcase #1 \z@ \or \@lowpenalty\or
119             \@medpenalty \or \@highpenalty
120             \else \@M \fi}

```

(End of definition for \@getpen.)

\if@nobreak Switch used to avoid page breaks caused by `\label` after a section heading, etc. It should be **GLOBALLY** set true after the `\nobreak` and **globally** set false by the next invocation of `\everypar`.

Commands that reset `\everypar` should globally set it false if appropriate.

```

121 \def\@nobreakfalse{\global\let\if@nobreak\iffalse}
122 \def\@nobreaktrue {\global\let\if@nobreak\iftrue}
123 \@nobreakfalse

```

(End of definition for \if@nobreak.)

\@savsk Registers used to save the space factor and last skip.

```

\@savsf 124 \newdimen\@savsk
125 \newcount\@savsf

```

(End of definition for \@savsk and \@savsf.)

`\@bsphack` `\@bsphack` and `\@esphack` used by macros such as `\index` and `\begin{@float} ... \end{@float}` that want to be invisible — i.e., not leave any extra space when used in the middle of text. Such a macro should begin with `\@bsphack` and end with `\@esphack`. The macro in question should not create any text, nor change the mode.

Before giving the current definition we give an extended definition that is currently not used (because it doesn't work as advertised:-)

These are generalised hacks which attempt to do sensible things when 'invisible commands' appear in vmode too.

They need to cope with space in both hmode (plus spacefactor) and vmode, and also cope with breaks etc. In vmode this means ensuring that any following `\addvspace`, etc sees the correct glue in `\lastskip`.

In fact, these improved versions should be used for other cases of 'whatsits, thingies etc' which should be invisible. They are only for commands, not environments (see notes on `\@Esphack`).

BTW, anyone know why the standard hacks are surrounded by `\ifmmode\else` rather than simply `\ifhmode`?

And are there any cases where saving the spacefactor is essential? I have some extensions where it is, but it does not appear to be so in the standard uses.

```
def \@bsphack{%
  \relax \ifvmode
    \@savsk \lastskip
    \ifdim \lastskip=\z@
      \else
        \vskip -\lastskip
      \fi
    \else
      \ifhmode
        \@savsk \lastskip
        \@savsf \spacefactor
      \fi
    \fi
  \fi
}
```

I think that, in vmode, it is the safest to put in a `\nobreak` immediately after such things since writes, inserts etc followed by glue give valid breakpoints and, in general, it is possible to create breaks but impossible to destroy them.

```
def \@esphack{%
  \relax \ifvmode
    \nobreak
    \ifdim \@savsk=\z@
      \else
        \vskip \@savsk
      \fi
    \else
      \ifhmode
        \spacefactor \@savsf
        \ifdim \@savsk>\z@
          \ignorespaces
        \fi
      \fi
    \fi
  \fi
}
```

```

    \fi
\fi

```

For the moment we are going to ignore the vertical versions until they are correct.

```

126 \def\@bsphack{%
127   \relax
128   \ifhmode
129     \@savsk\lastskip
130     \@savsf\spacefactor
131   \fi}

```

(End of definition for \@bsphack.)

\@esphack Companion to \@bsphack. If this command is not properly paired with \@bsphack one might end up with a low-level T_EX error: “BAD spacefactor”. One possible cause is calling \@bsphack in vertical mode, then doing something that gets you (sometimes) into horizontal mode and finally calling \@esphack. Even if no error is generated that is wrong, because \@esphack will then use the saved values for \@savsk and \@savsf from some earlier invocation of \@bsphack which will have nothing to do with the current situation.

```

132 </2kernel>
133 <latexrelease>\IncludeInRelease{2018/10/10}%
134 <latexrelease>           {\@esphack}{hyphenation and nobreak after space hack}%
135 <*2kernel | latexrelease>
136 \def\@esphack{%
137   \relax
138   \ifhmode
139     \spacefactor\@savsf
140     \ifdim\@savsk>\z@
141
142       \ifdim\lastskip=\z@
143         \nobreak \hskip\z@skip
144       \fi
145       \ignorespaces
146     \fi
147
148   \else
149     \ifvmode
150       \if@nobreak\nobreak\else\if@noskipsec\nobreak\fi\fi
151     \fi
152   \fi}%
153 </2kernel | latexrelease>
154 <latexrelease>\EndIncludeInRelease
155 <latexrelease>\IncludeInRelease{2015/10/01}%
156 <latexrelease>           {\@esphack}{hyphenation and nobreak after space hack}%
157 <latexrelease>\def\@esphack{%
158 <latexrelease>   \relax
159 <latexrelease>   \ifhmode
160 <latexrelease>     \spacefactor\@savsf
161 <latexrelease>     \ifdim\@savsk>\z@
162 <latexrelease>       \ifdim\lastskip=\z@
163 <latexrelease>         \nobreak \hskip\z@skip
164 <latexrelease>       \fi
165 <latexrelease>     \fi
166 <latexrelease>   \else
167 <latexrelease>     \ifvmode
168 <latexrelease>       \if@nobreak\nobreak\else\if@noskipsec\nobreak\fi\fi
169 <latexrelease>     \fi
170 <latexrelease>   \fi}%

```



```

163 <latexrelease> \ignorespaces
164 <latexrelease> \fi
165 <latexrelease> \fi}%
166 <latexrelease>\EndIncludeInRelease
167 <latexrelease>\IncludeInRelease{2015/01/01}%
168 <latexrelease> \{\@esphack}{hyphenation and nobreak after space hack}%
169 <latexrelease>\def\@esphack{%
170 <latexrelease> \relax
171 <latexrelease> \ifhmode
172 <latexrelease> \spacefactor\@savsf
173 <latexrelease> \ifdim\@savsk>\z@
174 <latexrelease> \nobreak \hskip\z@skip
175 <latexrelease> \ignorespaces
176 <latexrelease> \fi
177 <latexrelease> \fi}%
178 <latexrelease>\EndIncludeInRelease
179 <latexrelease>\IncludeInRelease{0000/00/00}%
180 <latexrelease> \{\@esphack}{hyphenation and nobreak after space hack}%
181 <latexrelease>\def\@esphack{%
182 <latexrelease> \relax
183 <latexrelease> \ifhmode
184 <latexrelease> \spacefactor\@savsf
185 <latexrelease> \ifdim\@savsk>\z@
186 <latexrelease> \ignorespaces
187 <latexrelease> \fi
188 <latexrelease> \fi}%
189 <latexrelease>\EndIncludeInRelease
190 <*2ekernel>

```

(End of definition for \@esphack.)

\@Esphack A variant of \@esphack that sets the @ignore switch to true (as \@esphack used to do previously). This is currently used only for floats and similar environments.

```

191 </2ekernel>
192 <latexrelease>\IncludeInRelease{2015/01/01}%
193 <latexrelease> \{\@Esphack}{hyphenation after space hack}%
194 <*2ekernel | latexrelease>
195 \def\@Esphack{%
196 \relax
197 \ifhmode
198 \spacefactor\@savsf
199 \ifdim\@savsk>\z@
200 \nobreak \hskip\z@skip
201 \@ignoretrue
202 \ignorespaces
203 \fi
204 \fi}%
205 </2ekernel | latexrelease>
206 <latexrelease>\EndIncludeInRelease
207 <latexrelease>\IncludeInRelease{0000/00/00}%
208 <latexrelease> \{\@Esphack}{hyphenation after space hack}%
209 <latexrelease>\def\@Esphack{%
210 <latexrelease> \relax
211 <latexrelease> \ifhmode

```

```

212 <latexrelease> \spacefactor\@savsf
213 <latexrelease> \ifdim\@savsk>\z@
214 <latexrelease> \@ignoretrue
215 <latexrelease> \ignorespaces
216 <latexrelease> \fi
217 <latexrelease> \fi}%
218 <latexrelease>\EndIncludeInRelease
219 <*2ekernel>

```

(End of definition for \@Esphack.)

`\@vbsphack` Another variant which is useful for invisible things which should not live in vmode (this is how some people feel about marginals).

If it occurs in vmode then it enters hmode and ensures that `\@savsk` is nonzero so that the `\ignorespaces` is put in later. It is not used at present.

```

\def \@vbsphack{ %
  \relax \ifvmode
    \leavevmode
    \@savsk 1sp
    \@savsf \spacefactor
  \else
    \ifhmode
      \@savsk \lastskip
      \@savsf \spacefactor
    \fi
  \fi
}

```

(End of definition for \@vbsphack.)

1.5 Vertical spacing

L^AT_EX supports the plain T_EX commands `\smallskip`, `\medskip` and `\bigskip`. However, it redefines them using `\vspace` instead of `\skip`.

Extra vertical space is added by the command `\addvspace{<skip>}`, which adds a vertical skip of `<skip>` to the document. The sequence `\addvspace{<s1>} \addvspace{<s2>}` is equivalent to `\addvspace{<maximum of s1, s2>}`.

`\addvspace` should be used only in vertical mode, and gives an error if it's not. The `\addvspace` command does *not* add vertical space if `@minipage` is true. The `minipage` environment uses this to inhibit the addition of extra vertical space at the beginning.

Penalties are put into the vertical list with the `\addpenalty{<penalty>}` command. It works properly when `\addpenalty` and `\addvspace` commands are mixed.

The `@nobreak` switch is set true used when in vertical mode and no page break should occur. (Right now, it is used only by the section heading commands to inhibit page breaking after a heading.)

`\@xaddvskip` Internal macro for `\vspace` handling the case that space has previously been added.

```

220 \def \@xaddvskip{%
221   \ifdim\lastskip<\@tempskipb
222     \vskip-\lastskip

```

```

223     \vskip\@tempskipb
224   \else
225     \ifdim\@tempskipb<\z@
226       \ifdim\lastskip<\z@
227         \else
228           \advance\@tempskipb\lastskip
229           \vskip-\lastskip
230           \vskip \@tempskipb
231         \fi
232       \fi
233     \fi}

```

(End of definition for \@xaddvskip.)

\addvspace Add vertical space taking into account space already added, as described above.

```

234 </2ekernel>
235 <*2ekernel | latexrelease>
236 <latexrelease>\IncludeInRelease{2024/11/01}%
237 <latexrelease>          {\addvspace}{drop unnecessary no-item error}%
238 \protected\def\addvspace#1{%

```

When this is encountered in hmode, we check whether we are in an hbox and if so generate a L^AT_EX error, as otherwise this would cause a bunch of low-level errors. In unrestricted hmode we simply switch to vmode by issuing a `\par`.

```

239   \ifhmode \ifinner \@LRmoderr \else \par \fi \fi
240   \if@minipage\else
241     \ifdim \lastskip =\z@
242       \@vspace@calcify{#1}%
243     \else
244       \setlength\@tempskipb{#1}%
245       \@xaddvskip
246     \fi
247   \fi
248 }
249 </2ekernel | latexrelease>
250 <latexrelease>\EndIncludeInRelease
251 <latexrelease>\IncludeInRelease{2020/10/01}%
252 <latexrelease>          {\addvspace}{\addvspace calc support}%
253 <latexrelease>\def\addvspace#1{%
254 <latexrelease>  \ifvmode
255 <latexrelease>    \if@minipage\else
256 <latexrelease>      \ifdim \lastskip =\z@
257 <latexrelease>        \@vspace@calcify{#1}%
258 <latexrelease>      \else
259 <latexrelease>        \setlength\@tempskipb{#1}%
260 <latexrelease>        \@xaddvskip
261 <latexrelease>      \fi
262 <latexrelease>    \fi
263 <latexrelease>  \else
264 <latexrelease>    \@noitemerr
265 <latexrelease>  \fi}
266 <latexrelease>\EndIncludeInRelease

```

```

267 <latexrelease>\IncludeInRelease{0000/00/00}%
268 <latexrelease>          {\addvspace}{\addvspace calc support}%
269 <latexrelease>
270 <latexrelease>\def\addvspace#1{%
271 <latexrelease>  \ifvmode
272 <latexrelease>    \if@minipage\else
273 <latexrelease>      \ifdim \lastskip =\z@
274 <latexrelease>        \vskip #1\relax
275 <latexrelease>      \else
276 <latexrelease>        \@tempskipb#1\relax
277 <latexrelease>        \xaddvskip
278 <latexrelease>        \fi
279 <latexrelease>      \fi
280 <latexrelease>    \else
281 <latexrelease>      \@noitemerr
282 <latexrelease>    \fi}
283 <latexrelease>\EndIncludeInRelease
284 <*2ekernel>

```

(End of definition for \addvspace.)

\addpenalty

```

285 </2ekernel>
286 <latexrelease>\IncludeInRelease{2024/11/01}%
287 <latexrelease>          {\addpenalty}{\addpenalty drop error}%
288 <*2ekernel | latexrelease>
289 \protected\def\addpenalty#1{%

```

See description of \addvspace for documentation of the next line of code.

```

290  \ifhmode \ifinner \@LRmoderr \else \par \fi \fi

```

Fix provided by Donald (though the original fix was not good enough). In 2005 Plamen Tanovski discovered that this fix wasn't good enough either as the \vskip kept getting bigger if several \addpenalty commands followed each other. Donald kindly send a new fix.

```

291  \if@minipage
292  \else
293    \if@nobreak
294    \else
295      \ifdim\lastskip=\z@
296      \penalty#1\relax
297    \else
298      \@tempskipb\lastskip

```

We have to make sure the final \vskip seen by T_EX is the correct one, namely \@tempskipb. However, we may have to adjust for \prevdepth when placing the penalty; that should not affect the skip we pass on to T_EX.

```

299      \begingroup
300      \@tempskipa\@tempskipb
301      \advance \@tempskipb
302      \ifdim\prevdepth>\maxdepth\maxdepth\else

```

If `\prevdepth` is `-1000pt` due to `\nointerlineskip` we better not add it!

```
303         \ifdim \prevdepth = -\@m\p@ \z@ \else \prevdepth \fi
304         \fi
305         \vskip -\@tempskipb
306         \penalty#1%
307         \ifdim\@tempskipa=\@tempskipb
```

Do nothing if the `\prevdepth` check made no adjustment.

```
308         \else
```

Combine the `prevdepth` adjustment into a single skip.

```
309         \advance\@tempskipb -\@tempskipa
310         \vskip \@tempskipb
311         \fi
```

The final skip is always the specified length.

```
312         \vskip \@tempskipa
313     \endgroup
314 \fi
315 \fi
316 \fi
317 }
```

```
318 </2kernel | latexrelease>
```

```
319 <latexrelease>\EndIncludeInRelease
```

```
320 <latexrelease>\IncludeInRelease{2015/01/01}%
```

```
321 <latexrelease>         {\addpenalty}{\addpenalty}%
```

```
322 <latexrelease>\def\addpenalty#1{%
```

```
323 <latexrelease> \ifvmode
```

```
324 <latexrelease>     \if@minipage
```

```
325 <latexrelease>     \else
```

```
326 <latexrelease>         \if@nobreak
```

```
327 <latexrelease>         \else
```

```
328 <latexrelease>             \ifdim\lastskip=\z@
```

```
329 <latexrelease>             \penalty#1\relax
```

```
330 <latexrelease>         \else
```

```
331 <latexrelease>             \@tempskipb\lastskip
```

```
332 <latexrelease>             \begingroup
```

```
333 <latexrelease>                 \@tempskipa\@tempskipb
```

```
334 <latexrelease>                 \advance \@tempskipb
```

```
335 <latexrelease>                     \ifdim\prevdepth>\maxdepth\maxdepth\else
```

```
336 <latexrelease>                         \ifdim \prevdepth = -\@m\p@ \z@ \else \prevdepth \fi
```

```
337 <latexrelease>                     \fi
```

```
338 <latexrelease>                 \vskip -\@tempskipb
```

```
339 <latexrelease>                 \penalty#1%
```

```
340 <latexrelease>                 \ifdim\@tempskipa=\@tempskipb
```

```
341 <latexrelease>                 \else
```

```
342 <latexrelease>                     \advance\@tempskipb -\@tempskipa
```

```
343 <latexrelease>                     \vskip \@tempskipb
```

```
344 <latexrelease>                 \fi
```

```
345 <latexrelease>                 \vskip \@tempskipa
```

```
346 <latexrelease>             \endgroup
```

```
347 <latexrelease>         \fi
```

```
348 <latexrelease>     \fi
```

```
349 <latexrelease> \fi
```

```
350 <latexrelease> \else
```

```

351 <latexrelease> \@noitemerr
352 <latexrelease> \fi}%

353 <latexrelease>\EndIncludeInRelease
354 <latexrelease>\IncludeInRelease{0000/00/00}%
355 <latexrelease> \addpenalty{\addpenalty}%
356 <latexrelease>\def\addpenalty#1{%
357 <latexrelease> \ifvmode
358 <latexrelease> \if@minipage
359 <latexrelease> \else
360 <latexrelease> \if@nobreak
361 <latexrelease> \else
362 <latexrelease> \ifdim\lastskip=\z@
363 <latexrelease> \penalty#1\relax
364 <latexrelease> \else
365 <latexrelease> \@tempskipb\lastskip
366 <latexrelease> \vskip -\lastskip
367 <latexrelease> \penalty#1%
368 <latexrelease> \vskip\@tempskipb
369 <latexrelease> \fi
370 <latexrelease> \fi
371 <latexrelease> \fi
372 <latexrelease> \else
373 <latexrelease> \@noitemerr
374 <latexrelease> \fi}%
375 <latexrelease>\EndIncludeInRelease
376 <*2ekernel>

```

(End of definition for \addpenalty.)

\vspace
\@vspace
\@vspacer

The new code for these commands depends on the following facts:

- The value of prevdepth is changed only when a box or rule is created and added to a vertical list;
- The value of prevdepth is used only when a box is created and added to a vertical list;
- The value of prevdepth is always local to the building of one vertical list.

```

377 \DeclareRobustCommand\vspace{\@ifstar\@vspacer\@vspace}
378 </2ekernel>
379 <*2ekernel | latexrelease>
380 <latexrelease>\IncludeInRelease{2020/10/01}%
381 <latexrelease> \@vspace}{Support calc in \vspace}%

```

We support calc syntax in the argument and therefore use \setlength.

```

382 \def\@vspace #1{%
383 \ifvmode
384 \@vspace@calcify{#1}%
385 \vskip\z@skip
386 \else
387 \@bsphack
388 \vadjust{\@restorepar
389 \@vspace@calcify{#1}%
390 \vskip\z@skip

```

```

391         }%
392     \@esphack
393 \fi}
394 \def\@vspacer#1{%
395     \ifvmode
396         \dimen@\prevdepth
397         \hrule \@height\z@
398         \nobreak
399         \@vspace@calcify{#1}%
400         \vskip\z@skip
401         \prevdepth\dimen@
402     \else
403         \@bsphack
404         \vadjust{\@restorepar
405             \hrule \@height\z@
406             \nobreak
407             \@vspace@calcify{#1}%
408             \vskip\z@skip}%
409         \@esphack
410     \fi}
411 </2ekernel | latexrelease>
412 <latexrelease>\EndIncludeInRelease
413 <latexrelease>\IncludeInRelease{0000/00/00}%
414 <latexrelease>         {\@vspace}{Support calc in \vspace}%
415 <latexrelease>
416 <latexrelease>\def\@vspace #1{%
417 <latexrelease>     \ifvmode
418 <latexrelease>         \vskip #1
419 <latexrelease>         \vskip\z@skip
420 <latexrelease>     \else
421 <latexrelease>         \@bsphack
422 <latexrelease>         \vadjust{\@restorepar
423 <latexrelease>             \vskip #1
424 <latexrelease>             \vskip\z@skip
425 <latexrelease>             }%
426 <latexrelease>         \@esphack
427 <latexrelease>     \fi}
428 <latexrelease>\def\@vspacer#1{%
429 <latexrelease>     \ifvmode
430 <latexrelease>         \dimen@\prevdepth
431 <latexrelease>         \hrule \@height\z@
432 <latexrelease>         \nobreak
433 <latexrelease>         \vskip #1
434 <latexrelease>         \vskip\z@skip
435 <latexrelease>         \prevdepth\dimen@
436 <latexrelease>     \else
437 <latexrelease>         \@bsphack
438 <latexrelease>         \vadjust{\@restorepar
439 <latexrelease>             \hrule \@height\z@
440 <latexrelease>             \nobreak
441 <latexrelease>             \vskip #1
442 <latexrelease>             \vskip\z@skip}%
443 <latexrelease>         \@esphack
444 <latexrelease>     \fi}

```

```

445 <latexrelease>\EndIncludeInRelease
446 <*2ekernel>

(End of definition for \vspace, \@vspace, and \@vspacer.)

```

```

\smallskip
\medskip 447 \def\smallskip{\vspace\smallskipamount}
\bigskip 448 \def\medskip{\vspace\medskipamount}
449 \def\bigskip{\vspace\bigskipamount}

(End of definition for \smallskip, \medskip, and \bigskip.)

```

```

\smallskipamount
\medskipamount 450 \newskip\smallskipamount \smallskipamount=3pt plus 1pt minus 1pt
\bigskipamount 451 \newskip\medskipamount \medskipamount =6pt plus 2pt minus 2pt
452 \newskip\bigskipamount \bigskipamount =12pt plus 4pt minus 4pt

(End of definition for \smallskipamount, \medskipamount, and \bigskipamount.)

```

1.6 Horizontal space (and breaks)

`\nobreakdashes` This idea is borrowed from the `amsmath` package but here we define a robust command. This command is a low-level command designed for use only before hyphens or dashes (such as `-`, `--`, or `---`).

It could probably be better implemented: it may need its own private token register and temporary command.

Setting the hyphen in a box and then unboxing it means that the normal penalty will not be added after it—and if the penalty is not there a break will not be taken (unless an explicit penalty or glue follows, thus the final `\nobreak`).

Note that even if it is not followed by a ‘-’, it still leaves `vmode` and sets the spacefactor; so use it carefully!

```

453 \DeclareRobustCommand{\nobreakdashes}{%
454   \leavevmode
455   \toks@{}%
456   \def\reserved@a##1{\toks@\expandafter{\the\toks@-}%
457     \futurelet\@let@token \reserved@b}%
458   \def\reserved@b  {\ifx\@let@token -%
459     \expandafter\reserved@a
460     \else
461     \setbox\z@ \hbox{\the\toks@\nobreak}%
462     \unhbox\z@
463     \spacefactor\sfcode'\-
464     \fi}%
465   \futurelet\@let@token \reserved@b
466 }

```

(End of definition for `\nobreakdashes`.)

`\nobreakspace` This is a robust command that produces a horizontal space at which, in paragraph-mode, `\@xobeysp` a line-break is not possible. We then define an active `~` to expand to it since this is the documented behaviour of `~`. One reason for introducing this is that some 8-bit input encodings have a slot for such a space and we do not want to use active characters as the \LaTeX internal commands.

The braces in the definition of `~` are needed to ensure that a following space is preserved when reading to/from internal files.

We need to keep `\@xobeysp` as it is widely used; so here it is let to the non-robust command `\nobreakspace` .

The fragile version of `~` needs a brace group after `\nobreakspace` to prevent loss of spaces if it occurs in an expansion context. That's not an issue with the updated `\protected` definition, so we keep the code shorter and avoid that.

```

467 \DeclareRobustCommand{\nobreakspace}{%
468   \leavevmode\nobreak\ }
469 \catcode '\-=13
470 \</2ekernel
471 \<latexrelease>\IncludeInRelease{2023/11/01}%
472 \<latexrelease>           {\tilde}{Protected tilde}%
473 \<*2ekernel | latexrelease>
474 \protected\edef~{%
475   \noexpand\ifincsname\noexpand\expandafter\string~%
476   \noexpand\else
477     \noexpand\expandafter\noexpand\nobreakspace
478   \noexpand\fi
479 }
480 \</2ekernel | latexrelease>
481 \<latexrelease>\EndIncludeInRelease
482 \<latexrelease>\IncludeInRelease{0000/00/00}%
483 \<latexrelease>           {\tilde}{Protected tilde}%
484 \<latexrelease>\def~{\nobreakspace{}}
485 \<latexrelease>\EndIncludeInRelease
486 \<*2ekernel
487 \expandafter\let\expandafter\@xobeysp\csname nobreakspace \endcsname

```

(End of definition for \nobreakspace and \@xobeysp.)

`\@xobeytab` Equivalent to the space case with the default settings.

```

488 \</2ekernel
489 \<latexrelease>\IncludeInRelease{2023/11/01}%
490 \<latexrelease>           {\@xobeytab}{Obeyed tabs}%
491 \<*2ekernel | latexrelease>
492 \let\@xobeytab\@xobeysp
493 \</2ekernel | latexrelease>
494 \<latexrelease>\EndIncludeInRelease
495 \<latexrelease>\IncludeInRelease{0000/00/00}%
496 \<latexrelease>           {\@xobeytab}{Obeyed tabs}%
497 \<latexrelease>\let\@xobeytab\@undefined
498 \<latexrelease>\EndIncludeInRelease
499 \<*2ekernel

```

(End of definition for \@xobeytab.)

`\@` Placed before a `'`, makes it a sentence-ending period. Does the right thing for other punctuation marks as well. Does this by setting spacefactor to 1000.

```

500 \</2ekernel
501 \<latexrelease>\IncludeInRelease{2015/01/01}%
502 \<latexrelease>           {\@}{Space after \@}%
503 \<*2ekernel | latexrelease>

```

```

504 \def\@{\spacefactor\@m{}}%
505 </2ekernel | latexrelease>
506 <latexrelease>\EndIncludeInRelease
507 <latexrelease>\IncludeInRelease{0000/00/00}%
508 <latexrelease>          {\@}{Space after \@}%
509 <latexrelease>\def\@{\spacefactor\@m{}}%
510 <latexrelease>\EndIncludeInRelease
511 <*2ekernel>

```

(End of definition for \@.)

\hspace

```
512 \DeclareRobustCommand\hspace{\@ifstar\@hspacer\@hspace}
```

(End of definition for \hspace.)

\@hspace

```

513 </2ekernel>
514 <*2ekernel | latexrelease>
515 <latexrelease>\IncludeInRelease{2020/10/01}%
516 <latexrelease>          {\@hspace}{Support calc with \hspace}%

```

We use a private register to calculate the space (if calc is used). Previously we used a group but that results in `\everypar` etc. being executed inside the group if the `\hspace` starts a paragraph. This is a bug fix so we do not provide rollback to the incorrect intermediate version.

```

517 \newskip\sp@ce@skip
518 \def\@hspace#1{\setlength\sp@ce@skip{#1}\hskip\sp@ce@skip}
519 </2ekernel | latexrelease>
520 <latexrelease>\EndIncludeInRelease
521 <latexrelease>\IncludeInRelease{0000/00/00}%
522 <latexrelease>          {\@hspace}{Support calc with \hspace}%
523
524 <latexrelease>
525 <latexrelease>\def\@hspace#1{\hskip #1\relax}
526 <latexrelease>\EndIncludeInRelease
527 <*2ekernel>

```

(End of definition for \@hspace.)

\@hspacer Extra `\hskip Opt` added 1985/17/12 to guard against a following `\unskip \relax` added 13 Oct 88 for usual T_EX lossage replaced both changes by `\hskip\z@skip` 27 Nov 91

```

528 \def\@hspacer#1{\vrule \width\z@\nobreak
529                \@hspace{#1}\hskip \z@skip}

```

(End of definition for \@hspacer.)

\fill

```

530 \newskip\fill
531 \fill = Opt plus 1fill

```

(End of definition for \fill.)

\stretch

```
532 \def\stretch#1{\z@ \@plus #1fill\relax}
```

(End of definition for `\stretch`.)

```
533 </2ekernel>
534 <*2ekernel | latexrelease>
535 <latexrelease>\IncludeInRelease{2018/12/01}%
536 <latexrelease>          {\thinspace}{Start LR-mode}%
```

`\enspace`

```
537 \DeclareRobustCommand\enspace{\leavevmode@ifvmode\kern.5em }
```

(End of definition for `\enspace`.)

`\leavevmode@ifvmode` Leave vmode but only if we are really in vmode, otherwise the expansion is empty (which is not the case with the default definition).

```
538 \protected\def\leavevmode@ifvmode{\ifvmode\expandafter\indent\fi}
```

(End of definition for `\leavevmode@ifvmode`.)

```
539 </2ekernel | latexrelease>
540 <latexrelease>\EndIncludeInRelease
541 <latexrelease>\IncludeInRelease{0000/00/00}%
542 <latexrelease>          {\thinspace}{Start LR-mode}%
543 <latexrelease>\def\thinspace{\kern .16667em }
544 <latexrelease>\def\negthinspace{\kern-.16667em }
545 <latexrelease>\def\enspace{\kern.5em }
546 <latexrelease>\let\leavevmode@ifvmode\@undefined
547 <latexrelease>\EndIncludeInRelease
548 <*2ekernel>
```

`\enskip`

`\quad`

`\qqquad`

```
549 \def\enskip{\hskip.5em\relax}
550 \def\quad{\hskip1em\relax}
551 \def\qqquad{\hskip2em\relax}
```

(End of definition for `\enskip`, `\quad`, and `\qqquad`.)

For Unicode engines, make the Unicode soft hyphen an active character defined as `\-`.

```
552 \ifx\Umathcode\@undefined\else
553   \catcode "AD=13
554   \def^^ad{\-}
555 \fi
```

`\obeycr` The following definitions will probably get deleted or moved to compatibility mode soon.
`\restorecr`

```
556 {\catcode'\^^M=13 \gdef\obeycr{\catcode'\^^M13 \def^^M{\relax}%
557   \@gobblecr}%
558 {\catcode'\^^M=13 \gdef\@gobblecr{\@ifnextchar
559   \@gobble\ignorespaces}}
560 \gdef\restorecr{\catcode'\^^M5 }}
```

(End of definition for `\obeycr` and `\restorecr`.)

```
561 </2ekernel>
```

File 19

ltlogos.dtx

1 Logos

Various logos are defined here.

`\TeX` The \TeX logo, adjusted so that a full stop after the logo counts as ending a sentence.

```
1 <*2kernel>
2 \DeclareRobustCommand\TeX{T\kern-.1667em\lower.5ex\hbox{E}\kern-.125emX\@}
```

(End of definition for \TeX.)

`\LaTeX` The \LaTeX logo.

```
3 \DeclareRobustCommand{\LaTeX}{L\kern-.36em%
4     {\sbox\z@ T%
5       \vbox to\ht\z@{\hbox{\check@mathfonts
6         \fontsize\sf@size\z@
7         \math@fontsfalse\selectfont
8         A}%
9         \vss}%
10      }%
11     \kern-.15em%
12     \TeX}
```

(End of definition for \LaTeX.)

`\LaTeXe` The $\LaTeX 2_{\epsilon}$ logo as proposed by A-W designers.

```
13 \DeclareRobustCommand{\LaTeXe}{\mbox{\m@th
14   \if b\expandafter\@car\f@series\@nil\boldmath\fi
15   \LaTeX\kern.15em2$_{\textstyle\varepsilon}$}}
16 </2kernel>
```

(End of definition for \LaTeXe.)

File 20

lfiles.dtx

1 File Handling

The following user commands are defined in this part:

<code>\document</code>	<code>(ie \begin{document})</code>	Reads in the .AUX files and <code>\catcode</code> 's @ to 12.
<code>\nofiles</code>		Suppresses all file output by setting <code>\@filesw</code> false.
<code>\includeonly</code>	<code>{\NAME1, ... ,NAMEn}</code>	Causes only parts NAME1, ... ,NAMEn to be read by their <code>\include</code> commands. Works by setting <code>partsw</code> true and setting <code>\@partlist</code> to NAME1, ... ,NAMEn.
<code>\include</code>	<code>{\NAME}</code>	Does an <code>\input</code> NAME unless <code>\@partsw</code> is true and NAME is not in <code>\@partlist</code> . If <code>\@filesw</code> is true, then it directs .AUX output to NAME.AUX, including a checkpoint at the end.
<code>\input</code>	<code>{\NAME}</code>	The same as TeX's <code>\input</code> , except it allows optional braces around the file name. In $\LaTeX 2_\epsilon$, it also avoids the primitive 'missing file' error, if the file can not be found.
<code>\IfFileExists</code>	<code>{\NAME}{\then}{\else}</code>	If the file exists on the system, execute <i>then</i> otherwise execute <i>else</i> .
<code>\InputIfFileExists</code>	<code>{\NAME}{\then}{\else}</code>	If the file exists on the system, execute <i>then</i> and input NAME otherwise execute <i>else</i> . <i>Historical $\LaTeX 2.09$ comments (not necessarily accurate any more):</i>

```

1 \*2kernel)
2 \message{files,}

```

VARIABLES, SWITCHES AND INTERNAL COMMANDS:

<code>\@mainaux</code>	:	Output file number for main .AUX file.
<code>\@partaux</code>	:	Output file number for current part's .AUX file.
<code>\@auxout</code>	:	Either <code>\@mainout</code> or <code>\@partout</code> , depending on which .AUX file output goes to.
<code>\@input{foo}</code>	:	If file foo exists, then <code>\input</code> 's it, otherwise types a warning message.
<code>@filesw</code>	:	Switch – set false if no .AUX, .TOC, .IDX etc files are to be written
<code>@partsw</code>	:	Set true by a <code>\includeonly</code> command.
<code>\@partlist</code>	:	Set to the argument of the <code>\includeonly</code> command.
<code>\cp@FOO</code>	:	The checkpoint for <code>\include</code> 'd file FOO.TEX, written by <code>\@writeckpt</code> at the end of file FOO.AUX

```

\includeonly{FILELIST} ==
BEGIN

```

```

\@partsw := T
\@partlist := FILELIST
END

\include{FILE} ==
BEGIN
\clearpage
if \@filesw = T
then \immediate\write\@mainaux{\string\@input{FILE.AUX}}
fi
if \@partsw = T
then \@tempswa := F
\reserved@b == FILE
for \reserved@a := \@partlist
do if eval(\reserved@a) = eval(\reserved@b)
then \@tempswa := T fi
od
fi

if \@tempswa = T
then \@auxout := \@partaux
if \@filesw = T
then \immediate\openout\@partaux{FILE.AUX}
\immediate\write\@partaux{\relax}
fi
\@input{FILE.TEX}
\clearpage
\@writeckpt{FILE}
if @filesw then \closeout \@partaux fi
\@auxout := \@mainaux
else \cp@FILE
fi
END

\@writeckpt{FILE} ==
BEGIN
if \@filesw = T
\immediate\write on file \@partaux:
\@setckpt{FILE}{
%% }
for \reserved@a := \cl@ckpt
do \immediate\write on file \@partaux:
\global\string\setcounter
{eval(\reserved@a)}{eval(\c@eval(\reserved@a))}
od
\immediate\write on file \@partaux: }
fi
END

\@setckpt{FILE}{LIST} ==
BEGIN

```

```
G \cp@FILE := LIST
END
```

```
INITIALIZATION
\@tempwa := T
```

End of historical L^AT_EX 2.09 comments.

```
\@mainaux
\@partaux 3 \newwrite\@mainaux
4 \newwrite\@partaux
```

(End of definition for \@mainaux and \@partaux.)

```
\if@filesw
\if@partsw 5 \newif\if@filesw \@fileswtrue
6 \newif\if@partsw \@partswfalse
```

(End of definition for \if@filesw and \if@partsw.)

\@clubpenalty This stores the current normal (non-infinite) value of `\clubpenalty`; it should therefore be reset whenever the normal value is changed (as in the bibliography in the standard styles).

```
7 \newcount\@clubpenalty
8 \@clubpenalty \@clubpenalty
```

(End of definition for \@clubpenalty.)

```
\document
```

```
9 \</2kernel)
10 \<latexrelease>\IncludeInRelease{2020/10/01}%
11 \<latexrelease> {\document}{Added hook to load l3backend code}%
12 \<*2kernel | latexrelease>
13 \def\document{%
```

We do cancel the grouping as part of the `\begin` handling (this is now done inside `\begin` instead) so that the `env/<env>/begin` hook is not hidden inside `\begingroup` ... `\endgroup`.

```
14 % \endgroup
15 \UseOneTimeHook{begindocument/before}%
16 \@kernel@after@begindocument@before
Added hook to load l3backend code:
17 \@expl@sys@load@backend@@
18 \ifx\@unusedoptionlist\@empty\else
19 \@latex@warning@no@line{Unused global option(s):^^J%
20 \spaces[\@unusedoptionlist]}%
21 \fi
22 \@colht\textheight
23 \@colroom\textheight \vsize\textheight
24 \@columnwidth\textwidth
25 \@clubpenalty\clubpenalty
26 \if@twocolumn
27 \advance\columnwidth -\columnsep
```

```

28   \divide\columnwidth\tw@ \hsize\columnwidth \@firstcolumntrue
29   \fi
30   \hsize\columnwidth \linewidth\hsize
31   \begingroup\@floatplacement\@dblfloatplacement
32   \makeatletter\let\@writefile\@gobbletwo

33   \global \let \@multiplelabels \relax
34   \input{\jobname.aux}%
35   \endgroup
36   \if@filesw
37   \immediate\openout\@mainaux\jobname.aux
38   \immediate\write\@mainaux{\relax}%
39   \fi

```

Dateline 1991/03/26: FMi added `\process@table` to support NFSS; This will also work with old fonts if no other style defines `\process@table`. The following line forces the initialization of the math fonts.

```

40   \process@table
41   \let\glb@currsiz@empty % Force math initialization.

42   \normalsize
43   \everypar{}%

```

So that punctuation in headings is not disturbed by verbatim or other local changes to the space factor codes, save the document default here. This will be locally reset by the output routine. For special cases a class may want to define `\normalsfcodes` directly, in case that definition will be used. (This is an old bug, problem existed in L^AT_EX 2.0x and plain T_EX.)

```

44   \ifx\normalsfcodes\@empty
45   \ifnum\sfcode'\.=\@m
46   \let\normalsfcodes\frenchspacing
47   \else
48   \let\normalsfcodes\nonfrenchspacing
49   \fi
50   \fi

```

For similar reasons also save the default language, this will be reset locally in the output routine. In particular it allows hyphenation in the page head even if the page break happens in verbatim. If this has already been set by a package, set to the value of `\language` at this point.

```

51   \ifx\document@default@language\m@ne
52   \chardef\document@default@language\language
53   \fi

```

Way back in 1991 (08/26) FMi & RmS set the `\@noskipsec` switch to true in the preamble and to false here. This was done to trap lists and related text in the preamble but it does not catch everything; hence Change 1.1g was introduced.

```

54   \@noskipsecfalse

55   \let \@refundefined \relax

```

Just before disabling the preamble commands we execute the begin document hook which contains any code contributed by `\AtBeginDocument`. Also disable the gathering of the file list, if no `\listfiles` has been issued. `\AtBeginDocument` is redefined at this point so that and such commands that get into the hook do not chase their tail...


```

56 \@kernel@before@begindocument
57 \UseOneTimeHook{begindocument}%
58 \@kernel@after@begindocument

```

Most of the following assignments will be done globally in case the user adds something like `\begin{multicols}` to the document hook, i.e. starts are group in `\begin{document}`.

Since a value of exactly 0pt for `\topskip` causes `\twocolumn[]` to misbehave, we add this check, hoping that it will not cause any problems elsewhere.

```

59 \ifdim\topskip<1sp\global\topskip 1sp\relax\fi
60 \global\maxdepth\maxdepth
61 \global\let\@begindocumenthook\@undefined
62 \ifx\@listfiles\@undefined
63   \global\let\@filelist\relax
64   \global\let\@addtofilelist\@gobble
65 \fi

```

At the very end we disable all preamble commands. This has to happen after the begin document hooks was executed so that this hook can still use such commands.

```

66 \gdef\do##1{\global\let ##1\@notprerr}%
67 \@preamblecmds

```

The next line saves tokens and also allows `\@nodocument` to be used directly to trap preamble errors.

```

68 \global\let \@nodocument \relax

```

The next line is a pure safety measure in case a do list is ever expanded at the wrong place. In addition it will save a few tokens to get rid of the above definition.

```

69 \global\let\do\noexpand
70 \UseOneTimeHook{begindocument/end}%

```

Use of the hook might mean that we are already in horizontal mode, so ignore the space after `\begin{document}`.

```

71 \ignorespaces}

```

Provide a global definition for `\do` as well, so that it is already defined in the preamble and not late as `\begin{document}` overwriting some definition given by the unsuspecting user in the preamble.

```

72 \let\do\noexpand

```

The `begindocument` hook already existed in the kernel since 1994 under the name `\atbegindocumenthook` the additional ones are originally from the `etoolbox` package under the names `\@endpreamblehook` `\afterpreamble`.

```

73 \NewHook{begindocument}
74 \NewHook{begindocument/before}
75 \NewHook{begindocument/end}

```

Above we used two kernel only hooks to be run after the public `begindocument/before` and after `begindocument` hooks.

In `\@kernel@after@begindocument@before` we already place one action: drop the fast execution code for the `env/document/begin` hook. That hook marks the end of the preamble and should therefore only be run once. In a normal document that is anyway the case (so the code would just sit there taking up space afterwards, which these days is rather harmless), however, in more complicated scenarios where several full documents are combined to a single document it might get applied several times with harmful effects.

```

\@kernel@after@begindocument@before
\@kernel@before@begindocument
\@kernel@after@begindocument

```

We therefore explicitly drop it at this point. the coding is somewhat obscure due to the name of the macro which requires constructing.

```

76 \edef \@kernel@after@begindocument@before {%
77   \let\expandafter\noexpand\csname
78     __hook env/document/begin\endcsname
79   \noexpand\@empty}

```

These internal hooks are already declared earlier (in `ltxpl`) so that other modules could write to them.

```

80 %\let \@kernel@before@begindocument \@empty
81 %\let \@kernel@after@begindocument \@empty

82 </2kernel | latexrelease>
83 <latexrelease>\EndIncludeInRelease

84 <latexrelease>\IncludeInRelease{2017/04/15}%
85 <latexrelease> {\document}{Save language for hyphenation}%
86 <latexrelease>
87 <latexrelease>\def\document{\endgroup
88 <latexrelease> \ifx\@unusedoptionlist\@empty\else
89 <latexrelease>   \@latex@warning@no@line{Unused global option(s):^^J%
90 <latexrelease>     \spaces[\@unusedoptionlist]}%
91 <latexrelease> \fi
92 <latexrelease> \@colht\textheight
93 <latexrelease> \@colroom\textheight \vsize\textheight
94 <latexrelease> \@columnwidth\textwidth
95 <latexrelease> \@clubpenalty\clubpenalty
96 <latexrelease> \if@twocolumn
97 <latexrelease>   \advance\columnwidth -\columnsep
98 <latexrelease>   \divide\columnwidth\tw@ \hsize\columnwidth \@firstcolumntrue
99 <latexrelease> \fi
100 <latexrelease> \hsize\columnwidth \linewidth\hsize
101 <latexrelease> \begingroup\@floatplacement\@dblfloatplacement
102 <latexrelease>   \makeatletter\let\@writefile\@gobbletwo
103 <latexrelease>   \global \let \@multiplelabels \relax
104 <latexrelease>   \@input{\jobname.aux}%
105 <latexrelease> \endgroup
106 <latexrelease> \if@filesw
107 <latexrelease>   \immediate\openout\@mainaux\jobname.aux
108 <latexrelease>   \immediate\write\@mainaux{\relax}%
109 <latexrelease> \fi
110 <latexrelease> \process@table
111 <latexrelease> \let\glb@currsize\@empty % Force math initialization.
112 <latexrelease> \normalsize
113 <latexrelease> \everypar{}%
114 <latexrelease> \ifx\normalsfcodes\@empty
115 <latexrelease>   \ifnum\sfcode'\.=\@m
116 <latexrelease>     \let\normalsfcodes\frenchspacing
117 <latexrelease>   \else
118 <latexrelease>     \let\normalsfcodes\nonfrenchspacing
119 <latexrelease>   \fi
120 <latexrelease> \fi
121 <latexrelease> \ifx\document@default@language@m@ne
122 <latexrelease>   \chardef\document@default@language\language
123 <latexrelease> \fi

```

```

124 <latexrelease> \@noskipsecfalse
125 <latexrelease> \let \@refundefined \relax
126 <latexrelease> \let \@AtBeginDocument \@firstofone
127 <latexrelease> \@begindocumenthook
128 <latexrelease> \ifdim\topskip<1sp\global\topskip 1sp\relax\fi
129 <latexrelease> \global\@maxdepth\maxdepth
130 <latexrelease> \global\let \@begindocumenthook \@undefined
131 <latexrelease> \ifx \@listfiles \@undefined
132 <latexrelease> \global\let \@filelist \relax
133 <latexrelease> \global\let \@addtofilelist \@gobble
134 <latexrelease> \fi
135 <latexrelease> \gdef\do##1{\global\let ##1\@notprerr}%
136 <latexrelease> \@preamblecmds
137 <latexrelease> \global\let \@nodocument \relax
138 <latexrelease> \global\let \do\noexpand
139 <latexrelease> \ignorespaces}
140 <latexrelease> \EndIncludeInRelease
141 <latexrelease>
142 <latexrelease> \IncludeInRelease{0000/00/00}%
143 <latexrelease> {\document}{Save language for hyphenation}
144 <latexrelease> \def \document {\endgroup
145 <latexrelease> \ifx \@unusedoptionlist \@empty \else
146 <latexrelease> \latex@warning@no@line{Unused global option(s): ^^J%
147 <latexrelease> \spaces [\@unusedoptionlist]}}%
148 <latexrelease> \fi
149 <latexrelease> \@colht\textheight
150 <latexrelease> \@colroom\textheight \vsize\textheight
151 <latexrelease> \columnwidth\textwidth
152 <latexrelease> \@clubpenalty\clubpenalty
153 <latexrelease> \if@twocolumn
154 <latexrelease> \advance\columnwidth -\columnsep
155 <latexrelease> \divide\columnwidth\tw@ \hsize\columnwidth
156 <latexrelease> \@firstcolumntrue
157 <latexrelease> \fi
158 <latexrelease> \hsize\columnwidth \linewidth\hsize
159 <latexrelease> \begingroup \@floatplacement \@dblfloatplacement
160 <latexrelease> \makeatletter \let \@writefile \@gobbletwo
161 <latexrelease> \global \let \@multiplelabels \relax
162 <latexrelease> \@input{\jobname.aux}%
163 <latexrelease> \endgroup
164 <latexrelease> \if@filesw
165 <latexrelease> \immediate \openout \@mainaux \jobname.aux
166 <latexrelease> \immediate \write \@mainaux {\relax}%
167 <latexrelease> \fi
168 <latexrelease> \process@table
169 <latexrelease> \let \glb@currsize \@empty
170 <latexrelease> \normalsize
171 <latexrelease> \everypar{}%
172 <latexrelease> \ifx \normalsfcodes \@empty
173 <latexrelease> \ifnum \sfcode' \. = \@m
174 <latexrelease> \let \normalsfcodes \frenchspacing
175 <latexrelease> \else
176 <latexrelease> \let \normalsfcodes \nonfrenchspacing
177 <latexrelease> \fi

```

```

178 <latexrelease> \fi
179 <latexrelease> \@noskipsecfalse
180 <latexrelease> \let \@refundefined \relax
181 <latexrelease> \let \@AtBeginDocument \@firstofone
182 <latexrelease> \@begindocumenthook
183 <latexrelease> \ifdim\topskip<1sp\global\topskip 1sp\relax\fi
184 <latexrelease> \global\@maxdepth\maxdepth
185 <latexrelease> \global\let \@begindocumenthook \@undefined
186 <latexrelease> \ifx \@listfiles \@undefined
187 <latexrelease> \global\let \@filelist \relax
188 <latexrelease> \global\let \@addtofilelist \@gobble
189 <latexrelease> \fi
190 <latexrelease> \gdef\do##1{\global\let ##1\@notprerr}%
191 <latexrelease> \@preamblecmds
192 <latexrelease> \global\let \@nodocument \relax
193 <latexrelease> \global\let\do\noexpand
194 <latexrelease> \ignorespaces}
195 <latexrelease>\EndIncludeInRelease
196 <*2ekernel>

197 \@onlypreamble\document

```

(End of definition for \document and others.)

\normalsfcodes The setting of `\@empty` is just a flag. This command may be defined in a class or package file. If it is still `\@empty` at `\begin{document}` it will be defined to be `\frenchspacing` or `\nonfrenchspacing`, depending on which of those appears to be in effect at that point.

```
198 \let\normalsfcodes\@empty
```

(End of definition for \normalsfcodes.)

\nofiles Set `\@fileswfalse` which suppresses the places where L^AT_EX makes `\immediate` writes. The `\makeindex` and `\makeglossary` are disabled. `\protected@write` is redefined not to write to the file specified, but rather to write a blank line to the log file. This ensures that a *<whatsit>* node is still created, and so spacing is not affected by the `\nofiles` command; to ensure this more generally, the `\if@nobreak` test is needed.

```

199 \def\nofiles{%
200   \@fileswfalse
201   \typeout{No auxiliary output files.^^J}%
202   \long\def\protected@write##1##2##3%
203     {\write\m@ne{ }\if@nobreak\ifvmode\nobreak\fi\fi}%
204   \let\makeindex\relax
205   \let\makeglossary\relax}
206 \@onlypreamble\nofiles

```

(End of definition for \nofiles.)

\protected@write This takes three arguments: an output stream, some initialization code, and some text to write. It then writes this, with appropriate handling of `\protect` and `\thepage`.

```

207 \long\def \protected@write#1#2#3{%
208   \begingroup
209   \let\thepage\relax
210   #2%
211   \let\protect\@unexpandable@protect
212   \edef\reserved@a{\write#1{#3}}%

```

```

213     \reserved@a
214     \endgroup
215     \if@nobeak\ifvmode\nobeak\fi\fi
216 }

```

(End of definition for \protected@write.)

```

217 \let\@auxout=\@mainaux

```

`\include` In the definition of `\include`, `\def\reserved@b` changed to `\edef\reserved@b` to be consistent with the `\edef` in `\includeonly`. (Suggested by Rainer Schöpf & Frank Mittelbach. Change made 20 Jul 88.)

Changed definition of `\include` to allow space at end of file name — otherwise, typing `\include{foo }` would cause L^AT_EX to overwrite `foo.tex`. Change made 24 May 89, suggested by Rainer Schöpf and Frank Mittelbach

Made `\include` check for being used inside an `\include`'d file, as this will not work and cause surprising results.

```

218 </2ekernel>
219 <*2ekernel | latexrelease>
220 <latexrelease>\IncludeInRelease{2020/10/01}%
221 <latexrelease>          {\includeonly}{Spaces in file names}%
222 \def\include#1{\relax
223   \ifnum\@auxout=\@partaux
224     \latex@error{\string\include\space cannot be nested}\@eha
225   \else

```

Here the normalization will add `.tex` for all files, (it uses the same normalization as the hooks), so we need to remove that manually. `\@strip@tex@ext` does that.

```

226   \set@curr@file{#1}%
227   \edef\@curr@file{\@strip@tex@ext\@curr@file}%

```

For historical reasons `\@include` expects an argument delimited by a space. This is kept (though unnecessary now) to avoid errors in other packages that use `\@include` directly.

```

228   \expandafter\@include\expandafter{\@curr@file} % deliberate space
229   \fi}

```

Here in `\includeonly` we also need to strip `.tex` after normalization:

```

230 \def\includeonly#1{%
231   \@partswtrue

```

Because the argument to `\includeonly` is a comma-separated list of filenames where there may be comma's preceding some of the filenames or trailing them. Therefore we need to take the list apart, remove the unwanted spaces while leaving the spaces *in* the filenames intact.

```

232   \let\@partlist\@empty
233   \@for\reserved@a:=#1 \do
234     {%
235       \expandafter\set@curr@file\expandafter{\reserved@a}%
236       \ifx\@partlist\@empty
237         \edef\@partlist{\@strip@tex@ext\@curr@file}%
238       \else
239         \edef\@partlist{\@partlist,\@strip@tex@ext\@curr@file}%
240       \fi
241     }%
242   }
243   \@onlypreamble\includeonly

```

(End of definition for \include and \includeonly.)

\@strip@tex@ext These macros take a (\detokenized file name and remove any .tex extension). Extra
\@strip@tex@ext@aux care is taken to not remove the string .tex from the middle of a file name: it is only removed if it's the very last thing in the file name.

```
244 \def\reserved@a#1{%
245 \def\@strip@tex@ext##1{%
246   \expandafter\@strip@tex@ext@aux
247     ##1\@nil\@nil
248     #1\@nil\relax\@nnil}
249 \def\@strip@tex@ext@aux##1#1\@nil##2\@nnil{%
250   \ifx\relax##2\@empty
251     \expandafter\@cdr\expandafter\@empty\@cdr{}\##1%
252   \else##1\fi}}%
253 \expandafter\reserved@a
254 \expandafter{\detokenize{.tex}}
255 /2kernel | latexrelease)
```

(End of definition for \@strip@tex@ext and \@strip@tex@ext@aux.)

```
256 <latexrelease>\EndIncludeInRelease
257 <latexrelease>\IncludeInRelease{2019/10/01}%
258 <latexrelease>           {\includeonly}{Spaces in file names}%
259 <latexrelease>
260 <latexrelease>\def\includeonly#1{%
261 <latexrelease>  \@partswtrue
262 <latexrelease>  \set@curr@file{\zap@space#1 \@empty}%
263 <latexrelease>  \let\@partlist\@curr@file
264 <latexrelease>  }
265 <latexrelease>
266 <latexrelease>\def\include#1{\relax
267 <latexrelease>  \ifnum\@auxout=\@partaux
268 <latexrelease>    \@latex@error{\string\include\space cannot be nested}\@eha
269 <latexrelease>  \else
270 <latexrelease>    \set@curr@file{#1 }%
271 <latexrelease>    \expandafter\@include\@curr@file
272 <latexrelease>  \fi}
273 <latexrelease>
274 <latexrelease>\let\@strip@tex@ext\@undefined
275 <latexrelease>\let\@strip@tex@ext@aux\@undefined
276 <latexrelease>
277 <latexrelease>\EndIncludeInRelease
278 <latexrelease>\IncludeInRelease{0000/00/00}%
279 <latexrelease>           {\includeonly}{Spaces in file names}%
280 <latexrelease>\def\includeonly#1{%
281 <latexrelease>  \@partswtrue
282 <latexrelease>  \edef\@partlist{\zap@space#1 \@empty}}
283 <latexrelease>
284 <latexrelease>\def\include#1{\relax
285 <latexrelease>  \ifnum\@auxout=\@partaux
286 <latexrelease>    \@latex@error{\string\include\space cannot be nested}\@eha
287 <latexrelease>  \else \@include#1 \fi}
288 <latexrelease>
289 <latexrelease>\EndIncludeInRelease
290 <*2kernel>
```

`\@include`

```
291 </2kernel>
292 <*2kernel | latexrelease>
293 <latexrelease> \IncludeInRelease{2022/06/01}%
294 <latexrelease>           {\@include}{Spaces in file names and hooks}%

295 \def\@include#1 {%
296 \ifx\nodocument\relax
297   \clearpage
298   \if@filesw
299     \immediate\write\@mainaux{\string\@input{#1.aux}}%
300   \fi
301   \@tempwatrue
302   \if@partsw
303     \@tempwafalse
304     \edef\reserved@b{#1}%
305     \@for\reserved@a:=\@partlist\do
306       {\ifx\reserved@a\reserved@b\@tempwatrue\fi}%
307   \fi
308   \if@tempswa
309     \let\@auxout\@partaux
310     \if@filesw
311       \immediate\openout\@partaux "#1.aux"
312       \immediate\write\@partaux{\relax}%
313     \fi
```

Now before going to the hooks we need to set `\CurrentFile`:

```
314 %-----
315   \@filehook@set@CurrentFile
```

Execute the before hooks just after we switched the `.aux` file ...

```
316   \UseHook{include/before}%
317   \UseOneTimeHook{include/#1/before}%
318 %-----
319   \@input@{#1.tex}%
320 %-----
```

... then end hooks ...

```
321   \UseOneTimeHook{include/#1/end}%
322   \UseHook{include/end}%
323 %-----
324   \clearpage
325 %-----
```

... and after the `\clearpage` the after hooks followed by another `\clearpage` just in case new material got added (after all we need to be in well defined state after the `\include`).

```
326   \UseOneTimeHook{include/#1/after}%
327   \UseHook{include/after}%
328   \clearpage
329 %-----
330   \@writeckpt{#1}%
331   \if@filesw
332     \immediate\closeout\@partaux
```

```

333 \fi
334 \else

```

If the file is not included, reset `\deadcycles`, so that a long list of non-included files does not generate an ‘Output loop’ error.

```

335 \deadcycles\z@
336 \@nameuse{cp@#1}%

```

We also execute a hook in this case, first a general used for every include file that is exclude and then a specific one that contains the name of the include file.

```

337 %-----
338 \UseHook{include/excluded}%
339 \UseOneTimeHook{include/#1/excluded}%
340 %-----
341 \fi
342 \let\@auxout\@mainaux
343 \else
344 \@latex@warning{%
345 \noexpand\include should only be used after \string\begin{document}}%
346 \@input@{#1}%
347 \fi}

```

Now declare the non-generic include hooks used above:

```

348 \NewHook{include/before}
349 \NewReversedHook{include/end}
350 \NewReversedHook{include/after}
351 \NewHook{include/excluded}
352 <latexrelease>\EndIncludeInRelease
353 </2ekernel | latexrelease>
354 <latexrelease>\IncludeInRelease{2020/10/01}%
355 <latexrelease> \{\@include}{Spaces in file names and hooks}%
356 <latexrelease>\EndIncludeInRelease
357 <latexrelease>\def\@include#1 {%
358 <latexrelease>\ifx\nodocument\relax
359 <latexrelease> \clearpage
360 <latexrelease> \if@filesw
361 <latexrelease> \immediate\write\@mainaux{\string\@input{#1.aux}}%
362 <latexrelease> \fi
363 <latexrelease> \@tempwatrue
364 <latexrelease> \if@partsw
365 <latexrelease> \@tempwafalse
366 <latexrelease> \edef\reserved@b{#1}%
367 <latexrelease> \@for\reserved@a:=\@partlist\do
368 <latexrelease> {\ifx\reserved@a\reserved@b\@tempwatrue\fi}%
369 <latexrelease> \fi
370 <latexrelease> \if@tempswa
371 <latexrelease> \let\@auxout\@partaux
372 <latexrelease> \if@filesw
373 <latexrelease> \immediate\openout\@partaux "#1.aux"
374 <latexrelease> \immediate\write\@partaux{\relax}%
375 <latexrelease> \fi
376 <latexrelease> \@filehook@set@CurrentFile
377 <latexrelease> \UseHook{include/before}%
378 <latexrelease> \UseOneTimeHook{include/#1/before}%

```



```

379 <latexrelease> \input@{#1.tex}%
380 <latexrelease> \UseOneTimeHook{include/#1/end}%
381 <latexrelease> \UseHook{include/end}%
382 <latexrelease> \clearpage
383 <latexrelease> \UseOneTimeHook{include/#1/after}%
384 <latexrelease> \UseHook{include/after}%
385 <latexrelease> \clearpage
386 <latexrelease> \@writeckpt{#1}%
387 <latexrelease> \if@filesw
388 <latexrelease> \immediate\closeout\@partaux
389 <latexrelease> \fi
390 <latexrelease> \else
391 <latexrelease> \deadcycles\z@
392 <latexrelease> \@nameuse{cp@#1}%
393 <latexrelease> \fi
394 <latexrelease> \let\@auxout\@mainaux
395 <latexrelease> \else
396 <latexrelease> \@latex@warning{%
397 <latexrelease> \noexpand\include should only be used after \string\begin{document}}%
398 <latexrelease> \@input@{#1}%
399 <latexrelease> \fi
400 <latexrelease> \NewHook{include/before}
401 <latexrelease> \NewReversedHook{include/end}
402 <latexrelease> \NewReversedHook{include/after}

403 <latexrelease> \IncludeInRelease{0000/00/00}%
404 <latexrelease> \include\@include\{Spaces in file names and hooks}%
405 <latexrelease> \def\@include#1 {%
406 <latexrelease> \clearpage
407 <latexrelease> \if@filesw
408 <latexrelease> \immediate\write\@mainaux{\string\@input{#1.aux}}%
409 <latexrelease> \fi
410 <latexrelease> \@tempwatrue
411 <latexrelease> \if@partsw
412 <latexrelease> \@tempwafalse
413 <latexrelease> \edef\reserved@b{#1}%
414 <latexrelease> \@for\reserved@a:=\@partlist\do
415 <latexrelease> {\ifx\reserved@a\reserved@b\@tempwatrue\fi}%
416 <latexrelease> \fi
417 <latexrelease> \if@tempswa
418 <latexrelease> \let\@auxout\@partaux
419 <latexrelease> \if@filesw
420 <latexrelease> \immediate\openout\@partaux #1.aux
421 <latexrelease> \immediate\write\@partaux{\relax}%
422 <latexrelease> \fi
423 <latexrelease> \input@{#1.tex}%
424 <latexrelease> \clearpage
425 <latexrelease> \@writeckpt{#1}%
426 <latexrelease> \if@filesw
427 <latexrelease> \immediate\closeout\@partaux
428 <latexrelease> \fi
429 <latexrelease> \else
430 <latexrelease> \deadcycles\z@
431 <latexrelease> \@nameuse{cp@#1}%
432 <latexrelease> \fi

```

```

433 <latexrelease> \let\@auxout\@mainaux}
434 <latexrelease>
435 <latexrelease>\EndIncludeInRelease
436 <*2kernel>

```

(End of definition for \@include.)

`\@writeckpt`

```

437 \def\@writeckpt#1{%
438   \if@filesw
439     \immediate\write\@partaux{\string\@setckpt{#1}\@charlb}%
440     {\let\@elt\@wckptelt \cl@ckpt}%
441     \immediate\write\@partaux{\@charrb}%
442   \fi}

```

(End of definition for \@writeckpt.)

`\@wckptelt`

```

443 \def\@wckptelt#1{%
444   \immediate\write\@partaux{%
445     \string\setcounter{#1}{\the\@nameuse{c@#1}}}

```

(End of definition for \@wckptelt.)

`\@setckpt` RmS 93/08/31: introduced `\@setckpt`

```

446 \def\@setckpt#1{\global\@namedef{cp@#1}}

```

(End of definition for \@setckpt.)

`\@charlb` The following defines `\@charlb` and `\@charrb` to be { and }, respectively with `\catcode`
`\@charrb` 11.

```

447 {\catcode' [=1 \catcode'] =2
448 \catcode' { =11 \catcode'} =11
449 \gdef\@charlb[{
450 \gdef\@charrb[}]
451 ]% }brace matching

```

(End of definition for \@charlb and \@charrb.)

1.1 Safe Input Macros

`\@curr@file` File name handling is done by generating a csname from the provided file name (which
`\set@curr@file` means that UTF-8 octets gets turned into strings as this is what happens if they appear in a csname due to the code in `utf8.def`). By setting `\escapchar` to -1 we ensure that we don't get a backslash in front. As a result we end up with all characters as catcode 12 (plus spaces). We then sometimes add quotes around the construct (removing any existing inner quotes). Sometimes we only remove the quotes if they have been supplied by the user. There is clearly some room for improvement.

A side effect of the new code is that we will see quotes around file name displays where there haven't been any before.

For compatibility with existing code using `{abc}.tex` or `{one.two}.png`, an initial brace group is discarded before expansion and `\string` is applied. The content of the brace group is discarded. This means that a leading space will be lost unless protected (by `{ }` or `" "` or `\space`) but filenames with a space are hopefully rare.

The definition below is from 2019 and only used during kernel bootstrapping, later on in `ltfilehook.dtx` it will get overwritten.

```

452 \def\set@curr@file#1{%
453   \begingroup
454     \escapechar\m@ne
455     \xdef\@curr@file{%
456       \expandafter\expandafter\expandafter\unquote@name
457       \expandafter\expandafter\expandafter{%
458         \expandafter\string
459         \csname\@firstofone#1\@empty\endcsname}}%
460   \endgroup
461 }

```

(End of definition for \@curr@file and \set@curr@file.)

```

\quote@name   Quoting spaces
\quote@@name  a b c    -> "a b c"
\unquote@name "a b c"  -> "a b c"
              a" "b" "c -> "a b c"
              -> ""

```

```

462 </2kernel>
463 <*2kernel | latexrelease>
464 <latexrelease>\IncludeInRelease{2019/10/01}%
465 <latexrelease>           {\quote@name}{Quote file names}%
466 \def\quote@name#1{"\quote@@name#1@gobble"}
467 \def\quote@@name#1{#1\quote@@name}

```

and removing quotes ...

```

468 \def\unquote@name#1{\quote@@name#1@gobble}

```

(End of definition for \quote@name, \quote@@name, and \unquote@name.)

\IfFileExists

```

469 \DeclareRobustCommand\IfFileExists[1]{%
470   \set@curr@file{#1}%
471   \expandafter\IfFileExists@\expandafter{\@curr@file}}

```

(End of definition for \IfFileExists.)

```

472 </2kernel | latexrelease>
473 <latexrelease>\EndIncludeInRelease
474 <latexrelease>\IncludeInRelease{0000/00/00}%
475 <latexrelease>           {\quote@name}{Quote file names}%
476 <latexrelease>
477 <latexrelease>\let\quote@name\@undefined
478 <latexrelease>\let\quote@@name\@undefined
479 <latexrelease>\let\unquote@name\@undefined
480 <latexrelease>
481 <latexrelease>\long\def \IfFileExists#1#2#3{%
482 <latexrelease> \openin\@inputcheck#1 %
483 <latexrelease> \ifeof\@inputcheck
484 <latexrelease> \ifx\input@path\@undefined
485 <latexrelease> \def\reserved@a{#3}%
486 <latexrelease> \else

```

```

487 <latexrelease> \def\reserved@a{\@iffileonpath{#1}{#2}{#3}}%
488 <latexrelease> \fi
489 <latexrelease> \else
490 <latexrelease> \closein\@inputcheck
491 <latexrelease> \edef\@filef@und{#1 }%
492 <latexrelease> \def\reserved@a{#2}%
493 <latexrelease> \fi
494 <latexrelease> \reserved@a}
495 <latexrelease>
496 <latexrelease>\EndIncludeInRelease
497 <*2ekernel)

```

`\IfFileExists@` Argument #1 is `\@curr@file` so catcode 12 string with no quotes.

`\IfFileExists@@` The original definition picked up arguments #2 and #3 in a way that they couldn't contain unbalanced conditionals. A better implementation would have been not to pick up the arguments at all but instead use the usual `\@firstoftwo` and `\secondoftwo`. However, that changes how # is interpreted and so we can't do that nowadays without invalidating a lot of code. Therefore the somewhat curious construction near the end.

Earlier versions used `\openin` here, but this led to two code paths, one in `expl3` and one here. To avoid that, and as the `expl3` approach works by expansion, we use that here. As we need the file name to include the path, the actual `expl3` function used is not the file existence test!

```

498 </2ekernel)
499 <*2ekernel | latexrelease)
500 <latexrelease>\IncludeInRelease{2023/06/01}%
501 <latexrelease> \IfFileExists@{\IfFileExists}
502 \long\def \IfFileExists@#1#2#3{%
503 \edef\@filef@und{\IfFileExists@@{#1}}%

```

The `expl3` function regards an empty argument as nothing at all, but the L^AT_EX 2_ε convention is that this is equal to the special `.tex` file.

```

504 \ifx\@filef@und\@empty
505 \if\relax\detokenize{#1}\relax
506 \let\reserved@a\@firstoftwo
507 \def\@filef@und{".tex" }%
508 \else
509 \let\reserved@a\@secondoftwo
510 \fi
511 \else
512 \let\reserved@a\@firstoftwo
513 \edef\@filef@und{"\@filef@und" }%
514 \fi

```

This is just there so that any # inside #2 or #3 needs doubling (as that was the case in the past).

```

515 \expandafter\def\expandafter\reserved@a
516 \expandafter{\reserved@a{#2}{#3}}%
517 \reserved@a}

```

Pipes are not really files, but at the document level they are supported. To quickly trim of any leading spaces, we use a blank test and `\use:n` rather than `\t1_trim_spaces:n` for speed as we don't care about the end of the input.

```

518 \ExplSyntaxOn
519 \cs_new:Npn \IfFileExists@@ #1

```

```

520 {
521   \tl_if_blank:nF {#1}
522   {
523     \tl_if_head_eq_charcode:oNTF { \use:n #1 } |
524     {#1}
525     { \file_full_name:n {#1} }
526   }
527 }
528 \cs_generate_variant:Nn \tl_if_head_eq_charcode:nNTF { o }
529 \ExplSyntaxOff
530 </2ekernel|latexrelease>
531 <latexrelease>\EndIncludeInRelease
532 <latexrelease>\IncludeInRelease{2021/06/01}%
533 <latexrelease>      {\IfFileExists@}{IfFileExists}
534 <latexrelease>
535 <latexrelease>\long\def \IfFileExists@#1#2#3{%
536 <latexrelease>  \openin\@inputcheck"#1" %
537 <latexrelease>  \ifeof\@inputcheck
538 <latexrelease>    \ifx\input@path\@undefined
539 <latexrelease>      \let\reserved@a\@secondoftwo
540 <latexrelease>    \else
541 <latexrelease>      \def\reserved@a{\@iffileonpath{#1}}%
542 <latexrelease>    \fi
543 <latexrelease>  \else
544 <latexrelease>    \closein\@inputcheck
545 <latexrelease>    \edef\@filef@und{"#1" }%
546 <latexrelease>    \let\reserved@a\@firstoftwo
547 <latexrelease>  \fi
548 <latexrelease>  \expandafter\def\expandafter\reserved@a
549 <latexrelease>    \expandafter{\reserved@a{#2}{#3}}%
550 <latexrelease>\reserved@a}
551 <latexrelease>\let\IfFileExists@@\@undefined
552 <latexrelease>\EndIncludeInRelease
553 <latexrelease>
554 <latexrelease>\IncludeInRelease{2019/10/01}%
555 <latexrelease>      {\IfFileExists@}{IfFileExists}
556 <latexrelease>
557 <latexrelease>\long\def \IfFileExists@#1#2#3{%
558 <latexrelease>  \openin\@inputcheck"#1" %
559 <latexrelease>  \ifeof\@inputcheck
560 <latexrelease>    \ifx\input@path\@undefined
561 <latexrelease>      \def\reserved@a{#3}%
562 <latexrelease>    \else
563 <latexrelease>      \def\reserved@a{\@iffileonpath{#1}{#2}{#3}}%
564 <latexrelease>    \fi
565 <latexrelease>  \else
566 <latexrelease>    \closein\@inputcheck
567 <latexrelease>    \edef\@filef@und{"#1" }%
568 <latexrelease>    \def\reserved@a{#2}%
569 <latexrelease>  \fi
570 <latexrelease>  \reserved@a}
571 <latexrelease>\EndIncludeInRelease
572 <latexrelease>\IncludeInRelease{0000/00/00}%
573 <latexrelease>      {\IfFileExists@}{IfFileExists}

```

```

574 <latexrelease>
575 <latexrelease>\let\IfFileExists@\@undefined
576 <latexrelease>
577 <latexrelease>
578 <latexrelease>\EndIncludeInRelease
579 <*2ekernel>

(End of definition for \IfFileExists@ and \IfFileExists@.)

```

\@iffileonpath If the file is not found by `\openin`, and `\input@path` is defined, look in all the directories specified in `\input@path`.

```

580 </2ekernel>
581 <*2ekernel | latexrelease>
582 <latexrelease>\IncludeInRelease{2019/10/01}%
583 <latexrelease>          {\@iffileonpath}{Quote file names}
584 \long\def\@iffileonpath#1{%
585   \let\reserved@a\@secondoftwo
586   \expandafter\@tfor\expandafter\reserved@b\expandafter
587     :\expandafter=\input@path\do{%
588     \openin\@inputcheck\expandafter\quote@name\expandafter{\reserved@b#1} %
589     \ifeof\@inputcheck\else
590     \edef\@filef@und{\expandafter\quote@name\expandafter{\reserved@b#1} }%
591     \let\reserved@a\@firstoftwo%
592     \closein\@inputcheck
593     \@break@tfor
594     \fi}%
595   \reserved@a}

(End of definition for \@iffileonpath.)

```

```

596 </2ekernel | latexrelease>
597 <latexrelease>\EndIncludeInRelease
598 <latexrelease>\IncludeInRelease{0000/00/00}%
599 <latexrelease>          {\quote@name}{Quote file names}
600 <latexrelease>
601 <latexrelease>\long\def\@iffileonpath#1{%
602 <latexrelease>   \let\reserved@a\@secondoftwo
603 <latexrelease>   \expandafter\@tfor\expandafter\reserved@b\expandafter
604 <latexrelease>     :\expandafter=\input@path\do{%
605 <latexrelease>     \openin\@inputcheck\reserved@b#1 %
606 <latexrelease>     \ifeof\@inputcheck\else
607 <latexrelease>     \edef\@filef@und{\reserved@b#1 }%
608 <latexrelease>     \let\reserved@a\@firstoftwo%
609 <latexrelease>     \closein\@inputcheck
610 <latexrelease>     \@break@tfor
611 <latexrelease>     \fi}%
612 <latexrelease>   \reserved@a}
613 <latexrelease>
614 <latexrelease>\EndIncludeInRelease
615 <*2ekernel>

```

\InputIfFileExists Now define `\InputIfFileExists` to input #1 if it seems to exist. Immediately prior to the input, #2 is executed. If the file #1 does not exist, execute ‘#3’.

This here is a temporary definition for the kernel. The real one comes somewhat later in the file `lfilehook.dtx`.

```

616 \DeclareRobustCommand \InputIfFileExists[2]{%
617   \IfFileExists{#1}%
618   {%
619   \expandafter\@swaptwoargs\expandafter
620     {\filef@und}{#2\@addtofilelist{#1}\@input}}

```

(End of definition for \InputIfFileExists.)

\@swaptwoargs Swap two arguments and return them unbraced (like \@firstoftwo etc).

```

621 \</2kernel>
622 \<*2kernel | latexrelease>
623 \<latexrelease> \IncludeInRelease{2019/10/01}%
624 \<latexrelease> {\@swaptwoargs}{Don't lose the file name}%
625 \long\def\@swaptwoargs#1#2{#2#1}
626 \</2kernel | latexrelease>
627 \<latexrelease> \EndIncludeInRelease
628 \<latexrelease> \IncludeInRelease{0000/00/00}%
629 \<latexrelease> {\@swaptwoargs}{Don't lose the file name}%
630 \<latexrelease> \let\@swaptwoargs\@undefined
631 \<latexrelease> \EndIncludeInRelease
632 \<*2kernel>

```

(End of definition for \@swaptwoargs.)

\input Input a file: if the argument is given in braces use safe input macros, otherwise use T_EX's primitive \input command (which is called \@input in L^AT_EX).

```

633 \def\input{\@ifnextchar\bgroup\iinput\@input}

```

(End of definition for \input.)

\@iinput Define \@iinput (i.e., \input) in terms of \InputIfFileExists.

Changes to \@iinput: adapt to the changes to \@missingfileerror.

```

634 \</2kernel>
635 \<*2kernel | latexrelease>
636 \<latexrelease> \IncludeInRelease{2020/10/01}%
637 \<latexrelease> {\@iinput}{Change in file error handling}%
638 \def\@iinput#1{%
639   \InputIfFileExists{#1}{%
640     {\filename@parse\curr@file
641       \edef\reserved@a{\noexpand\@missingfileerror
642         {\filename@area\filename@base}%
643         {\ifx\filename@ext\relax tex\else\filename@ext\fi}}%

```

This line now just sets \@missingfile@<part>:

```

644   \reserved@a

```

Now here we have to use it. The file here is guaranteed to exist, because \@missingfileerror ensures so, but we have to use \InputIfFileExists because it executes the file hooks.

```

645   \edef\reserved@a{\noexpand\@iinput{%
646     \@missingfile@area\@missingfile@base.\@missingfile@ext}}%
647   \reserved@a}}
648 \</2kernel | latexrelease>
649 \<latexrelease> \EndIncludeInRelease

```

```

650 <latexrelease>\IncludeInRelease{2019/10/01}%
651 <latexrelease>          {\@iinput}{Quote file names}%
652 <latexrelease>
653 <latexrelease>\def\@iinput#1{%
654 <latexrelease>  \InputIfFileExists{#1}{}%
655 <latexrelease>  {\filename@parse\@curr@file
656 <latexrelease>   \edef\reserved@a{\noexpand\@missingfileerror
657 <latexrelease>    {\filename@area\filename@base}%
658 <latexrelease>    {\ifx\filename@ext\relax tex\else\filename@ext\fi}}%
659 <latexrelease>   \reserved@a}}
660 <latexrelease>\EndIncludeInRelease

661 <latexrelease>\IncludeInRelease{0000/00/00}%
662 <latexrelease>          {\@iinput}{Quote file names}%
663 <latexrelease>\def\@iinput#1{%
664 <latexrelease>  \InputIfFileExists{#1}{}%
665 <latexrelease>  {\filename@parse{#1}%
666 <latexrelease>   \edef\reserved@a{\noexpand\@missingfileerror
667 <latexrelease>    {\filename@area\filename@base}%
668 <latexrelease>    {\ifx\filename@ext\relax tex\else\filename@ext\fi}}%
669 <latexrelease>   \reserved@a}}
670 <latexrelease>\EndIncludeInRelease
671 <*2ekernel>

```

(End of definition for \@iinput.)

\@input Define \@input in terms of \IfFileExists. So this is a ‘safe input’ command, but the files input are not listed by \listfiles.

We don’t want .aux, .toc files etc be listed by \listfiles. However, something like .bbl probably should be listed and thus should be implemented not by \@input.

```

672 \def\@input#1{%
673   \IfFileExists{#1}{\@input\@filef@und}{\typeout{No file #1.}}

```

(End of definition for \@input.)

\@input@ Version of \@input that does add the file to \@filelist.

```

674 \def\@input@#1{\InputIfFileExists{#1}{\typeout{No file #1.}}

```

(End of definition for \@input@.)

\@missingfileerror This ‘error’ command avoids T_EX’s primitive missing file loop.

Missing file error. Prompt for a new filename, offering a default extension.

Changes to \@missingfileerror: rather than trying to input the file by force, now \@missingfileerror just returns three \@missingfile@{part} and the caller macro is responsible for doing the right thing with it.

```

675 </2ekernel>
676 <*2ekernel | latexrelease>
677 <latexrelease>\IncludeInRelease{2020/10/01}%
678 <latexrelease>          {\@missingfileerror}{Do not load missing file immediately}%
679 \gdef\@missingfileerror#1#2{%
680   \typeout{^^J! LaTeX Error: File ‘#1.#2’ not found.^^J^^J%
681   Type X to quit or <RETURN> to proceed,^^J%
682   or enter new name. (Default extension: #2)^^J}%
683   \message{Enter file name: }%
684   {\endlinechar\m@ne

```



```

685     \global\read\m@ne to\@gtempa}%
686     \ifx\@gtempa\@empty
If the user answers with \return, fallback to the .tex file (previously it did nothing).
687     \let\@missingfile@area\@empty
688     \let\@missingfile@base\@empty
689     \def\@missingfile@ext{tex}%
690     \else
Use \batchmode\read-1 to <tl> to end the TEX run, same as expl3 does (it was
\batchmode\@end before).
691     \def\reserved@b{\batchmode\read-1 to \reserved@a}%
692     \def\reserved@a{x}\ifx\reserved@a\@gtempa\reserved@b\fi
693     \def\reserved@a{X}\ifx\reserved@a\@gtempa\reserved@b\fi
694     \filename@parse\@gtempa
695     \edef\filename@ext{%
696     \ifx\filename@ext\relax#2\else\filename@ext\fi}%
697     \edef\reserved@a{%
Only check \IfFileExists (it was \InputIfFileExists).
698     \noexpand\IfFileExists
699     {\filename@area\filename@base.\filename@ext}%
If the file exists, define \@missingfile@<part>.
700     {\def\noexpand\@missingfile@area{\filename@area}%
701     \def\noexpand\@missingfile@base{\filename@base}%
702     \def\noexpand\@missingfile@ext {\filename@ext}}%
703     {\noexpand\@missingfileerror
704     {\filename@area\filename@base}{\filename@ext}}}%
705     \reserved@a
706     \fi
707 }
708 </2ekernel | latexrelease>
709 <latexrelease>\EndIncludeInRelease
710 <latexrelease>\IncludeInRelease{0000/00/00}%
711 <latexrelease>     {\@missingfileerror}{Do not load missing file immediately}%
712 <latexrelease>
713 <latexrelease>\gdef\@missingfileerror#1#2{%
714 <latexrelease>     \typeout{^^J! LaTeX Error: File ‘#1.#2’ not found.^^J^^J%
715 <latexrelease>     Type X to quit or <RETURN> to proceed,^^J%
716 <latexrelease>     or enter new name. (Default extension: #2)^^J}%
717 <latexrelease>     \message{Enter file name: }%
718 <latexrelease>     {\endlinechar\m@ne
719 <latexrelease>     \global\read\m@ne to\@gtempa}%
720 <latexrelease>     \ifx\@gtempa\@empty
721 <latexrelease>     \else
722 <latexrelease>         \def\reserved@a{x}\ifx\reserved@a\@gtempa\batchmode\@end\fi
723 <latexrelease>         \def\reserved@a{X}\ifx\reserved@a\@gtempa\batchmode\@end\fi
724 <latexrelease>         \filename@parse\@gtempa
725 <latexrelease>         \edef\filename@ext{%
726 <latexrelease>             \ifx\filename@ext\relax#2\else\filename@ext\fi}%
727 <latexrelease>         \edef\reserved@a{%
728 <latexrelease>             \noexpand\InputIfFileExists
729 <latexrelease>             {\filename@area\filename@base.\filename@ext}%
730 <latexrelease>             {}%

```

```

731 <latexrelease>          {\noexpand\@missingfileerror
732 <latexrelease>          {\filename@area\filename@base}{\filename@ext}}}%
733 <latexrelease>          \reserved@a
734 <latexrelease>          \fi}
735 <latexrelease>
736 <latexrelease>\EndIncludeInRelease
737 <*2ekernel>

```

(End of definition for \@missingfileerror.)

\@obsoletefile For compatibility with L^AT_EX 2.09 document styles, we distribute files called `article.sty`, `book.sty`, `report.sty`, `slides.sty` and `letter.sty`. These use the command `\@obsoletefile`, which produces a warning message.

```

738 \def\@obsoletefile#1#2{%
739   \@latex@warning@no@line{inputting ‘#1’ instead of obsolete ‘#2’}}
740 \@onlypreamble\@obsoletefile

```

1.2 Listing files

\@filelist A list of files input so far. The initial value of `\@gobble` eats the comma before the first file name.

```

741 \let\@filelist\@gobble

```

\@addtofilelist Add to the list of files input so far. This ‘real’ definition is only used for ‘cfg’ files during `initex`. An initial definition of `\@gobble` has already been set.

```

742 %\def\@addtofilelist#1{\xdef\@filelist{\@filelist,#1}}

```

```

\if@listfiles@hashes 743 \ExplSyntaxOn
\if@listfiles@sizes 744 \keys_define:nm { __kernel / listfiles }
745 {
746   hashes .legacy_if_set:n = @listfiles@hashes ,
747   sizes .legacy_if_set:n = @listfiles@sizes
748 }
749 \ExplSyntaxOff

```

\listfiles A preamble command to cause `\end{document}` to list files input from the main file.

```

750 \NewDocumentCommand\listfiles{0}{-}%
751 \SetKeys[__kernel/listfiles]{#1}%
752 \let\listfiles\relax
753 \def\@listfiles##1##2##3##4##5##6##7##8##9\@{-%
754   \def\reserved@d{\}%
755   \@tfor\reserved@c:=##1##2##3##4##5##6##7##8\do{%
756     \ifx\reserved@c\reserved@d
757       \edef\filename@area{ \filename@area}%
758     \fi}}%
759 \def\@dofilelist{%
760   \typeout{^^J *File List*}%
761   \@for\currname:=\@filelist\do{%
762     \filename@parse\currname
763     \edef\reserved@a{%
764       \filename@base.%

```

```

765     \ifx\filename@ext\relax tex\else\filename@ext\fi}%
766 \expandafter\let\expandafter\reserved@b
767     \csname ver@\reserved@a\endcsname

```

Packages that `\relax` their `\ver@...` string to allow for multiple loading (e.g., fontenc) can use `\ver@@...` to store the version information instead.

```

768     \ifx\reserved@b\relax
769     \expandafter\let\expandafter\reserved@b
770         \csname ver@@\reserved@a\endcsname
771 \fi
772 \expandafter\expandafter\expandafter\@listfiles\expandafter
773     \filename@area\filename@base\@
774 \typeout{%
775     \filename@area\reserved@a
776     \ifx\reserved@b\relax\else\@spaces\reserved@b\fi

```

Now we add the additional information if requested.

```

777     \ifnum0%
778     \if@listfiles@hashes1\fi
779     \if@listfiles@sizes1\fi
780     >0 %
781     ^^J\@spaces
782     (%
783         \if@listfiles@sizes
784             size \@dofilelist@size\@currname
785             \if@listfiles@hashes
786                 , %
787             \fi
788         \fi
789         \if@listfiles@hashes
790             hash \@dofilelist@hash\@currname
791         \fi
792     )%
793 \fi
794 }}%
795 \typeout{ *****^^J}}

```

```

796 \ExplSyntaxOn
\@dofilelist@hash 797 \cs_new_eq:NN \@dofilelist@hash \file_mdfive_hash:n
\@dofilelist@size 798 \cs_new_eq:NN \@dofilelist@size \file_size:n
799 \ExplSyntaxOff

```

The `\@filelist` will be de-activated if `\listfiles` does not appear in the preamble. `\begin{document}` contains code equivalent to the following:

```

\AtBeginDocument{%
    \ifx\@listfiles\@undefined
        \let\@filelist\relax
        \let\@addtofilelist\@gobble
    \fi}
800 \@onlypreamble\listfiles

```

```
\@dofilelist 801 \let\@dofilelist\relax
802 \</2kernel>
(End of definition for \@obsoletefile and others.)
```

File 21

ltoutenc.dtx

1 Font encodings

This section of the kernel contains commands for declaring encoding-specific commands, such as accents. It also contains the code for some of the encoding files, including `omlenc.def`, `omsenc.def`, `t1enc.def` and `ot1enc.def` files, which define the OML, OMS, T1 and OT1 encodings, and the `fontenc` package for selecting encodings.

The `fontenc` package has options for encodings, of which the last option is the default encoding. For example, to use the OT2, OT3 and T1 encodings, with T1 as the default, you say:

```
\usepackage[OT2,OT3,T1]{fontenc}
```

The standard kernel set-up loads font encoding files and selects an encoding as follows.

```
\input {omlenc.def}
\input {t1enc.def}
\input {ot1enc.def}
\input {omsenc.def}
\fontencoding{OT1}
```

Note that the files in the standard `inputenc` package depend on this behaviour of the kernel.

The syntax for declaring encoding-specific commands is:

```
\DeclareTextCommand{<command>}{<encoding>}
                               [<number>] [<default>]{<commands>}
```

This command is like `\newcommand`, except that it defines a command which is specific to one encoding. The resulting command is always robust, even if its definition is fragile. For example, the definition of `\l` in the OT1 encoding is:

```
\DeclareTextCommand{\l}{OT1}{\@xxxii l}
```

`\DeclareTextCommand` takes the same optional arguments as `\newcommand`.

```
\ProvideTextCommand{<command>}{<encoding>}
                               [<number>] [<default>]{<commands>}
```

This acts like `\DeclareTextCommand`, but does nothing if the command is already defined.

```
\DeclareTextSymbol{<command>}{<encoding>}{<slot>}
```

This command defines a text symbol, with a particular slot in that encoding. The commands:

```
\DeclareTextSymbol{\ss}{OT1}{25}
\DeclareTextCommand{\ss}{OT1}{\char25 }
```

have the same effect, but the `\DeclareTextSymbol` is faster.

```
\DeclareTextAccent{<command>}{<encoding>}{<slot>}
```

This command declares a text accent. The commands:

```
\DeclareTextAccent{"}{OT1}{127}
\DeclareTextCommand{"}{OT1}{\add@accent {127}}
```

have the same effect.

```
\DeclareTextComposite{<command>}
                               {<encoding>}{<argument>}{<slot>}
```

This command declares a composite letter, for example in the T1 encoding `\'a` is slot 225, which is declared by:

```
\DeclareTextComposite{\'}{T1}{a}{225}
```

The *command* will normally have been declared with `\DeclareTextAccent`, or as a one-argument `\DeclareTextCommand`.

`\DeclareTextComposite` is the most common example of using the more general declaration `\DeclareTextCompositeCommand`, which can define a composite to be an arbitrary piece of text.

```
\DeclareTextCompositeCommand{<command>}
                               {<encoding>}{<argument>}{<text>}
```

For example, in the OT1 encoding `Å` has a hand-crafted definition this is declared as follows

```
\DeclareTextCompositeCommand{\r}{OT1}{A}
{\leavevmode\setbox\z@\hbox{!}\dimen@ht\z@\advance\dimen@-1ex%
 \rlap{\raise.67\dimen@\hbox{\char23}}A}
```

The *command* will normally have been declared with `\DeclareTextAccent`, or as a one-argument `\DeclareTextCommand`.

The commands defined using the above declarations can be used in two ways. Normally they are used by just calling the command in the appropriate encoding, for example `\ss`. However, sometimes you may wish to use a command in an encoding where it is not defined. If the command has no arguments, then you can use it in another encoding by calling `\UseTextSymbol`:

```
\UseTextSymbol{<encoding>}{<command>}
```

For example, `\UseTextSymbol{OT1}{\ss}` has the same effect as:

```
{\fontencoding{OT1}\selectfont\ss}
```

If the command has one argument then you can use it in another encoding by calling `\UseTextAccent`:

```
\UseTextAccent{<encoding>}{<command>}{<text>}
```

For example, if the current encoding is OT2 then `\UseTextAccent{OT1}{\'}{a}` has the same effect as:

```
{\fontencoding{OT1}\selectfont\'{\fontencoding{OT2}\selectfont a}}
```

You can also declare a default definition for a text command, which will be used if the current encoding has no appropriate definition. Such use will also set the definition for this command in the current encoding to equal this default definition; this makes subsequent uses of the command much faster.

```
\DeclareTextCommandDefault{<command>}{<definition>}
```

For example, the default definition of the command `\textonequarter` (which produces the fraction $\frac{1}{4}$) could be built using math mode:

```
\DeclareTextCommandDefault{\textonequarter}{\ensuremath {\frac{1}{4}}}
```

There is a matching `\Provide` command which will not override an existing default definition:

```
\ProvideTextCommandDefault{<command>}{<definition>}
```

The most common use for these commands is to use symbols from other encodings, so there are some optimizations provided:

```
\DeclareTextSymbolDefault{<command>}{<encoding>}
\DeclareTextAccentDefault{<command>}{<encoding>}
```

are short for:

```
\DeclareTextCommandDefault{<command>}
      {\UseTextSymbol{<encoding>}{<command>}}
\DeclareTextCommandDefault[1]{<command>}
      {\UseTextAccent{<encoding>}{<command>}{#1}}
```

For example, to make OT1 the default encoding for `\ss` and `\'` you say:

```
\DeclareTextSymbolDefault{\ss}{OT1}
\DeclareTextAccentDefault{\'}{OT1}
```

Note that you can use these commands on any zero- or one-argument commands declared with `\DeclareText*` or `\ProvideText*`, not just those defined using `\DeclareTextSymbol` or `\DeclareTextAccent`.

1.1 Removing encoding-specific commands

In some cases encoding definitions are given to provide some limited support since nothing better is available, for example, the definition for `\textdollar` in OT1 is a hack since \$ and £ actually share the same slot in this encoding. Thus if such a glyph becomes available in a different encoding (e.g., TS1) one would like to get rid of the flaky one and make the default definition point to the new encoding. In such a case defining

```
\DeclareTextSymbol{\textdollar}{TS1}{36}
\DeclareTextSymbolDefault{\textdollar}{TS1}
```

is not enough since if typesetting in OT1 L^AT_EX will still find the encoding specific-definition for OT1 and therefore ignore the new default. Therefore to ensure that in this case the TS1 version is used we have to remove the OT1 declaration:

```
\UndeclareTextCommand{\textdollar}{OT1}
```

Since the \$ sign is a proper glyph in the T1 encoding there is no point removing its definition and forcing L^AT_EX to pick up the TS1 version if typesetting in this encoding. However, assume you want to use the variant dollar sign, i.e., \$ for your dollars. In that case you have to get rid of the T1 declaration as well, e.g., the following would do that for you:

```
\UndeclareTextCommand{\textdollar}{OT1}
\UndeclareTextCommand{\textdollar} {T1}
\DeclareTextCommandDefault{\textdollar}
{\UseTextSymbol{TS1}\textdollaroldstyle}
```

1.2 The order of declarations

If an encoding-specific command is defined for more than one encoding, then it will execute fastest in the encoding in which it was defined last since its top-level definition will be set up to execute in that encoding without any overhead.

For this reason the file `fonttext.ltx` currently first loads the definitions for the T1 encoding and then those for the OT1 encoding so that typesetting in OT1 is optimized since that is (still) the default. However, when T1 is explicitly requested (via `\usepackage[T1]{fontenc}`) the top-level definitions are automatically changed to favour T1 since its declarations are reloaded in the process.

For the same reason default declarations should never come last since they are implemented as a special encoding themselves (with the name ?). Specifying them last would simply mean to make those encoding-specific commands equally inefficient in all encodings. Therefore the `textcomp` package, for example, first sets up all defaults to point to TS1 and then declares the commands in the TS1 encoding.

1.3 Docstrip modules

This `.dtx` file is be used to generate several related files containing font encoding definitions. The mutually exclusive docstrip options are listed here.

T1	generates <code>t1enc.def</code> for the Cork encoding.
TS1	generates <code>ts1enc.def</code> for the Text Companion encoding.
TS1sty	generates <code>textcomp.sty</code> , package that sets up use of the Text Companion encoding.
OT1	generates <code>ot1enc.def</code> for Knuth's CM encoding.
OMS	generates <code>omsenc.def</code> for Knuth's math symbol encoding.
OML	generates <code>omlenc.def</code> for Knuth's math letters encoding.
OT4	generates <code>ot4enc.def</code> for the Polish extension to the OT1 encoding, created by B. Jackowski and M. Ryćko for use with the Polish version of Computer Modern and Computer Concrete.
TU	generates <code>tuenc.def</code> for Unicode font encoding.
package	generates <code>fontenc.sty</code> for selecting encodings.
2ekernel	for the kernel commands.

1.4 Definitions for the kernel

1.4.1 Declaration commands

This section contains definitions for commands such as accents which depend on the current encoding. These commands will usually be kept in .def files, for example `ot1enc.def` contains the definitions for the OT1 encoding.

```
1 (*2kernel)
2 \message{font encodings,}
   Far too many macros in one block here!
```

If you say:

```
\DeclareTextCommand{\foo}{T1}...
```

then `\foo` is defined to be `\T1-cmd \foo \T1\foo`, where `\T1\foo` is *one* control sequence, not two! We then call `\newcommand` to define `\T1\foo`.

```
\DeclareTextCommand
\ProvideTextCommand
\DeclareTextSymbol
  \@dec@text@cmd
\chardef@text@cmd
  \@changed@cmd
  \@changed@x
\TextSymbolUnavailable
  \@inmathwarn
```

```
3 \def\DeclareTextCommand{%
4   \@dec@text@cmd\newcommand}
5 \def\ProvideTextCommand{%
6   \@dec@text@cmd\providecommand}
7 \def\@enc@info#1{%
8   \GenericInfo{(Encoding)\@spaces\@spaces\@spaces\space\space}%
9   {LaTeX Encoding Info: \space\space\space#1}}%
10 \def\@dec@text@cmd#1#2#3{%
```

For logging, there are a number of different cases to cater for.

```
11 \ifcsname #3\string#2\endcsname
12 \@enc@info{%
13   \ifx#1\providecommand
14     Ignoring declaration for
15   \else
16     Redeclaring
17   \fi
18   text
```

This test distinguishes between commands and symbols:

```
19 \ifx#1\chardef@text@cmd
20   symbol
21 \else
22   command
23 \fi
24 \string#2 (encoding #3)}%
25 \fi
26 \expandafter\def\expandafter#2%
27   \expandafter{%
28     \csname#3-cmd\expandafter\endcsname
29     \expandafter#2%
30     \csname#3\string#2\endcsname
31   }%
32 \let\@ifdefinable\@rc@ifdefinable
33 \expandafter#1\csname#3\string#2\endcsname}
```

This command was introduced to fix a major bug in `\@dec@text@cmd` without changing that command itself. This was thought to be necessary because it is defined in more than one package. (Perhaps the more serious bug is to put complex low-level commands like this in packages?)

The problem it solves is that whereas both `\newcommand` and `\providecommand` (used just above) both handle the resetting of `\@ifdefinable` (following its disabling in `\@dec@text@cmd`), the primitive `\chardef` neither needs the disabling, nor does the resetting.

```

34 \def\chardef@text@cmd{%
35   \let\@ifdefinable\@ifdefinable
36   \chardef
37 }
38 \def\DeclareTextSymbol#1#2#3{%
39   \@dec@text@cmd\chardef@text@cmd#1{#2}#3\relax
40 }

```

The declarations are only available before `\begin{document}`.

```

41 \@onlypreamble\DeclareTextCommand
42 \@onlypreamble\DeclareTextSymbol

```

The sneaky bit in all this is what `\T1-cmd \foo \T1\foo` does. There are five possibilities, depending on the current values of `\protect`, `\cf@encoding` and `\ifmmode`:

- If `\protect` is `\@typeset@protect` and `\cf@encoding` is `T1`, then we execute `\T1\foo`. This should be the normal behaviour, and is optimized for speed.
- If `\protect` is `\@typeset@protect`, `\cf@encoding` is (say) `OT1`, and `\OT1\foo` is defined, then we execute `\OT1\foo`.
- If `\protect` is `\@typeset@protect`, `\cf@encoding` is (say) `OT1`, we're in text mode, and `\OT1\foo` is undefined, then we define `\OT1\foo` to be the default value of `\foo`, and execute `\OT1\foo`.
- If `\protect` is `\@typeset@protect`, `\cf@encoding` is (say) `OT1`, we're in math mode, and `\OT1\foo` is undefined, then we execute the default value of `\foo`. (This is necessary so that things like `X_\copyright` work properly.)
- If `\protect` is not `\@typeset@protect` then we execute `\noexpand\foo`. For example, if we are writing to a file, then this results in `\foo` being written. If we are in a `\mark`, then `\foo` will be put in the mark—since `\foo` is robust, it will then survive all the things which may happen to it whilst it's a `\mark`.

So after all that, we will either execute the appropriate definition of `\foo` for the current encoding, or we will execute `\noexpand\foo`.

The default value of `\foo` is `\?\foo` if it is defined, and an error message otherwise.

When the encoding is changed from `T1` to `OT1`, `\T1-cmd` is defined to be `\@changed@cmd` and `\OT1-cmd` is defined to be `\@current@cmd`. This means that the test for what the current encoding is can be performed quickly.

```

43 \def\@current@cmd#1{%
44   \ifx\protect\@typeset@protect
45     \inmathwarn#1%
46   \else
47     \noexpand#1\expandafter\@gobble
48   \fi}

```

```

49 \def\@changed@cmd#1#2{%
50   \ifx\protect\@typeset@protect
51     \@inmathwarn#1%
52     \expandafter\ifx\csname\cf@encoding\string#1\endcsname\relax
53       \expandafter\ifx\csname ?\string#1\endcsname\relax
54         \expandafter\def\csname ?\string#1\endcsname{%
55           \TextSymbolUnavailable#1%
56         }%
57       \fi
58     \global\expandafter\let
59       \csname\cf@encoding\string#1\endcsname
60       \csname ?\string#1\endcsname
61     \fi
62     \csname\cf@encoding\string#1%
63       \expandafter\endcsname
64   \else
65     \noexpand#1%
66   \fi}

67 \gdef\TextSymbolUnavailable#1{%
68   \@latex@error{%
69     Command \protect#1 unavailable in encoding \cf@encoding%
70   }\@eha}

```

The command `\@inmathwarn` produces a warning message if we are currently in math mode. Note that since this command is used inside text commands, it can't call `\relax` before the `\ifmmode`. This means that it is possible for the warning to fail to be issued at the beginning of a row of an `halign` whose template enters math mode. This is probably a bad feature, but there's not much that can be done about it, since adding a `\relax` would break ligatures and kerning between text symbols.

A more efficient solution would be to make `\@inmathwarn` and `\@inmatherr` equal to `\@empty` and `\relax` by default, and to have `\everymath` reset them to their usual definitions. This is left for future investigation (for example it may break some third party code).

```

71 \def\@inmathwarn#1{%
72   \ifmmode
73     \@latex@warning{Command \protect#1 invalid in math mode}%
74   \fi}

```

(End of definition for `\DeclareTextCommand` and others.)

`\DeclareTextCommandDefault` These define commands with encoding ?.

`\ProvideTextCommandDefault` Note that `\DeclareTextCommandDefault` can only be used in the preamble, but that the `\Provide` version is allowed in inputenc .def files, so is allowed anywhere.

```

75 \def\DeclareTextCommandDefault#1{%
76   \DeclareTextCommand#1?}

77 \def\ProvideTextCommandDefault#1{%
78   \ProvideTextCommand#1?}

79 \@onlypreamble\DeclareTextCommandDefault
80 %\@onlypreamble\ProvideTextCommandDefault

```

They require `\? -cmd` to be initialized as `\@changed@cmd`.

```

81 \expandafter\let\csname?-cmd\endcsname\@changed@cmd

```

(End of definition for `\DeclareTextCommandDefault` and `\ProvideTextCommandDefault`.)

`\DeclareTextAccent` This is just a disguise for defining a T_EX `\accent` command.

```
82 \def\DeclareTextAccent#1#2#3{%
83   \DeclareTextCommand#1{#2}{\add@accent{#3}}
84 \@onlypreamble\DeclareTextAccent
```

(End of definition for `\DeclareTextAccent`.)

`\add@accent` To save space this code is shared between all text accents that are set using the `\accent` primitive. The argument is pre-set in a box so that any font loading that is needed is already done within the box. This is needed because font-loading involves grouping and that would prevent the accent mechanism from working so that the accent would not be positioned over the argument. Declarations that change the font should be allowed (only low-level ones are at present) inside the argument of an accent command, but not size changes, as they involve `\setbox` operations which also inhibit the mechanism of the `\accent` primitive.

Note that the whole process is within a group. For a detailed discussion of this reimplementaion and its deficiencies, see pr/3160.

```
85 \def\add@accent#1#2{\hmode@bgroup
```

Turn off the group in `\UseTextSymbol` in case this is used inside the argument of `\add@accent`.

```
86   \let\hmode@start@before@group\@firstofone
87   \setbox\@tempboxa\hbox{#2%
```

When presetting the argument in a box we record its `\spacefactor` for later use after the accent got typeset. This way something like `\‘A` gets the spacefactor of `A` (i.e., 999) rather than the default value of 1000.

```
88     \global\mathchardef\accent@spacefactor\spacefactor}%
```

The accent primitive doesn't allow things `\begingroup` to interfere between accent and base character. Therefore we need to avoid that (they are some hidden inside `\maybe@load@fontshape`). As we don't have to load the fontshape in this case (as that already happened in the box above, if necessary), we simply disable that part of the code temporarily. We also ignore `\ignorespaces` which has the same issue and may show up as part of `\normalfont` if that is used.

```
89   \let\maybe@load@fontshape\relax
90   \let\ignorespaces\relax
91   \accent#1 #2\egroup\ifmmode\else\spacefactor\accent@spacefactor\fi}
```

Default definition for `\accent@spacefactor` prevents a horrible death of the above macro inside an unprotected `\edef`.

```
92 \let\accent@spacefactor\relax
```

(End of definition for `\add@accent`.)

`\hmode@bgroup`

```
93 \def\hmode@bgroup{\leavevmode\bgroup}
```

(End of definition for `\hmode@bgroup`.)

```

\DeclareTextCompositeCommand Another amusing game to play with \expandafter, \csname, and \string. When you
\DeclareTextComposite say \DeclareTextCompositeCommand{\foo}{T1}{a}{bar}, we look to see if the expansion
  \@text@composite of \T1\foo begins with \@text@composite, and if it doesn't, we redefine \T1\foo
\@text@composite@x to be:
  \@strip@args
    #1 -> \@text@composite \T1\foo #1\@empty \@text@composite {...}

```

where ... is the previous definition of \T1\foo. Finally, we define \\T1\foo-a to expand to bar.

```

94 </2ekernel>
95 <latexrelease>\IncludeInRelease{2017/04/15}{\DeclareTextCompositeCommand}
96 <latexrelease> {test for undeclared accent}%
97 <*2ekernel | latexrelease>
98 \def\DeclareTextCompositeCommand#1#2#3#4{%
99   \expandafter\let\expandafter\reserved@a\csname#2\string#1\endcsname
100  \ifx\reserved@a\relax
101
102   \DeclareTextCommand#1{#2}{%
103     \@latex@error{Text composite \string#1 undeclared in encoding #2}\@eha}%
104   \enc@info{Text composite with undeclared \string#1 in encoding #2}%
105   \expandafter\let\expandafter\reserved@a\csname#2\string#1\endcsname
106   \fi
107   \expandafter\expandafter\expandafter\ifx
108   \expandafter\@car\reserved@a\relax\relax\@nil \@text@composite \else
109   \edef\reserved@b##1{%
110     \def\expandafter\noexpand
111       \csname#2\string#1\endcsname###1{%
112       \noexpand\@text@composite
113         \expandafter\noexpand\csname#2\string#1\endcsname
114         ###1\noexpand\@empty\noexpand\@text@composite
115         {##1}}}%
116   \expandafter\reserved@b\expandafter{\reserved@a{##1}}%
117   \fi
118   \expandafter\def\csname\expandafter\string\csname
119   #2\endcsname\string#1-\string#3\@empty\endcsname{#4}%
120 }
121 </2ekernel | latexrelease>
122 <latexrelease>\EndIncludeInRelease
123 <latexrelease> {test for undeclared accent}%
124 <latexrelease>\def\DeclareTextCompositeCommand#1#2#3#4{%
125 <latexrelease> \expandafter\let\expandafter\reserved@a
126 <latexrelease> \csname#2\string#1\endcsname
127 <latexrelease> \expandafter\expandafter\expandafter\ifx
128 <latexrelease> \expandafter\@car\reserved@a\relax\relax\@nil
129 <latexrelease> \@text@composite \else
130 <latexrelease> \edef\reserved@b##1{%
131 <latexrelease> \def\expandafter\noexpand
132 <latexrelease> \csname#2\string#1\endcsname###1{%
133 <latexrelease> \noexpand\@text@composite
134 <latexrelease> \expandafter\noexpand\csname#2\string#1\endcsname
135 <latexrelease> ###1\noexpand\@empty\noexpand\@text@composite
136 <latexrelease> {##1}}}%
137 <latexrelease> \expandafter\reserved@b\expandafter{\reserved@a{##1}}%

```

```

138 <latexrelease> \fi
139 <latexrelease> \expandafter\def\csname\expandafter\string\csname
140 <latexrelease> #2\endcsname\string#1-\string#3\@empty\endcsname{#4}}
141 <latexrelease>\EndIncludeInRelease
142 <*2ekernel>
143 \@onlypreamble\DeclareTextCompositeCommand

```

This all works because:

```
\@text@composite \T1\foo A\@empty \@text@composite {...}
```

expands to `\T1\foo-A` if `\T1\foo-A` has been defined, and `{...}` otherwise.

Note that `\@text@composite` grabs the first token of the argument and puts just that in the csname. This is so that `\'\{\textit{e}\}` will work—it checks whether `\T1\'-\textit` is defined (which presumably it isn't) and so expands to `{\accent 1 \textit{e}}`.

This trick won't always work, for example `\'\{\itshape e}\}` will expand to (with spaces added for clarity):

```
\csname \string \T1\' - \string {\itshape e} \@empty \endcsname
```

which will die pretty horribly. Unfortunately there's not much can be done about this if we're going to use `\csname` lookups as a fast way of accessing composites.

This has an unfortunate 'misfeature' though, which is that in the T1 encoding, `\'\{aa}` produces á. This is not the expected behaviour, and should perhaps be fixed if the fix doesn't affect performance too badly.

Finally, it's worth noting that the `\@empty` is used in `\@text@composite` so that accents will work even when the argument is empty. If you say `\'\{\}` then this looks up `\T1\'-\@empty`, which ought to be `\relax`, and so all is well. If we didn't include the `\@empty`, then `\'\{\}` would expand to:

```
\csname \string \T1\' - \string \endcsname
```

so the `\endcsname` would be `\string`'ed and the whole of the rest of the document would be put inside the `\csname`. This would not be good.

```

144 \def\@text@composite#1#2#3\@text@composite{%
145   \expandafter\@text@composite@x
146     \csname\string#1-\string#2\endcsname}

```

Originally the `\@text@composite@x` macro had two arguments and if `#1` was not `\relax` it was executed, otherwise `#2` was executed. All this happened within the `\ifx` code so that neither `#1` nor `#2` could have picked up any additional arguments from the input stream. This has now been changed using the typical `\@firstoftwo / \@secondoftwo` coding. This way the final expansion will happen without any `\else` or `\fi` intervening in the case that we need to get a further token from the input stream.

```

147 \def\@text@composite@x#1{%
148   \ifx#1\relax
149     \expandafter\@secondoftwo
150   \else
151     \expandafter\@firstoftwo
152   \fi
153   #1}

```

The command `\DeclareTextComposite` uses `\DeclareTextCompositeCommand` to declare a command which expands out to a single glyph.

```

154 \catcode\z@=11\relax
155 \def\DeclareTextComposite#1#2#3#4{%
156   \def\reserved@a{\DeclareTextCompositeCommand#1{#2}{#3}}%
157   \bgroup
158     \lccode\z@#4%
159     \lowercase{%
160   \egroup
161     \reserved@a ^^@}}
162 \catcode\z@=15\relax
163 \@onlypreamble\DeclareTextComposite

```

(End of definition for `\DeclareTextCompositeCommand` and others.)

```

164 </2kernel>
165 <*2kernel | latexrelease>
166 <latexrelease>\IncludeInRelease{2019/10/01}%
167 <latexrelease>           {\UseTextAccent}{Make commands robust}%

```

`\UseTextAccent` `\UseTextSymbol` `\@use@text@encoding` These fragile commands access glyphs from different encodings. They use grotty low-level calls to the font selection scheme for speed, and in order to make sure that `\UseTextSymbol` doesn't do anything which you're not allowed to do between an `\accent` and its glyph.

For a detailed discussion of this reimplementaion and its deficiencies, see pr/3160.

```

168 \DeclareRobustCommand*\UseTextAccent [3] {%
169   \hmode@start@before@group
170   {%

```

Turn off the group in `\UseTextSymbol` in case this is used inside the arguments of `\UseTextAccent`.

```

171   \let\hmode@start@before@group\@firstofone
172   \let\@curr@enc\cf@encoding
173   \@use@text@encoding{#1}%
174   #2{\@use@text@encoding\@curr@enc#3}%
175   }}

```

```

176 \DeclareRobustCommand*\UseTextSymbol [2] {%
177   \hmode@start@before@group
178   {%
179     \def\@wrong@font@char{\MessageBreak
180       for \noexpand\symbol'\string#2'}%
181     \@use@text@encoding{#1}%
182     #2%
183     }%
184   }

```

```

185 </2kernel | latexrelease>
186 <latexrelease>\EndIncludeInRelease
187 <latexrelease>\IncludeInRelease{0000/00/00}%
188 <latexrelease>           {\UseTextAccent}{Make commands robust}%
189 <latexrelease>
190 <latexrelease>\kernel@make@fragile\UseTextAccent

```

```

191 <latexrelease>\kernel@make@fragile\UseTextSymbol
192 <latexrelease>
193 <latexrelease>\EndIncludeInRelease
194 <*2kernel>

```

Switch to a different text encoding without any grouping for use in `\UseTextAccent` or `\UseTextSymbol` (and for `\oldstylenums`).

```

195 \def\@use@text@encoding#1{%
196   \edef\f@encoding{#1}%
197   \xdef\font@name{%
198     \csname\curr@fontshape/\f@size\endcsname}%
199   \pickup@font
200   \font@name
201   \@@enc@update}

```

(End of definition for \UseTextAccent, \UseTextSymbol, and \@use@text@encoding.)

`\hmode@start@before@group` The `\hmode@start@before@group` starts `hmode` and should be immediately followed by an explicit `{...}`. Its purpose is to ensure that `hmode` is started before this group is opened. Inside `\add@accent` and `\UseTextAccent` it is redefined to remove this group so that it doesn't conflict with the `\accent` primitive.

For a detailed discussion see pr/3160.

```

202 \let\hmode@start@before@group\leavevmode

```

(End of definition for \hmode@start@before@group.)

`\DeclareTextSymbolDefault` Some syntactic sugar. Again, these should probably be optimized for speed.
`\DeclareTextAccentDefault`

```

203 \def\DeclareTextSymbolDefault#1#2{%
204   \DeclareTextCommandDefault#1{\UseTextSymbol{#2}#1}}
205 \def\DeclareTextAccentDefault#1#2{%
206   \DeclareTextCommandDefault#1{\UseTextAccent{#2}#1}}
207 \@onlypreamble\DeclareTextSymbolDefault
208 \@onlypreamble\DeclareTextAccentDefault

```

(End of definition for \DeclareTextSymbolDefault and \DeclareTextAccentDefault.)

`\UndeclareTextCommand` This command safely removes an encoding specific declaration for a given encoding. It is helpful if one intends to use the default definition always and therefore wants to get rid of a declaration for some specific encoding.

```

209 \def\UndeclareTextCommand#1#2{%
If there is no declaration for the current encoding do nothing.
210   \ifcsname #2\string#1\endcsname
Else: throw away that declaration.
211     \@enc@info{Undeclare text command \string#1 (encoding #2)}%
212     \global\expandafter\let\csname#2\string#1\endcsname
213     \undefined

```


But this is unfortunately not enough, we have to take a look at the top-level definition of the encoding specific command which for a command `\foo` would look similar to `\T1-cmd \foo \T1\foo` (three tokens).

Of course, instead of `T1` one could see a different encoding name; which one depends the encoding for which `\foo` was declared last.

Now assume we have just removed the declaration for `\foo` in `T1` and the top-level of `\foo` expands to the above. Then we better change that pretty fast otherwise we do get an “undefined csname error” when we try to typeset `\foo` within `T1` instead of getting the default definition for `\foo`. And what is the best way to change that top-level definition? Well, the only “encoding” we know for sure will still be around is the default encoding denoted by `?`.

Thus in case the last token of the top-level expansion is now undefined we change the declaration to look like `\?-cmd \foo \?\foo` which is done by the following (readable?) code:

```

214     \expandafter\expandafter\expandafter
215     \ifx\expandafter\@thirdofthree#1@\undefined
216     \expandafter\gdef\expandafter#1\expandafter
217         {\csname ?-cmd\expandafter\endcsname\expandafter
218          #1\csname?\string#1\endcsname}%
219     \fi
220 \else
221     \@enc@info{Text command \string#1 (encoding #2) is not declared}%
222 \fi
223 }
224 \@onlypreamble\UndeclareTextCommand

```

(End of definition for \UndeclareTextCommand.)

1.4.2 Hyphenation

```

\patterns We redefine \patterns and \hyphenation to allow the use of commands declared with
\@patterns \DeclareText* to be used inside them.
\hyphenation
\@hyphenation
225 %\let\@patterns\patterns
226 %\let\@hyphenation\hyphenation
227 %\def\patterns{%
228 %   \bgroup
229 %     \let\protect\@empty
230 %     \let\@typeset@protect\@empty
231 %     \let\@changed@x\@changed@x@mouth
232 %   \afterassignment\egroup
233 %   \@patterns
234 %}
235 %\def\hyphenation{%
236 %   \bgroup
237 %     \let\protect\@empty
238 %     \let\@typeset@protect\@empty
239 %     \let\@changed@x\@changed@x@mouth
240 %   \afterassignment\egroup
241 %   \@hyphenation
242 %}

```

(End of definition for \patterns and others.)

1.4.3 Miscellanea

`\a` The `\a` command is used to access the accent commands even when they have been redefined (for example by the `tabbing` environment). Its internal name is `\@tabacckludge`.

The `\string` within the `\csname` guards against something like `'` being active at the point of use.

```
243 \def\@tabacckludge#1{\expandafter\@changed@cmd
244                               \csname\string#1\endcsname\relax}
245 \let\a=\@tabacckludge
```

(End of definition for \a.)

1.4.4 Default encodings

We define the default encodings for most commands to be either OT1, OML or OMS. These defaults are in the kernel and therefore fonts with these encodings must be available unless these defaults are redefined elsewhere. Recall that the standard kernel loads the encoding files for these encodings, and also that for the T1 encoding.

The naming conventions in the kernel are not what we would use if we were starting from scratch. . . Those defined by DEK (like `\ae` and `\ss`) or by the T_EX Users Group Technical Working Group on multi-lingual typesetting (like `\th` and `\ng`) have short names. Those which were added to the kernel in 1993 and early 1994 are named after their Adobe glyph names (like `\guillemotleft` and `\quotedblbase`). Unfortunately, this naming scheme won't work for all glyphs, since some names (like `\space`) are already used, and some (like `\endash`) are very likely to be defined by users. So we're now using the naming scheme of `\text` followed by the Adobe name, (like `\textendash` and `\textsterling`). Except that some glyphs don't have Adobe names, so we're using the names used by fontinst for those (like `\textcompwordmark`). Sigh.

Some accents from OT1:

```
246 \DeclareTextAccentDefault{\"}{OT1}
247 \DeclareTextAccentDefault{\'}{OT1}
248 \DeclareTextAccentDefault{\.}{OT1}
249 \DeclareTextAccentDefault{\=}{OT1}
250 \DeclareTextAccentDefault{\H}{OT1}
251 \DeclareTextAccentDefault{\^}{OT1}
252 \DeclareTextAccentDefault{\'}{OT1}
253 \DeclareTextAccentDefault{\b}{OT1}
254 \DeclareTextAccentDefault{\c}{OT1}
255 \DeclareTextAccentDefault{\d}{OT1}
256 \DeclareTextAccentDefault{\r}{OT1}
257 \DeclareTextAccentDefault{\u}{OT1}
258 \DeclareTextAccentDefault{\v}{OT1}
259 \DeclareTextAccentDefault{\~}{OT1}
```

Some symbols from OT1:

```
260 %\DeclareTextSymbolDefault{\AA}{OT1}
261 \DeclareTextSymbolDefault{\AE}{OT1}
262 \DeclareTextSymbolDefault{\L}{OT1}
263 \DeclareTextSymbolDefault{\OE}{OT1}
264 \DeclareTextSymbolDefault{\O}{OT1}
265 %\DeclareTextSymbolDefault{\aa}{OT1}
266 \DeclareTextSymbolDefault{\ae}{OT1}
267 \DeclareTextSymbolDefault{\i}{OT1}
```

```

268 \DeclareTextSymbolDefault{\j}{OT1}
269 \DeclareTextSymbolDefault{\ij}{OT1}
270 \DeclareTextSymbolDefault{\IJ}{OT1}
271 \DeclareTextSymbolDefault{\l}{OT1}
272 \DeclareTextSymbolDefault{\oe}{OT1}
273 \DeclareTextSymbolDefault{\o}{OT1}
274 \DeclareTextSymbolDefault{\ss}{OT1}
275 \DeclareTextSymbolDefault{\textdollar}{OT1}
276 \DeclareTextSymbolDefault{\textemdash}{OT1}
277 \DeclareTextSymbolDefault{\textendash}{OT1}
278 \DeclareTextSymbolDefault{\textexclamdown}{OT1}
279 %\DeclareTextSymbolDefault{\textthyphenchar}{OT1}
280 %\DeclareTextSymbolDefault{\textthyphen}{OT1}
281 \DeclareTextSymbolDefault{\textquestiondown}{OT1}
282 \DeclareTextSymbolDefault{\textquotedblleft}{OT1}
283 \DeclareTextSymbolDefault{\textquotedblright}{OT1}
284 \DeclareTextSymbolDefault{\textquoteleft}{OT1}
285 \DeclareTextSymbolDefault{\textquoteright}{OT1}
286 \DeclareTextSymbolDefault{\textsterling}{OT1}

```

Some symbols from OMS:

```

287 \DeclareTextSymbolDefault{\textasteriskcentered}{OMS}
288 \DeclareTextSymbolDefault{\textbackslash}{OMS}
289 \DeclareTextSymbolDefault{\textbar}{OMS}
290 \DeclareTextSymbolDefault{\textbardbl}{OMS}
291 \DeclareTextSymbolDefault{\textbraceleft}{OMS}
292 \DeclareTextSymbolDefault{\textbraceright}{OMS}
293 \DeclareTextSymbolDefault{\textbullet}{OMS}
294 \DeclareTextSymbolDefault{\textdaggerdbl}{OMS}
295 \DeclareTextSymbolDefault{\textdagger}{OMS}
296 \DeclareTextSymbolDefault{\textparagraph}{OMS}
297 \DeclareTextSymbolDefault{\textperiodcentered}{OMS}
298 \DeclareTextSymbolDefault{\textsection}{OMS}
299 \DeclareTextAccentDefault{\textcircled}{OMS}

```

Some symbols from OML:

```

300 \DeclareTextSymbolDefault{\textless}{OML}
301 \DeclareTextSymbolDefault{\textgreater}{OML}
302 \DeclareTextAccentDefault{\t}{OML}

```

Some defaults we can fake.

The interface for defining `\copyright` changed, it used to use `\expandafter` to add braces at the appropriate points.

```

303 \DeclareTextCommandDefault{\textcopyright}{\textcircled{c}}
304 % \expandafter\def\expandafter
305 % \copyright\expandafter{\expandafter{\copyright}}
306 \DeclareTextCommandDefault{\textasciicircum}{\~{}}
307 \DeclareTextCommandDefault{\textasciitilde}{\~{}}
308 \DeclareTextCommandDefault{\textunderscore}{%
309 \leavevmode \kern.06em\vbox{\hrule\@width.3em}}

```

There is no good reason anymore to fake `\textcompwordmark`.

```

310 %\DeclareTextCommandDefault{\textcompwordmark}{\leavevmode\kern\z@}
311 \DeclareTextSymbolDefault{\textcompwordmark}{T1}

```

```

312 \DeclareTextCommandDefault{\textvisiblespace}{%
313   \mbox{\kern.06em\vrule \@height.3ex}%
314   \vbox{\hrule \@width.3em}%
315   \hbox{\vrule \@height.3ex}}

```

Using `\fontdimen3` in the next definition is some sort of a kludge (since it is the interword stretch) but it makes the ellipsis come out right in mono-spaced fonts too (since there it is zero).

```

316 \DeclareTextCommandDefault{\textellipsis}{%
317   .\kern\fontdimen3\font
318   .\kern\fontdimen3\font
319   .\kern\fontdimen3\font}

320 %\DeclareTextCommandDefault{\textregistered}{\textcircled{\scshape r}}
321 \DeclareTextCommandDefault{\textregistered}{\textcircled{%
322   \check@mathfonts\fontsize\sf@size\z@\math@fontsfalse\selectfont R}}
323 \DeclareTextCommandDefault{\texttrademark}{\textsuperscript{TM}}
324 \DeclareTextCommandDefault{\SS}{SS}

325 \DeclareTextCommandDefault{\textordfeminine}{\textsuperscript{a}}
326 \DeclareTextCommandDefault{\textordmasculine}{\textsuperscript{o}}

```

1.4.5 Math material

Some commands can be used in both text and math mode:

```

327 \DeclareRobustCommand{\$}{\ifmmode\mathdollar\else\textdollar\fi}

```

We use `\protected` not `\DeclareRobustCommand` so that `\bigl\{` etc. works inside `\protected@edef`.

```

328 \protected\def\lbrace{\ifmmode\lbrace\else\textbraceleft\fi}
329 \protected\def\rbrace{\ifmmode\rbrace\else\textbraceright\fi}

330 \DeclareRobustCommand{\P}{\ifmmode\mathparagraph\else\textparagraph\fi}
331 \DeclareRobustCommand{\S}{\ifmmode\mathsection\else\textsection\fi}
332 \DeclareRobustCommand{\dag}{\ifmmode{\dagger}\else\textdagger\fi}
333 \DeclareRobustCommand{\ddag}{\ifmmode{\ddagger}\else\textdaggerdbl\fi}

```

For historical reasons `\copyright` needs `{}` around the definition in maths.

```

334 \DeclareRobustCommand{\_}{%
335   \ifmmode\nfss@text{\textunderscore}\else\textunderscore\fi}
336 \DeclareRobustCommand{\copyright}{%
337   \ifmmode{\nfss@text{\textcopyright}}\else\textcopyright\fi}
338 \DeclareRobustCommand{\pounds}{%
339   \ifmmode\mathsterling\else\textsterling\fi}

340 \DeclareRobustCommand{\dots}{%
341   \ifmmode\mathellipsis\else\textellipsis\fi}

342 \let\ldots\dots

```

Default definition of the commabelow accent.

```

343 </2ekernel)
344 <latexrelease>\IncludeInRelease{2015/10/01}{\textcommabelow}{comma accent}%
345 <*2ekernel | latexrelease)
346 \DeclareTextCommandDefault\textcommabelow[1]
347   {\hmode\bgroup\oalign{\null#1\crr\hidewidth\raise-.31ex
348   \hbox{\check@mathfonts\fontsize\ssf@size\z@

```

```

349   \math@fontsfalse\selectfont,}\hidewidth}\egroup}
350 <latexrelease>\EndIncludeInRelease
351 </2ekernel | latexrelease>
352 <latexrelease>\IncludeInRelease{0000/00/00}{\textcommabelow}{comma accent}%
353 <latexrelease>\let\textcommabelow\@undefined
354 <latexrelease>\expandafter
355 <latexrelease> \let\csname\string\T1\string\c-G\endcsname\@undefined
356 <latexrelease>\expandafter
357 <latexrelease> \let\csname\string\T1\string\c-K\endcsname\@undefined
358 <latexrelease>\expandafter
359 <latexrelease> \let\csname\string\T1\string\c-k\endcsname\@undefined
360 <latexrelease>\expandafter
361 <latexrelease> \let\csname\string\T1\string\c-L\endcsname\@undefined
362 <latexrelease>\expandafter
363 <latexrelease> \let\csname\string\T1\string\c-l\endcsname\@undefined
364 <latexrelease>\expandafter
365 <latexrelease> \let\csname\string\T1\string\c-N\endcsname\@undefined
366 <latexrelease>\expandafter
367 <latexrelease> \let\csname\string\T1\string\c-n\endcsname\@undefined
368 <latexrelease>\expandafter
369 <latexrelease> \let\csname\string\T1\string\c-R\endcsname\@undefined
370 <latexrelease>\expandafter
371 <latexrelease> \let\csname\string\T1\string\c-r\endcsname\@undefined
372 <latexrelease>\EndIncludeInRelease
    Default definition of the commaabove accent(E.G.).
373 <latexrelease>\IncludeInRelease{2016/02/01}{\textcommaabove}{comma above}%
374 <*2ekernel | latexrelease>
375 \DeclareTextCommandDefault\textcommaabove[1]{%
376   \hmode\bgroup
377   \oalign{%
378     \hidewidth
379     \raise.7ex\hbox{%
380       \check@mathfonts\fontsize\ssf@size\z@\math@fontsfalse\selectfont‘%
381     }%
382     \hidewidth\crrc
383     \null#1\crrc
384   }%
385   \egroup
386 }
387 <latexrelease>\EndIncludeInRelease
388 </2ekernel | latexrelease>
389 <latexrelease>\IncludeInRelease{0000/00/00}{\textcommaabove}{comma above}%
390 <latexrelease>\let\textcommaabove\@undefined
391 <latexrelease>\expandafter
392 <latexrelease> \let\csname\string\OT1\string\c-g\endcsname\@undefined
393 <latexrelease>\expandafter
394 <latexrelease> \let\csname\string\T1\string\c-g\endcsname\@undefined
395 <latexrelease>\EndIncludeInRelease

```

1.5 Definitions for the OT1 encoding

The definitions for the ‘TEX text’ (OT1) encoding.

Declare the encoding.

```

396 <*OT1>
397 \DeclareFontEncoding{OT1}{-}{-}

```

Declare the accents.

```

398 \DeclareTextAccent{"}{OT1}{127}
399 \DeclareTextAccent{'}{OT1}{19}
400 \DeclareTextAccent{.}{OT1}{95}
401 \DeclareTextAccent{=}{OT1}{22}
402 \DeclareTextAccent{^}{OT1}{94}
403 \DeclareTextAccent{\~}{OT1}{18}
404 \DeclareTextAccent{\-}{OT1}{126}
405 \DeclareTextAccent{H}{OT1}{125}
406 \DeclareTextAccent{u}{OT1}{21}
407 \DeclareTextAccent{v}{OT1}{20}
408 \DeclareTextAccent{r}{OT1}{23}

```

Some accents have to be built by hand: Note that `\oalign` and `\o@lign` must be inside a group. In these definitions we no longer use the helper function `\sh@ft` from `plain.tex` since that now has two incompatible definitions.

```

409 \DeclareTextCommand{b}{OT1}[1]
410   {\hmode@bgroup\o@lign{\relax#1\cr cr\hidewidth\ltx@sh@ft{-3ex}%
411     \vbox to.2ex{\hbox{\char22}\vss}\hidewidth}\egroup}
412 \DeclareTextCommand{c}{OT1}[1]
413   {\leavevmode\setbox\z@\hbox{#1}\ifdim\ht\z@=1ex\accent24 #1%
414     \else\oalign{\unhbox\z@\cr cr\hidewidth\char24\hidewidth}}\fi}
415 \DeclareTextCommand{d}{OT1}[1]
416   {\hmode@bgroup
417     \o@lign{\relax#1\cr cr\hidewidth\ltx@sh@ft{-1ex}.\hidewidth}\egroup}

```

Declare the text symbols.

```

418 \DeclareTextSymbol{\AE}{OT1}{29}
419 \DeclareTextSymbol{\OE}{OT1}{30}
420 \DeclareTextSymbol{\O}{OT1}{31}
421 \DeclareTextSymbol{\ae}{OT1}{26}
422 \DeclareTextSymbol{\i}{OT1}{16}
423 \DeclareTextSymbol{\j}{OT1}{17}
424 \DeclareTextSymbol{\oe}{OT1}{27}
425 \DeclareTextSymbol{\o}{OT1}{28}
426 \DeclareTextSymbol{\ss}{OT1}{25}
427 \DeclareTextSymbol{\textendash}{OT1}{124}
428 \DeclareTextSymbol{\textendash}{OT1}{123}

```

The `\nobreak\hskip\z@` is there to prevent a break after the hyphen but allow later breaks in the remainder of the word.

```

429 \DeclareTextCommand{\textnonbreakinghyphen}{OT1}{\mbox{-}\nobreak\hskip\z@}
430 \DeclareTextCommand{\textfiguredash}{OT1}{\textendash}
431 \DeclareTextCommand{\texthorizontalbar}{OT1}{\textendash}

```

Using the ligatures helps with OT1 fonts that have `\textexclamdown` and `\textquestiondown` in unusual positions.

```

432 %\DeclareTextSymbol{\textexclamdown}{OT1}{60}
433 %\DeclareTextSymbol{\textquestiondown}{OT1}{62}
434 \DeclareTextCommand{\textexclamdown}{OT1}{!'}
435 \DeclareTextCommand{\textquestiondown}{OT1}{?'}
436 %\DeclareTextSymbol{\texthyphenchar}{OT1}{'\-}
437 %\DeclareTextSymbol{\texthyphen}{OT1}{'\-}

```

```

438 \DeclareTextSymbol{\textquotedblleft}{OT1}{92}
439 \DeclareTextSymbol{\textquotedblright}{OT1}{'\'}
440 \DeclareTextSymbol{\textquoteleft}{OT1}{'\'}
441 \DeclareTextSymbol{\textquoteright}{OT1}{'\'}

```

Some symbols which are faked from others:

```

442 % \DeclareTextCommand{\aa}{OT1}
443 %   {{\accent23a}}
444 \DeclareTextCommand{\L}{OT1}
445   {\leavevmode\setbox\z@\hbox{L}\hbext@\wd\z@\hss\@xxxii L}}
446 \DeclareTextCommand{\l}{OT1}
447   {\hmode@bgroup\@xxxii l\egroup}
448 % \DeclareTextCommand{\AA}{OT1}
449 %   {\leavevmode\setbox\z@\hbox{h}\dimen@ht\z@\advance\dimen@-1ex%
450 %     \rlap{\raise.67\dimen@\hbox{\char23}}A}

```

In the OT1 encoding Å has a hand-crafted definition, so we have here the first recorded explicit use of `\DeclareTextCompositeCommand`.

```

451 \DeclareTextCompositeCommand{\r}{OT1}{A}
452   {\leavevmode\setbox\z@\hbox{!}\dimen@ht\z@\advance\dimen@-1ex%
453   \rlap{\raise.67\dimen@\hbox{\char23}}A}

```

The dutch language uses the letter ‘ij’. It is available in T1 encoded fonts, but not in the OT1 encoded fonts. Therefore we fake it for the OT1 encoding.

```

454 \DeclareTextCommand{\ij}{OT1}{%
455   \nobreak\hskip\z@skip i\kern-0.02em\nobreak\hskip\z@skip j}
456 \DeclareTextCommand{\IJ}{OT1}{%
457   \nobreak\hskip\z@skip I\kern-0.02em\nobreak\hskip\z@skip J}

```

In the OT1 encoding, £ and \$ share a slot.

```

458 \DeclareTextCommand{\textdollar}{OT1}{\hmode@bgroup
459   \ifdim \fontdimen\@ne\font >\z@
460     \slshape
461   \else
462     \upshape
463   \fi
464   \char'\$\egroup}

465 \DeclareTextCommand{\textsterling}{OT1}{\hmode@bgroup
466   \ifdim \fontdimen\@ne\font >\z@
467     \itshape
468   \else
469     \fontshape{ui}\selectfont
470   \fi
471   \char'\$\egroup}

```

Here we are adding some more composite commands to the OT1 encoding. This makes the use of certain accents with `i` compatible with their use with the T1 encoding; this enables them to become true \LaTeX internal representations. However, it will make these accents work a little less fast since a check will always be made for the existence of a composite.

```

472 \DeclareTextComposite{\.}{OT1}{i}{'\i}
473 \DeclareTextComposite{\.}{OT1}{\i}{'\i}
474 \DeclareTextCompositeCommand{\'}{OT1}{i}{\@tabacckludge'\i}
475 \DeclareTextCompositeCommand{\'}{OT1}{\i}{\@tabacckludge'\i}
476 \DeclareTextCompositeCommand{\^}{OT1}{i}{'\^i}

```

```

477 \DeclareTextCompositeCommand{\"}{OT1}{i}{\i}
478 \DeclareTextCompositeCommand{\=} {OT1}{i}{\=\i}

```

T1 encoding is given more extensive set of overloads for \c But here we just adjust \c{g}.

```

479 \ifx\textcommaabove\undefined\else
480 \DeclareTextCompositeCommand{\c}{OT1}{g}{\textcommaabove{g}}
481 \fi
482 \</OT1>

```

1.6 Definitions for the T1 encoding

The definitions for the ‘Extended T_EX text’ (T1) encoding.

Declare the encoding.

```

483 (*T1)
484 \DeclareFontEncoding{T1}{}{}

```

Declare the accents.

```

485 \DeclareTextAccent{\`}{T1}{0}
486 \DeclareTextAccent{\'}{T1}{1}
487 \DeclareTextAccent{\~}{T1}{2}
488 \DeclareTextAccent{\-}{T1}{3}
489 \DeclareTextAccent{\"}{T1}{4}
490 \DeclareTextAccent{\H}{T1}{5}
491 \DeclareTextAccent{\r}{T1}{6}
492 \DeclareTextAccent{\v}{T1}{7}
493 \DeclareTextAccent{\u}{T1}{8}
494 \DeclareTextAccent{\=}{T1}{9}
495 \DeclareTextAccent{\.}{T1}{10}

```

Some accents have to be built by hand. Note that \ooalign and \o@lign must be inside a group. In these definitions we no longer use the helper function \sh@ft from plain.tex since that now has two incompatible definitions.

```

496 \DeclareTextCommand{\b}{T1}[1]
497   {\hmode@bgroup\o@lign{\relax#1\crrc\hidewidth\ltx@sh@ft{-3ex}%
498   \vbox to.2ex{\hbox{\char9}\vss}\hidewidth}\egroup}
499 \DeclareTextCommand{\c}{T1}[1]
500   {\leavevmode\setbox\z@\hbox{#1}\ifdim\ht\z@=1ex\accent11 #1%
501   \else{\ooalign{\unhbox\z@\crrc
502   \hidewidth\char11\hidewidth}}\fi}
503 \DeclareTextCommand{\d}{T1}[1]
504   {\hmode@bgroup
505   \o@lign{\relax#1\crrc\hidewidth\ltx@sh@ft{-1ex}.\hidewidth}\egroup}
506 \DeclareTextCommand{\k}{T1}[1]
507   {\hmode@bgroup\ooalign{\null#1\crrc\hidewidth\char12}\egroup}
508 \DeclareTextCommand{\textogonekcentered}{T1}[1]
509   {\hmode@bgroup\ooalign{%
510   \null#1\crrc\hidewidth\char12\hidewidth}\egroup}

```

Some symbols are constructed.

Slot 24 contains a small circle intended for construction of these two glyphs.

```

511 \DeclareTextCommand{\textperthousand}{T1}
512   {\%\char 24 } % space or ‘relax as delimiter?’
513 \DeclareTextCommand{\textpertenthousand}{T1}
514   {\%\char 24\char 24 } % space or ‘relax as delimiter?’

```


For Maltese, `\Hwithstroke` and `\hwithstroke` are needed.

```

515 \DeclareTextCommand{\Hwithstroke}{T1}
516   {%
517     \hmode@bgroup
518     \vphantom{H}%
519     \sbox\z@{H}%
520     \oalign{%
521       H\cr
522       \hidewidth
523       \vrule
524         height \dimexpr 0.7\ht\z@+0.1ex\relax
525         depth  -0.7\ht\z@
526         width  0.8\wd\z@
527       \hidewidth\cr
528     }%
529   \egroup
530 }
531 \DeclareTextCommand{\hwithstroke}{T1}
532   {%
533     \hmode@bgroup
534     \vphantom{h}%
535     \sbox\z@{h}%
536     \oalign{%
537       h\cr
538       \kern0.075\wd\z@
539       \vrule
540         height \dimexpr 0.7\ht\z@+0.1ex\relax
541         depth  -0.7\ht\z@
542         width  0.4\wd\z@
543       \hidewidth\cr
544     }%
545   \egroup
546 }

```

Declare the text symbols.

```

547 %\DeclareTextSymbol{\AA}{T1}{197}
548 \DeclareTextSymbol{\AE}{T1}{198}
549 \DeclareTextSymbol{\DH}{T1}{208}
550 \DeclareTextSymbol{\DJ}{T1}{208}
551 \DeclareTextSymbol{\L}{T1}{138}
552 \DeclareTextSymbol{\NG}{T1}{141}
553 \DeclareTextSymbol{\OE}{T1}{215}
554 \DeclareTextSymbol{\O}{T1}{216}
555 \DeclareTextSymbol{\SS}{T1}{223}
556 \DeclareTextSymbol{\TH}{T1}{222}
557 %\DeclareTextSymbol{\aa}{T1}{229}
558 \DeclareTextSymbol{\ae}{T1}{230}
559 \DeclareTextSymbol{\dh}{T1}{240}
560 \DeclareTextSymbol{\dj}{T1}{158}

561 \DeclareTextSymbol{\guillemetleft}{T1}{19}
562 \DeclareTextSymbol{\guillemetright}{T1}{20}
563 % old Adobe names
564 \DeclareTextSymbol{\guillemotleft}{T1}{19}

```

```

565 \DeclareTextSymbol{\guillemotright}{T1}{20}
566 \DeclareTextSymbol{\guilsinglleft}{T1}{14}
567 \DeclareTextSymbol{\guilsinglright}{T1}{15}
568 \DeclareTextSymbol{\i}{T1}{25}
569 \DeclareTextSymbol{\j}{T1}{26}
570 \DeclareTextSymbol{\ij}{T1}{188}
571 \DeclareTextSymbol{\IJ}{T1}{156}
572 \DeclareTextSymbol{\l}{T1}{170}
573 \DeclareTextSymbol{\ng}{T1}{173}
574 \DeclareTextSymbol{\oe}{T1}{247}
575 \DeclareTextSymbol{\o}{T1}{248}
576 \DeclareTextSymbol{\quotedblbase}{T1}{18}
577 \DeclareTextSymbol{\quotesinglbase}{T1}{13}
578 \DeclareTextSymbol{\ss}{T1}{255}
579 \DeclareTextSymbol{\textasciicircum}{T1}{`\~}
580 \DeclareTextSymbol{\textasciitilde}{T1}{`\~}
581 \DeclareTextSymbol{\textbackslash}{T1}{`\}
582 \DeclareTextSymbol{\textbar}{T1}{`\|}
583 \DeclareTextSymbol{\textbraceleft}{T1}{`\{ }
584 \DeclareTextSymbol{\textbraceright}{T1}{`\} }
585 \DeclareTextSymbol{\textcompwordmark}{T1}{23}
586 \DeclareTextSymbol{\textdollar}{T1}{`\$}
587 \DeclareTextSymbol{\textendash}{T1}{22}
588 \DeclareTextSymbol{\textendash}{T1}{21}

```

The `\nobreak\hskip\z@` is there to prevent a break after the hyphen but allow later breaks in the remainder of the word.

```

589 \DeclareTextCommand{\textnonbreakinghyphen}{T1}{\mbox{-}\nobreak\hskip\z@}
590 \DeclareTextCommand{\textfiguredash}{T1}{\textendash}
591 \DeclareTextCommand{\texthorizontalbar}{T1}{\textendash}
592 \DeclareTextSymbol{\textexclamdown}{T1}{189}
593 \DeclareTextSymbol{\textgreater}{T1}{`\>}
594 %\DeclareTextSymbol{\texthyphenchar}{T1}{127}
595 %\DeclareTextSymbol{\texthyphen}{T1}{`\-}
596 \DeclareTextSymbol{\textless}{T1}{`\<}
597 \DeclareTextSymbol{\textquestiondown}{T1}{190}
598 \DeclareTextSymbol{\textquotedblleft}{T1}{16}
599 \DeclareTextSymbol{\textquotedblright}{T1}{17}
600 \DeclareTextSymbol{\textquotedbl}{T1}{`"}
601 \DeclareTextSymbol{\textquoteleft}{T1}{`\'}
602 \DeclareTextSymbol{\textquoteright}{T1}{`\'}
603 \DeclareTextSymbol{\textsection}{T1}{159}
604 \DeclareTextSymbol{\textsterling}{T1}{191}
605 \DeclareTextSymbol{\textunderscore}{T1}{95}
606 \DeclareTextSymbol{\textvisiblespace}{T1}{32}
607 \DeclareTextSymbol{\th}{T1}{254}

```

Declare the composites.

```

608 \DeclareTextComposite{\.}{T1}{i}{`\i}
609 \DeclareTextComposite{\.}{T1}{i}{`\i}
"80 = 128
610 \DeclareTextComposite{\u}{T1}{A}{128}
611 \DeclareTextComposite{\k}{T1}{A}{129}

```

```

612 \DeclareTextComposite{\'}{T1}{C}{130}
613 \DeclareTextComposite{\v}{T1}{C}{131}
614 \DeclareTextComposite{\v}{T1}{D}{132}
615 \DeclareTextComposite{\v}{T1}{E}{133}
616 \DeclareTextComposite{\k}{T1}{E}{134}
617 \DeclareTextComposite{\u}{T1}{G}{135}
"88 = 136
618 \DeclareTextComposite{\'}{T1}{L}{136}
619 \DeclareTextComposite{\v}{T1}{L}{137}
620 \DeclareTextComposite{\'}{T1}{N}{139}
621 \DeclareTextComposite{\v}{T1}{N}{140}
622 \DeclareTextComposite{\H}{T1}{O}{142}
623 \DeclareTextComposite{\'}{T1}{R}{143}
"90 = 144
624 \DeclareTextComposite{\v}{T1}{R}{144}
625 \DeclareTextComposite{\'}{T1}{S}{145}
626 \DeclareTextComposite{\v}{T1}{S}{146}
627 \DeclareTextComposite{\c}{T1}{S}{147}
628 \DeclareTextComposite{\v}{T1}{T}{148}
629 \DeclareTextComposite{\c}{T1}{T}{149}
630 \DeclareTextComposite{\H}{T1}{U}{150}
631 \DeclareTextComposite{\r}{T1}{U}{151}
"98 = 152
632 \DeclareTextComposite{\"}{T1}{Y}{152}
633 \DeclareTextComposite{\'}{T1}{Z}{153}
634 \DeclareTextComposite{\v}{T1}{Z}{154}
635 \DeclareTextComposite{\.}{T1}{Z}{155}
636 \DeclareTextComposite{\.}{T1}{I}{157}
"A0 = 160
637 \DeclareTextComposite{\u}{T1}{a}{160}
638 \DeclareTextComposite{\k}{T1}{a}{161}
639 \DeclareTextComposite{\'}{T1}{c}{162}
640 \DeclareTextComposite{\v}{T1}{c}{163}
641 \DeclareTextComposite{\v}{T1}{d}{164}
642 \DeclareTextComposite{\v}{T1}{e}{165}
643 \DeclareTextComposite{\k}{T1}{e}{166}
644 \DeclareTextComposite{\u}{T1}{g}{167}
"A8 = 168
645 \DeclareTextComposite{\'}{T1}{l}{168}
646 \DeclareTextComposite{\v}{T1}{l}{169}
647 \DeclareTextComposite{\'}{T1}{n}{171}
648 \DeclareTextComposite{\v}{T1}{n}{172}
649 \DeclareTextComposite{\H}{T1}{o}{174}
650 \DeclareTextComposite{\'}{T1}{r}{175}
"B0 = 176
651 \DeclareTextComposite{\v}{T1}{r}{176}
652 \DeclareTextComposite{\'}{T1}{s}{177}
653 \DeclareTextComposite{\v}{T1}{s}{178}
654 \DeclareTextComposite{\c}{T1}{s}{179}
655 \DeclareTextComposite{\v}{T1}{t}{180}

```

```

656 \DeclareTextComposite{\c}{T1}{t}{181}
657 \DeclareTextComposite{\H}{T1}{u}{182}
658 \DeclareTextComposite{\r}{T1}{u}{183}
"B8 = 184
659 \DeclareTextComposite{\"}{T1}{y}{184}
660 \DeclareTextComposite{\'}{T1}{z}{185}
661 \DeclareTextComposite{\v}{T1}{z}{186}
662 \DeclareTextComposite{\.}{T1}{z}{187}
"C0 = 192
663 \DeclareTextComposite{\'}{T1}{A}{192}
664 \DeclareTextComposite{\'}{T1}{A}{193}
665 \DeclareTextComposite{\^}{T1}{A}{194}
666 \DeclareTextComposite{\~}{T1}{A}{195}
667 \DeclareTextComposite{\"}{T1}{A}{196}
668 \DeclareTextComposite{\r}{T1}{A}{197}
669 \DeclareTextComposite{\c}{T1}{C}{199}
"C8 = 200
670 \DeclareTextComposite{\'}{T1}{E}{200}
671 \DeclareTextComposite{\'}{T1}{E}{201}
672 \DeclareTextComposite{\^}{T1}{E}{202}
673 \DeclareTextComposite{\"}{T1}{E}{203}
674 \DeclareTextComposite{\'}{T1}{I}{204}
675 \DeclareTextComposite{\'}{T1}{I}{205}
676 \DeclareTextComposite{\^}{T1}{I}{206}
677 \DeclareTextComposite{\"}{T1}{I}{207}
"D0 = 208
678 \DeclareTextComposite{\~}{T1}{N}{209}
679 \DeclareTextComposite{\'}{T1}{O}{210}
680 \DeclareTextComposite{\'}{T1}{O}{211}
681 \DeclareTextComposite{\^}{T1}{O}{212}
682 \DeclareTextComposite{\~}{T1}{O}{213}
683 \DeclareTextComposite{\"}{T1}{O}{214}
"D8 = 216
684 \DeclareTextComposite{\'}{T1}{U}{217}
685 \DeclareTextComposite{\'}{T1}{U}{218}
686 \DeclareTextComposite{\^}{T1}{U}{219}
687 \DeclareTextComposite{\"}{T1}{U}{220}
688 \DeclareTextComposite{\'}{T1}{Y}{221}
"E0 = 224
689 \DeclareTextComposite{\'}{T1}{a}{224}
690 \DeclareTextComposite{\'}{T1}{a}{225}
691 \DeclareTextComposite{\^}{T1}{a}{226}
692 \DeclareTextComposite{\~}{T1}{a}{227}
693 \DeclareTextComposite{\"}{T1}{a}{228}
694 \DeclareTextComposite{\r}{T1}{a}{229}
695 \DeclareTextComposite{\c}{T1}{c}{231}
"E8 = 232
696 \DeclareTextComposite{\'}{T1}{e}{232}
697 \DeclareTextComposite{\'}{T1}{e}{233}
698 \DeclareTextComposite{\^}{T1}{e}{234}

```

```

699 \DeclareTextComposite{"}{T1}{e}{235}
700 \DeclareTextComposite{'}{T1}{i}{236}
701 \DeclareTextComposite{'}{T1}{\i}{236}
702 \DeclareTextComposite{'}{T1}{i}{237}
703 \DeclareTextComposite{'}{T1}{\i}{237}
704 \DeclareTextComposite{^}{T1}{i}{238}
705 \DeclareTextComposite{^}{T1}{\i}{238}
706 \DeclareTextComposite{"}{T1}{i}{239}
707 \DeclareTextComposite{"}{T1}{\i}{239}

"F0 = 240

708 \DeclareTextComposite{~}{T1}{n}{241}
709 \DeclareTextComposite{'}{T1}{o}{242}
710 \DeclareTextComposite{'}{T1}{o}{243}
711 \DeclareTextComposite{^}{T1}{o}{244}
712 \DeclareTextComposite{~}{T1}{o}{245}
713 \DeclareTextComposite{"}{T1}{o}{246}

"F8 = 248

714 \DeclareTextComposite{'}{T1}{u}{249}
715 \DeclareTextComposite{'}{T1}{u}{250}
716 \DeclareTextComposite{^}{T1}{u}{251}
717 \DeclareTextComposite{"}{T1}{u}{252}
718 \DeclareTextComposite{'}{T1}{y}{253}

719 \DeclareTextCompositeCommand{\k}{T1}{o}{\textogonekcentered{o}}
720 \DeclareTextCompositeCommand{\k}{T1}{0}{\textogonekcentered{0}}

721 \ifx\textcommaabove@undefined\else
722 \DeclareTextCompositeCommand{\c}{T1}{g}{\textcommaabove{g}}
723 \fi
724 \ifx\textcommabelow@undefined\else
725 \DeclareTextCompositeCommand{\c}{T1}{G}{\textcommabelow{G}}
726 \DeclareTextCompositeCommand{\c}{T1}{K}{\textcommabelow{K}}
727 \DeclareTextCompositeCommand{\c}{T1}{k}{\textcommabelow{k}}
728 \DeclareTextCompositeCommand{\c}{T1}{L}{\textcommabelow{L}}
729 \DeclareTextCompositeCommand{\c}{T1}{l}{\textcommabelow{l}}
730 \DeclareTextCompositeCommand{\c}{T1}{N}{\textcommabelow{N}}
731 \DeclareTextCompositeCommand{\c}{T1}{n}{\textcommabelow{n}}
732 \DeclareTextCompositeCommand{\c}{T1}{R}{\textcommabelow{R}}
733 \DeclareTextCompositeCommand{\c}{T1}{r}{\textcommabelow{r}}
734 \fi

    One oddity.

735 \DeclareTextCompositeCommand{\=}{T1}{i}{\=\i}
736 </T1>

```

1.7 Definitions for the OMS encoding

The definitions for the ‘ \TeX math symbol’ (OMS) encoding. Even though this is meant to be a math font, it includes some of the standard \LaTeX text symbols.

Declare the encoding.

```

737 <*OMS>
738 \DeclareFontEncoding{OMS}{-}{-}

```

Declare the symbols. Note that slot 13 has in places been named `\Orb`: please root out and destroy this impolity wherever you find it!

```

739 \DeclareTextSymbol{\textasteriskcentered}{OMS}{3} % "03
740 \DeclareTextSymbol{\textbackslash}{OMS}{110} % "6E
741 \DeclareTextSymbol{\textbar}{OMS}{106} % "6A
742 \DeclareTextSymbol{\textbardbl}{OMS}{107} % "6B
743 \DeclareTextSymbol{\textbraceleft}{OMS}{102} % "66
744 \DeclareTextSymbol{\textbraceright}{OMS}{103} % "67
745 \DeclareTextSymbol{\textbullet}{OMS}{15} % "0F
746 \DeclareTextSymbol{\textdaggerdbl}{OMS}{122} % "7A
747 \DeclareTextSymbol{\textdagger}{OMS}{121} % "79
748 \DeclareTextSymbol{\textparagraph}{OMS}{123} % "7B
749 \DeclareTextSymbol{\textperiodcentered}{OMS}{1} % "01
750 \DeclareTextSymbol{\textsection}{OMS}{120} % "78
751 \DeclareTextSymbol{\textbigcircle}{OMS}{13} % "0D
752 \DeclareTextCommand{\textcircled}{OMS}[1]{\hmode@bgroup
753   \oalign{%
754     \hfil \raise .07ex\hbox {\upshape#1}\hfil \crcr
755     \char 13 % "0D
756   }%
757   \egroup}
758 </OMS>

```

1.8 Definitions for the OML encoding

The definitions for the ‘ \TeX math italic’ (OML) encoding. Even though this is meant to be a math font, it includes some of the standard \LaTeX text symbols.

Declare the encoding.

```

759 <*OML>
760 \DeclareFontEncoding{OML}{}{}

```

Declare the symbols.

```

761 \DeclareTextSymbol{\textless}{OML}{'\<}
762 \DeclareTextSymbol{\textgreater}{OML}{'\>}
763 \DeclareTextAccent{\t}{OML}{127} % "7F
764 </OML>

```

1.9 Definitions for the OT4 encoding

These definitions are for the Polish extension to the ‘ \TeX text’ (OT1) encoding. This encoding was created by B. Jackowski and M. Ryćko for use with the Polish version of Computer Modern and Computer Concrete. In positions 0–127 it is identical to OT1 but it contains some additional characters in the upper half. The \LaTeX support was developed by Mariusz Olko.

The PL fonts that use it are available as follows:

Metafont sources <ftp://ftp.gust.org.pl/TeX/language/polish/pl-mf.zip>;

Font files <ftp://ftp.gust.org.pl/TeX/language/polish/pl-tfm.zip>.

Declare the encoding.

```

765 <*OT4>
766 \DeclareFontEncoding{OT4}{}{}
767 \DeclareFontSubstitution{OT4}{cmr}{m}{n}

```

Declare the accents.

```
768 \DeclareTextAccent{"}{OT4}{127}
769 \DeclareTextAccent{'}{OT4}{19}
770 \DeclareTextAccent{.}{OT4}{95}
771 \DeclareTextAccent{=}{OT4}{22}
772 \DeclareTextAccent{^}{OT4}{94}
773 \DeclareTextAccent{'}{OT4}{18}
774 \DeclareTextAccent{~}{OT4}{126}
775 \DeclareTextAccent{H}{OT4}{125}
776 \DeclareTextAccent{u}{OT4}{21}
777 \DeclareTextAccent{v}{OT4}{20}
778 \DeclareTextAccent{r}{OT4}{23}
```

The ogonek accent is available only under a e A & E. But we have to provide some definition for \k. Some other accents have to be built by hand as in OT1:

```
779 \DeclareTextCommand{\k}{OT4}[1]{%
780   \TextSymbolUnavailable{\k{#1}}#1}
```

In these definitions we no longer use the helper function \sh@ft from plain.tex since that now has two incompatible definitions.

```
781 \DeclareTextCommand{\b}{OT4}[1]
782   {\hmode@bgroup\o@lign{\relax#1\crrc\hidewidth\ltx@sh@ft{-3ex}%
783     \vbox to.2ex{\hbox{\char22}\vss}\hidewidth}\egroup}
784 \DeclareTextCommand{\c}{OT4}[1]
785   {\leavevmode\setbox\z@\hbox{#1}\ifdim\ht\z@=1ex\accent24 #1%
786     \else{\oalign{\unhbox\z@\crrc\hidewidth\char24\hidewidth}}\fi}
787 \DeclareTextCommand{\d}{OT4}[1]
788   {\hmode@bgroup
789     \o@lign{\relax#1\crrc\hidewidth\ltx@sh@ft{-1ex}.\hidewidth}\egroup}
```

Declare the text symbols.

```
790 \DeclareTextSymbol{\AE}{OT4}{29}
791 \DeclareTextSymbol{\OE}{OT4}{30}
792 \DeclareTextSymbol{\O}{OT4}{31}
793 \DeclareTextSymbol{\L}{OT4}{138}
794 \DeclareTextSymbol{\ae}{OT4}{26}

795 \DeclareTextSymbol{\guillemetleft}{OT4}{174}
796 \DeclareTextSymbol{\guillemetright}{OT4}{175}
797 % old Adobe names
798 \DeclareTextSymbol{\guillemotleft}{OT4}{174}
799 \DeclareTextSymbol{\guillemotright}{OT4}{175}

800 \DeclareTextSymbol{\i}{OT4}{16}
801 \DeclareTextSymbol{\j}{OT4}{17}
802 \DeclareTextSymbol{\l}{OT4}{170}
803 \DeclareTextSymbol{\o}{OT4}{28}
804 \DeclareTextSymbol{\oe}{OT4}{27}
805 \DeclareTextSymbol{\quotedblbase}{OT4}{255}
806 \DeclareTextSymbol{\ss}{OT4}{25}
807 \DeclareTextSymbol{\textendash}{OT4}{124}
808 \DeclareTextSymbol{\textendash}{OT4}{123}
809 \DeclareTextSymbol{\textexclamdown}{OT4}{60}
810 %\DeclareTextSymbol{\textthyphenchar}{OT4}{'\-}
811 %\DeclareTextSymbol{\textthyphen}{OT4}{'\-}
812 \DeclareTextSymbol{\textquestiondown}{OT4}{62}
```

```

813 \DeclareTextSymbol{\textquotedblleft}{OT4}{92}
814 \DeclareTextSymbol{\textquotedblright}{OT4}{'\'}
815 \DeclareTextSymbol{\textquoteleft}{OT4}{'\'}
816 \DeclareTextSymbol{\textquoteright}{OT4}{'\'}

```

Definition for Å as in OT1:

```

817 \DeclareTextCompositeCommand{\r}{OT4}{A}
818   {\leavevmode\setbox\z@\hbox{!}\dimen@ht\z@\advance\dimen@-1ex%
819   \rlap{\raise.67\dimen@\hbox{\char23}}A}

```

In the OT4 encoding, £ and \$ share a slot.

```

820 \DeclareTextCommand{\textdollar}{OT4}{\hmode@bgroup
821   \ifdim \fontdimen\@ne\font >\z@
822     \slshape
823   \else
824     \upshape
825   \fi
826   \char'\$\egroup}
827 \DeclareTextCommand{\textsterling}{OT4}{\hmode@bgroup
828   \ifdim \fontdimen\@ne\font >\z@
829     \itshape
830   \else
831     \fontshape{ui}\selectfont
832   \fi
833   \char'\$\egroup}

```

Declare the composites.

```

834 \DeclareTextComposite{\k}{OT4}{A}{129}
835 \DeclareTextComposite{\'}{OT4}{C}{130}
836 \DeclareTextComposite{\k}{OT4}{E}{134}
837 \DeclareTextComposite{\'}{OT4}{N}{139}
838 \DeclareTextComposite{\'}{OT4}{S}{145}
839 \DeclareTextComposite{\'}{OT4}{Z}{153}
840 \DeclareTextComposite{\.}{OT4}{Z}{155}
841 \DeclareTextComposite{\k}{OT4}{a}{161}
842 \DeclareTextComposite{\'}{OT4}{c}{162}
843 \DeclareTextComposite{\k}{OT4}{e}{166}
844 \DeclareTextComposite{\'}{OT4}{n}{171}
845 \DeclareTextComposite{\'}{OT4}{s}{177}
846 \DeclareTextComposite{\'}{OT4}{z}{185}
847 \DeclareTextComposite{\.}{OT4}{z}{187}
848 \DeclareTextComposite{\'}{OT4}{0}{211}
849 \DeclareTextComposite{\'}{OT4}{o}{243}
850 </OT4>

```

1.10 Definitions for the TS1 encoding

```

851 <*TS1>
852 \DeclareFontEncoding{TS1}{}{}
853 \DeclareFontSubstitution{TS1}{cmr}{m}{n}

```

Some accents have to be built by hand. Note that \ooalign and \o@lign must be inside a group.

```

854 \DeclareTextCommand{\capitalcedilla}{TS1}[1]
855   {\hmode@bgroup

```



```

856 \ooalign{\null#1\crrc\hidewidth\char11\hidewidth}\egroup}
857 \DeclareTextCommand{\capitalogonek}{TS1}[1]
858   {\hmode@bgroup
859   \ooalign{\null#1\crrc\hidewidth\char12\hidewidth}\egroup}

```

Accents for capital letters.

These commands can be used by the end user either directly or through definitions of the type

```
\DeclareTextCompositeCommand{\'}{T1}{X}{\capitalacute X}
```

None of the latter definitions are provided by default, since they are probably rarely used.

"00 = 0

```

860 \DeclareTextAccent{\capitalgrave}{TS1}{0}
861 \DeclareTextAccent{\capitalacute}{TS1}{1}
862 \DeclareTextAccent{\capitalcircumflex}{TS1}{2}
863 \DeclareTextAccent{\capitaltilde}{TS1}{3}
864 \DeclareTextAccent{\capitaldieresis}{TS1}{4}
865 \DeclareTextAccent{\capitalhungarumlaut}{TS1}{5}
866 \DeclareTextAccent{\capitalring}{TS1}{6}
867 \DeclareTextAccent{\capitalcaron}{TS1}{7}

```

"08 = 8

```

868 \DeclareTextAccent{\capitalbreve}{TS1}{8}
869 \DeclareTextAccent{\capitalmacron}{TS1}{9}
870 \DeclareTextAccent{\capitaldotaccent}{TS1}{10}

```

Tie accents.

The tie accent was borrowed from the `cmmi` font. The `tc` fonts now provide four tie accents, the first two are done in the classical way with asymmetric glyphs hanging out of their boxes; the new ties are centered in their boxes like all other accents. They need a name: please tell us if you know what to call them.

" =

```

871 \DeclareTextAccent{\t}{TS1}{26}
872 \DeclareTextAccent{\capitaltie}{TS1}{27}
873 \DeclareTextAccent{\newtie}{TS1}{28}
874 \DeclareTextAccent{\capitalnewtie}{TS1}{29}

```

Compound word marks.

The text companion fonts contain two compound word marks of different heights, one has `cap_height`, the other `asc_height`.

```

875 \DeclareTextSymbol{\textcapitalcompwordmark}{TS1}{23}
876 \DeclareTextSymbol{\textascendercompwordmark}{TS1}{31}

```

The text companion symbols.

```
877 \DeclareTextSymbol{\textquotestraightbase}{TS1}{13}
```

"10 = 16

```

878 \DeclareTextSymbol{\textquotestraightdblbase}{TS1}{18}
879 \DeclareTextSymbol{\texttwelveudash}{TS1}{21}
880 \DeclareTextSymbol{\textthreequartersemdash}{TS1}{22}

```

"18 = 24

```

881 \DeclareTextSymbol{\textleftarrow}{TS1}{24}
882 \DeclareTextSymbol{\textrightarrow}{TS1}{25}

```

"20 = 32

```
883 \DeclareTextSymbol{\textblank}{TS1}{32}
884 \DeclareTextSymbol{\textdollar}{TS1}{36}
885 \DeclareTextSymbol{\textquotesingle}{TS1}{39}
```

"28 = 40

The symbol `\textasteriskcentered` “*” is supposed to be always available in TS1 and that is important as it is used in footnote symbols. However, in a few fonts it is missing even though they are otherwise fairly complete. We therefore use a rather elaborate method and check if the slot has a glyph and if not produce a poor man’s version by using a normal “*” slightly enlarged and somewhat lowered. The main application for this symbol is in footnote symbols and there it should produce a comparable size and show a similar placement.

```
886 %\DeclareTextSymbol{\textasteriskcentered}{TS1}{42} % that’s wanted
887 \DeclareTextCommand \textasteriskcentered{TS1}{% % and that’s needed
888 \iffontchar\font 42 \char42 \else
889 \begingroup\fontencoding{T1}%
890 \fontsize
891 {\the\dimexpr1.3\dimexpr\fontsize pt\relax}%
892 {\font@baselineskip}%
893 \selectfont
894 \raisebox{-0.7ex}{\dimexpr\height-0.7ex}[0pt]{*}%
895 \endgroup
896 \fi
897 }
```

Note that ’054 is a comma and ’056 is a full stop: these make numbers using oldstyle digits easier to input.

```
898 \DeclareTextSymbol{\textdblhyphen}{TS1}{45}
899 \DeclareTextSymbol{\textfractionsolidus}{TS1}{47}
```

Oldstyle digits.

"30 = 48

```
900 \DeclareTextSymbol{\textzerooldstyle}{TS1}{48}
901 \DeclareTextSymbol{\textoneoldstyle}{TS1}{49}
902 \DeclareTextSymbol{\texttwooldstyle}{TS1}{50}
903 \DeclareTextSymbol{\textthreeoldstyle}{TS1}{51}
904 \DeclareTextSymbol{\textfouroldstyle}{TS1}{52}
905 \DeclareTextSymbol{\textfiveoldstyle}{TS1}{53}
906 \DeclareTextSymbol{\textsixoldstyle}{TS1}{54}
907 \DeclareTextSymbol{\textsevenoldstyle}{TS1}{55}
```

"38 = 56

```
908 \DeclareTextSymbol{\texteightoldstyle}{TS1}{56}
909 \DeclareTextSymbol{\textnineoldstyle}{TS1}{57}
```

More text companion symbols.

```
910 \DeclareTextSymbol{\textlangle}{TS1}{60}
911 \DeclareTextSymbol{\textminus}{TS1}{61}
912 \DeclareTextSymbol{\textrightangle}{TS1}{62}
```

"48 = 72

```
913 \DeclareTextSymbol{\textmho}{TS1}{77}
```

The big circle is here to define the command `\textcircled`. Formerly it was taken from the `cmsy` font.

```

914 \DeclareTextSymbol{\textbigcircle}{TS1}{79}
915 \DeclareTextCommand{\textcircled}{TS1}[1]{\hmode@bgroup
916   \oalign{%
917     \hfil \raise .07ex\hbox {\upshape#1}\hfil \crcr
918     \char 79   % '117 = "4F
919   }%
920 \egroup}
    More text companion symbols.
    "50 = 80
921 \DeclareTextSymbol{\textohm}{TS1}{87}
"58 = 88
922 \DeclareTextSymbol{\textlbrackdbl}{TS1}{91}
923 \DeclareTextSymbol{\textrbrackdbl}{TS1}{93}
924 \DeclareTextSymbol{\textuparrow}{TS1}{94}
925 \DeclareTextSymbol{\textdownarrow}{TS1}{95}
"60 = 96
926 \DeclareTextSymbol{\textasciigrave}{TS1}{96}
927 \DeclareTextSymbol{\textborn}{TS1}{98}
928 \DeclareTextSymbol{\textdivorced}{TS1}{99}
929 \DeclareTextSymbol{\textdied}{TS1}{100}
"68 = 104
930 \DeclareTextSymbol{\textleaf}{TS1}{108}
931 \DeclareTextSymbol{\textmarried}{TS1}{109}
932 \DeclareTextSymbol{\textmusicalnote}{TS1}{110}
"78 = 120
933 \DeclareTextSymbol{\texttildebelow}{TS1}{126}
    This glyph, \textdblhyphenchar is hanging, like the hyphenchar of the ec fonts.
934 \DeclareTextSymbol{\textdblhyphenchar}{TS1}{127}
"80 = 128
935 \DeclareTextSymbol{\textasciibreve}{TS1}{128}
936 \DeclareTextSymbol{\textasciicaron}{TS1}{129}
    This next glyph is not the same as \textquotedbl.
937 \DeclareTextSymbol{\textacutedbl}{TS1}{130}
938 \DeclareTextSymbol{\textgravedbl}{TS1}{131}
939 \DeclareTextSymbol{\textdagger}{TS1}{132}
940 \DeclareTextSymbol{\textdaggerdbl}{TS1}{133}
941 \DeclareTextSymbol{\textbardbl}{TS1}{134}
942 \DeclareTextSymbol{\textperthousand}{TS1}{135}
"88 = 136
943 \DeclareTextSymbol{\textbullet}{TS1}{136}
944 \DeclareTextSymbol{\textcelsius}{TS1}{137}
945 \DeclareTextSymbol{\textdollaroldstyle}{TS1}{138}
946 \DeclareTextSymbol{\textcentoldstyle}{TS1}{139}
947 \DeclareTextSymbol{\textflorin}{TS1}{140}
948 \DeclareTextSymbol{\textcolonmonetary}{TS1}{141}
949 \DeclareTextSymbol{\textwon}{TS1}{142}
950 \DeclareTextSymbol{\textnaira}{TS1}{143}

```

"90 = 144

```
951 \DeclareTextSymbol{\textguarani}{TS1}{144}
952 \DeclareTextSymbol{\textpeso}{TS1}{145}
953 \DeclareTextSymbol{\textlira}{TS1}{146}
954 \DeclareTextSymbol{\textrecipe}{TS1}{147}
955 \DeclareTextSymbol{\textinterrobang}{TS1}{148}
956 \DeclareTextSymbol{\textinterrobangdown}{TS1}{149}
957 \DeclareTextSymbol{\textdong}{TS1}{150}
958 \DeclareTextSymbol{\texttrademark}{TS1}{151}
```

"98 = 152

```
959 \DeclareTextSymbol{\textpertenthousand}{TS1}{152}
960 \DeclareTextSymbol{\textpilcrow}{TS1}{153}
961 \DeclareTextSymbol{\textbaht}{TS1}{154}
962 \DeclareTextSymbol{\textnumero}{TS1}{155}
```

This next name may change. For the following sign we know only a german name, which is abzüglich. The meaning is something like “commercial minus”. An ASCII ersatz is ./ (dot slash dot). The temporary English name is `\textdiscount`.

```
963 \DeclareTextSymbol{\textdiscount}{TS1}{156}
964 \DeclareTextSymbol{\textestimated}{TS1}{157}
965 \DeclareTextSymbol{\textopenbullet}{TS1}{158}
966 \DeclareTextSymbol{\textservicemark}{TS1}{159}
```

"A0 = 160

```
967 \DeclareTextSymbol{\textlquill}{TS1}{160}
968 \DeclareTextSymbol{\textrquill}{TS1}{161}
969 \DeclareTextSymbol{\textcent}{TS1}{162}
970 \DeclareTextSymbol{\textsterling}{TS1}{163}
971 \DeclareTextSymbol{\textcurrency}{TS1}{164}
972 \DeclareTextSymbol{\textyen}{TS1}{165}
973 \DeclareTextSymbol{\textbrokenbar}{TS1}{166}
974 \DeclareTextSymbol{\textsection}{TS1}{167}
```

"A8 = 168

```
975 \DeclareTextSymbol{\textasciidieresis}{TS1}{168}
976 \DeclareTextSymbol{\textcopyright}{TS1}{169}
977 \DeclareTextSymbol{\textordfeminine}{TS1}{170}
978 \DeclareTextSymbol{\textcopyleft}{TS1}{171}
979 \DeclareTextSymbol{\textlnot}{TS1}{172}
```

The meaning of the circled-P is “sound recording copyright”.

```
980 \DeclareTextSymbol{\textcircledP}{TS1}{173}
981 \DeclareTextSymbol{\textregistered}{TS1}{174}
982 \DeclareTextSymbol{\textasciimacron}{TS1}{175}
```

"B0 = 176

```
983 \DeclareTextSymbol{\textdegree}{TS1}{176}
984 \DeclareTextSymbol{\textpm}{TS1}{177}
985 \DeclareTextSymbol{\texttwosuperior}{TS1}{178}
986 \DeclareTextSymbol{\textthreesuperior}{TS1}{179}
987 \DeclareTextSymbol{\textasciacute}{TS1}{180}
988 \DeclareTextSymbol{\textmu}{TS1}{181} % micro sign
989 \DeclareTextSymbol{\textparagraph}{TS1}{182}
990 \DeclareTextSymbol{\textperiodcentered}{TS1}{183}
```

```

"B8 = 184
991 \DeclareTextSymbol{\textreferencemark}{TS1}{184}
992 \DeclareTextSymbol{\textonesuperior}{TS1}{185}
993 \DeclareTextSymbol{\textordmasculine}{TS1}{186}
994 \DeclareTextSymbol{\textsurd}{TS1}{187}
995 \DeclareTextSymbol{\textonequarter}{TS1}{188}
996 \DeclareTextSymbol{\textonehalf}{TS1}{189}
997 \DeclareTextSymbol{\textthreequarters}{TS1}{190}
998 \DeclareTextSymbol{\texteuro}{TS1}{191}
"E0 = 208
999 \DeclareTextSymbol{\texttimes}{TS1}{214}
"F0 = 240
1000 \DeclareTextSymbol{\textdiv}{TS1}{246}
1001 \end{TS1}

```

1.11 Definitions for the TU encoding

The TU encoding was originally introduced in the contributed package `fontspec` as a Unicode encoding for XeTeX and LuaTeX.

Normally for these engines, the input consists of Unicode characters encoded in UTF-8. There is therefore little need to use the traditional (ASCII) encoding-specific commands

However, sometimes (e.g. for backwards compatibility) it can be useful to access these Unicode characters via such ASCII-based markup. The commands provided here cover the characters in the T1 and TS1 encodings, but specified in Unicode position. Almost all the command names have been mechanically extracted from the `inputenc` UTF-8 support, which is essentially doing a reverse mapping from UTF-8 data to L^AT_EX LICR commands.

A few additional names for character which were supported in the original `fontspec` version of this file have also been added, even though they are not currently in the default `inputenc` UTF-8 declarations.

```
1002 \*TU
```

In the base interface the Unicode encoding is always known as TU. But we parameterize the encoding name to allow for modelling differences in Unicode support by different fonts.

```
1003 \providecommand\UnicodeEncodingName{TU}
```

As the Unicode encoding, TU, is only currently available with XeTeX or LuaTeX, we detect these engines first, and make adjustments for the differing font loading syntax. For other engines, we issue a warning then abort this file, switching back to T1 encoding.

```

1004 \begingroup\expandafter\expandafter\expandafter\endgroup
1005 \expandafter\ifx\csname XeTeXrevision\endcsname\relax
1006   \begingroup\expandafter\expandafter\expandafter\endgroup
1007   \expandafter\ifx\csname directlua\endcsname\relax

```

Not LuaTeX or XeTeX, abort with a warning.

```

1008   \PackageWarningNoLine{fontenc}
1009     {\UnicodeEncodingName\space
1010      encoding is only available with XeTeX and LuaTeX.\MessageBreak
1011      Defaulting to T1 encoding}

```

```

1012     \def\encodingdefault{T1}
1013     \expandafter\expandafter\expandafter\endinput
1014 \else
    LuaTeX. For LuaTeX 1.10+, define a Lua function to disable any handling by the
    font code. Otherwise we reload the font without TeX ligatures.
1015     \def\UnicodeFontTeXLigatures{+tlig;}
1016     \ifnum\luatexversion<110
1017         \def\reserved@a#1{%
1018             \def\@remove@tlig##1{\@remove@tlig@##1\@nil#1\@nil\relax}
1019             \def\@remove@tlig@##1#1{\@remove@tlig@##1}}
1020         \edef\reserved@b{\detokenize{+tlig;}}
1021         \expandafter\reserved@a\expandafter{\reserved@b}
1022         \def\@remove@tlig@#1\@nil#2\relax{#1}
1023
1024         \def\remove@tlig#1{%
1025             \begingroup
1026             \font\remove@tlig
1027             \expandafter\@remove@tlig\expandafter{\fontname\font}%
1028             \remove@tlig
1029             \char#1\relax
1030             \endgroup
1031         }
1032     \else
1033         \newprotectedluacmd\@remove@tlig@@@

```

Now we can define the function. Mostly we just have to insert a protected glyph node, which is a glyph node with subtype 256. But we have to keep track of the current mode to avoid inserting the glyph into a vlist.

```

1033     \now@and@everyjob{\directlua{
1034         local rawchar_func = token.create'\@remove@tlig@@@'.index
1035         local forcehmode = tex.forcehmode
1036         local put_next = token.put_next
1037         local glyph_id = node.id'glyph'
1038         local rawchar_token = token.new(rawchar_func, token.command_id'lua_call')
1039         lua.get_functions_table()[rawchar_func] = function()
1040             local mode = tex.nest.top.mode
1041             if mode == 1 or mode == -1 then
1042                 put_next(rawchar_token)
1043                 return forcehmode(true)
1044             end
1045             local n = node.new(glyph_id, 256)
1046             n.font = font.current()
1047             n.char = token.scan_int()
1048             return node.write(n)
1049         end
1050     }}

```

Now `\remove@tlig` can be implemented almost as in XeTeX.

```

1051     \def\remove@tlig#1{\@remove@tlig@##1\relax}
1052 \fi
1053 \fi
1054 \else

```

XeTeX

```
1055 \def\UnicodeFontTeXLigatures{mapping=tex-text;}
1056 \def\remove@tlig#1{\XeTeXglyph\numexpr\XeTeXcharglyph#1\relax}
1057 \fi
1058 \def\UnicodeFontFile#1#2{"[#1]:#2"}
1059 \def\UnicodeFontName#1#2{"#1:#2"}
    Declare the encoding
1060 \DeclareFontEncoding\UnicodeEncodingName{}{}
    Declare accent command to use a postpended combining character rather than the
TeX \accent primitive
1061 \def\add@unicode@accent#1#2{%
1062 \if\relax\detokenize{#2}\relax^~a0\else#2\fi
1063 \char#1\relax}
```

In its original implementation `\DeclareUnicodeAccent` was given 3 arguments (with second the “Unicode encoding” a.k.a., `\UnicodeEncodingName`) while in other places, e.g., `\DeclareUnicodeComposite`, we always made encoding implicit. So we now change it here to implicit too so that the interfaces become a bit more consistent. To avoid making that a breaking change (even though it only affects two packages on CTAN) we test for `#2` being `\UnicodeEncodingName`. This would not catch if somebody used `\DeclareUnicodeAccent{=} {TU-sub} {"0304}` but that fortunately hasn’t happened. With the implicit argument you would need to change `\UnicodeEncodingName` instead, as you have to do anyway for the other interface commands.

```
1064 \def\DeclareUnicodeAccent#1#2{%
1065 \edef\reserved@a{#2}%
1066 \edef\reserved@b{\UnicodeEncodingName}%
1067 \ifx\reserved@a\reserved@b
1068 \def\reserved@a{\DeclareUnicodeAccent@{#1}}%
1069 \else
1070 \def\reserved@a{\DeclareUnicodeAccent@{#1}\UnicodeEncodingName}%
1071 \fi
1072 \reserved@a{#2}%
1073 }
1074 \def\DeclareUnicodeAccent@#1#2#3{%
1075 \DeclareTextCommand{#1}{#2}{\add@unicode@accent{#3}}%
1076 }
```

Wrapper around `\DeclareTextCompositeCommand` that uses the declared composite if it exists in the current font or falls back to the default definition for the TU accent if not.

```
1077 {
1078 \catcode\z@=11\relax
1079 \gdef\DeclareUnicodeComposite#1#2#3{%
1080 \def\reserved@a##1##2{%
1081 \DeclareTextCompositeCommand#1\UnicodeEncodingName{#2}{%
1082 \iffontchar\font#3 ##2%
1083 \else ##1\fi}}%
1084 \expandafter\expandafter\expandafter\extract@default@composite
1085 \csname\UnicodeEncodingName\string#1\endcsname{#2}\@nil
1086 \bgroup
1087 \lccode\z@#3 %
```

```

1088     \lowercase{\egroup
1089     \expandafter\reserved@a\expandafter{\reserved@b}{^^@}}}%
1090 }

1091 \def\extract@default@composite#1{%
1092   \ifx\@text@composite#1%
1093     \expandafter\extract@default@composite@a
1094   \else
1095     \expandafter\extract@default@composite@b\expandafter#1%
1096   \fi}

1097 \def\extract@default@composite@a#1\@text@composite#2\@nil{%
1098   \def\reserved@b{#2}}
1099 \def\extract@default@composite@b#1#2\@nil{%
1100   \def\reserved@b{#1#2}}

```

Next two commands are simply syntactic sugar to go with the other `\DeclareUnicode...` declarations.

```

1101 \def\DeclareUnicodeSymbol#1{\DeclareTextSymbol{#1}{\UnicodeEncodingName}}
1102 \def\DeclareUnicodeCommand#1{\DeclareTextCommand{#1}{\UnicodeEncodingName}}

1103 \DeclareUnicodeCommand\textquotesingle {\remove@tlig{"0027}}
1104 \DeclareUnicodeCommand\textasciigrave {\remove@tlig{"0060}}
1105 \DeclareUnicodeCommand\textquotedbl {\remove@tlig{"0022}}

1106 \DeclareUnicodeSymbol{\textdollar} {"0024}
1107 \DeclareUnicodeSymbol{\textless} {"003C}
1108 \DeclareUnicodeSymbol{\textgreater} {"003E}
1109 \DeclareUnicodeSymbol{\textbackslash} {"005C}
1110 \DeclareUnicodeSymbol{\textasciicircum} {"005E}
1111 \DeclareUnicodeSymbol{\textunderscore} {"005F}
1112 \DeclareUnicodeSymbol{\textbraceleft} {"007B}
1113 \DeclareUnicodeSymbol{\textbar} {"007C}
1114 \DeclareUnicodeSymbol{\textbraceright} {"007D}
1115 \DeclareUnicodeSymbol{\textasciitilde} {"007E}
1116 \DeclareUnicodeSymbol{\textexclamdown} {"00A1}
1117 \DeclareUnicodeSymbol{\textcent} {"00A2}
1118 \DeclareUnicodeSymbol{\textsterling} {"00A3}
1119 \DeclareUnicodeSymbol{\textcurrency} {"00A4}
1120 \DeclareUnicodeSymbol{\textyen} {"00A5}
1121 \DeclareUnicodeSymbol{\textbrokenbar} {"00A6}
1122 \DeclareUnicodeSymbol{\textsection} {"00A7}
1123 \DeclareUnicodeSymbol{\textasciidieresis} {"00A8}
1124 \DeclareUnicodeSymbol{\textcopyright} {"00A9}
1125 \DeclareUnicodeSymbol{\textordfeminine} {"00AA}

1126 \DeclareUnicodeSymbol{\guillemetleft} {"00AB}
1127 % old Adobe name
1128 \DeclareUnicodeSymbol{\guillemotleft} {"00AB}

1129 \DeclareUnicodeSymbol{\textlnot} {"00AC}
1130 \DeclareUnicodeSymbol{\textregistered} {"00AE}
1131 \DeclareUnicodeSymbol{\textasciimacron} {"00AF}
1132 \DeclareUnicodeSymbol{\textdegree} {"00B0}
1133 \DeclareUnicodeSymbol{\textpm} {"00B1}
1134 \DeclareUnicodeSymbol{\texttwosuperior} {"00B2}
1135 \DeclareUnicodeSymbol{\textthreesuperior} {"00B3}

```



```

1136 \DeclareUnicodeSymbol{\textasciiacute} {"00B4}
1137 \DeclareUnicodeSymbol{\textmu} {"00B5}
1138 \DeclareUnicodeSymbol{\textparagraph} {"00B6}
1139 \DeclareUnicodeSymbol{\textperiodcentered} {"00B7}
1140 \DeclareUnicodeSymbol{\textonesuperior} {"00B9}
1141 \DeclareUnicodeSymbol{\textordmasculine} {"00BA}

1142 \DeclareUnicodeSymbol{\guillemetright} {"00BB}
1143 % old Adobe name
1144 \DeclareUnicodeSymbol{\guillemotright} {"00BB}

1145 \DeclareUnicodeSymbol{\textonequarter} {"00BC}
1146 \DeclareUnicodeSymbol{\textonehalf} {"00BD}
1147 \DeclareUnicodeSymbol{\textthreequarters} {"00BE}
1148 \DeclareUnicodeSymbol{\textquestiondown} {"00BF}
1149 \DeclareUnicodeSymbol{\AE} {"00C6}
1150 \DeclareUnicodeSymbol{\DH} {"00D0}
1151 \DeclareUnicodeSymbol{\texttimes} {"00D7}
1152 \DeclareUnicodeSymbol{\O} {"00D8}
1153 \DeclareUnicodeSymbol{\TH} {"00DE}
1154 \DeclareUnicodeSymbol{\ss} {"00DF}
1155 \DeclareUnicodeSymbol{\ae} {"00E6}
1156 \DeclareUnicodeSymbol{\dh} {"00F0}
1157 \DeclareUnicodeSymbol{\textdiv} {"00F7}
1158 \DeclareUnicodeSymbol{\o} {"00F8}
1159 \DeclareUnicodeSymbol{\th} {"00FE}
1160 \DeclareUnicodeSymbol{\DJ} {"0110}
1161 \DeclareUnicodeSymbol{\dj} {"0111}
1162 \DeclareUnicodeSymbol{\i} {"0131}
1163 \DeclareUnicodeSymbol{\IJ} {"0132}
1164 \DeclareUnicodeSymbol{\ij} {"0133}
1165 \DeclareUnicodeSymbol{\L} {"0141}
1166 \DeclareUnicodeSymbol{\l} {"0142}
1167 \DeclareUnicodeSymbol{\NG} {"014A}
1168 \DeclareUnicodeSymbol{\ng} {"014B}
1169 \DeclareUnicodeSymbol{\OE} {"0152}
1170 \DeclareUnicodeSymbol{\oe} {"0153}
1171 \DeclareUnicodeSymbol{\textflorin} {"0192}
1172 \DeclareUnicodeSymbol{\j} {"0237}
1173 \DeclareUnicodeSymbol{\textasciicaron} {"02C7}
1174 \DeclareUnicodeSymbol{\textasciibreve} {"02D8}
1175 \DeclareUnicodeSymbol{\textacutedbl} {"02DD}
1176 \DeclareUnicodeSymbol{\textgravedbl} {"02F5}
1177 \DeclareUnicodeSymbol{\texttildelow} {"02F7}
1178 \DeclareUnicodeSymbol{\textbaht} {"0E3F}
1179 \DeclareUnicodeSymbol{\SS} {"1E9E}
1180 \DeclareUnicodeSymbol{\textcompwordmark} {"200C}

1181 %\DeclareUnicodeSymbol{\textnonbreakinghyphen} {"2011}
1182 %\DeclareUnicodeSymbol{\textfiguredash} {"2012}
1183 \DeclareUnicodeSymbol{\textendash} {"2013}
1184 \DeclareUnicodeSymbol{\textemdash} {"2014}
1185 %\DeclareUnicodeSymbol{\texthorizontalbar} {"2015}

```

Unfortunately some fonts do not implement "2011, "2012 and/or "2015 (including the

L^AT_EX default fonts for Unicode engines) so we provide some approximations if the glyph is missing, like we do for OT1 and T1.

The `\nobreak\hskip\z@` is there to prevent a break after the hyphen but allow later breaks in the remainder of the word.

```

1186 \DeclareUnicodeCommand{\textnonbreakinghyphen}
1187     {\iffontchar\font "2011 \char "2011 \else \mbox{-}\nobreak\hskip\z@ \fi}
1188 \DeclareUnicodeCommand{\textfiguredash}
1189     {\iffontchar\font "2012 \char "2012 \else \char "2013 \fi}
1190 \DeclareUnicodeCommand{\texthorizontalbar}
1191     {\iffontchar\font "2015 \char "2015 \else \char "2014 \fi}

1192 \DeclareUnicodeSymbol{\textbardbl}           {"2016}
1193 \DeclareUnicodeSymbol{\textquoteleft}       {"2018}
1194 \DeclareUnicodeSymbol{\textquoteright}      {"2019}
1195 \DeclareUnicodeSymbol{\quotesinglbase}      {"201A}
1196 \DeclareUnicodeSymbol{\textquotedblleft}   {"201C}
1197 \DeclareUnicodeSymbol{\textquotedblright}  {"201D}
1198 \DeclareUnicodeSymbol{\quotedblbase}       {"201E}
1199 \DeclareUnicodeSymbol{\textdagger}          {"2020}
1200 \DeclareUnicodeSymbol{\textdaggerdbl}       {"2021}
1201 \DeclareUnicodeSymbol{\textbullet}          {"2022}
1202 \DeclareUnicodeSymbol{\textellipsis}        {"2026}
1203 \DeclareUnicodeSymbol{\textperthousand}    {"2030}
1204 \DeclareUnicodeSymbol{\textpertenthousand} {"2031}
1205 \DeclareUnicodeSymbol{\guilsinglleft}      {"2039}
1206 \DeclareUnicodeSymbol{\guilsinglright}     {"203A}
1207 \DeclareUnicodeSymbol{\textreferencemark}   {"203B}
1208 \DeclareUnicodeSymbol{\textinterrobang}     {"203D}
1209 \DeclareUnicodeSymbol{\textfractionsolidus} {"2044}
1210 \DeclareUnicodeSymbol{\textlquill}          {"2045}
1211 \DeclareUnicodeSymbol{\textrquill}          {"2046}
1212 \DeclareUnicodeSymbol{\textdiscount}        {"2052}
1213 \DeclareUnicodeSymbol{\textcolonmonetary}   {"20A1}
1214 \DeclareUnicodeSymbol{\textlira}            {"20A4}
1215 \DeclareUnicodeSymbol{\textnaira}           {"20A6}
1216 \DeclareUnicodeSymbol{\textwon}            {"20A9}
1217 \DeclareUnicodeSymbol{\textdong}           {"20AB}
1218 \DeclareUnicodeSymbol{\texteuro}           {"20AC}
1219 \DeclareUnicodeSymbol{\textpeso}           {"20B1}
1220 \DeclareUnicodeSymbol{\textcelsius}        {"2103}
1221 \DeclareUnicodeSymbol{\textnumero}         {"2116}
1222 \DeclareUnicodeSymbol{\textcircledP}       {"2117}
1223 \DeclareUnicodeSymbol{\textrecipe}         {"211E}
1224 \DeclareUnicodeSymbol{\textservicemark}    {"2120}
1225 \DeclareUnicodeSymbol{\texttrademark}      {"2122}
1226 \DeclareUnicodeSymbol{\textohm}           {"2126}
1227 \DeclareUnicodeSymbol{\textmho}           {"2127}
1228 \DeclareUnicodeSymbol{\textestimated}     {"212E}
1229 \DeclareUnicodeSymbol{\textleftarrow}     {"2190}
1230 \DeclareUnicodeSymbol{\textuparrow}        {"2191}
1231 \DeclareUnicodeSymbol{\textrightarrow}     {"2192}
1232 \DeclareUnicodeSymbol{\textdownarrow}     {"2193}
1233 \DeclareUnicodeSymbol{\textminus}         {"2212}
1234

```

```

1235 \DeclareUnicodeSymbol{\Hwithstroke}      {"0126}
1236 \DeclareUnicodeSymbol{\hwithstroke}     {"0127}

    Not all fonts have U+2217 but using U+002A requires some adjustment.

1237 \DeclareUnicodeCommand{\textasteriskcentered}{%
1238   \iffontchar\font"2217 \char"2217 \else
1239     \begingroup
1240       \fontsize
1241       {\the\dimexpr1.3\dimexpr\f@size pt\relax}%
1242       {\f@baselineskip}%
1243       \selectfont
1244       \raisebox{-0.7ex}{\dimexpr\height-0.7ex}[0pt]{*}%
1245     \endgroup
1246   \fi
1247 }

1248 \DeclareUnicodeSymbol{\textsurd}         {"221A}
1249 \DeclareUnicodeSymbol{\textlangle}      {"2329}
1250 \DeclareUnicodeSymbol{\textrangle}      {"232A}
1251 \DeclareUnicodeSymbol{\textblank}       {"2422}
1252 \DeclareUnicodeSymbol{\textvisiblespace} {"2423}
1253 \DeclareUnicodeSymbol{\textopenbullet}  {"25E6}
1254 \DeclareUnicodeSymbol{\textbigcircle}   {"25EF}
1255 \DeclareUnicodeSymbol{\textmusicalnote} {"266A}
1256 \DeclareUnicodeSymbol{\textmarried}     {"26AD}
1257 \DeclareUnicodeSymbol{\textdivorced}    {"26AE}
1258 \DeclareUnicodeSymbol{\textinterrobangdown} {"2E18}

```

Accents must be declared before the composites that use them.

```

1259 \DeclareUnicodeAccent{\`}{"0300}
1260 \DeclareUnicodeAccent{\'}{"0301}
1261 \DeclareUnicodeAccent{\~}{"0302}
1262 \DeclareUnicodeAccent{\~}{"0303}
1263 \DeclareUnicodeAccent{\=}{"0304}
1264 \DeclareUnicodeAccent{\u}{"0306}
1265 \DeclareUnicodeAccent{\.}{"0307}
1266 \DeclareUnicodeAccent{\"}{"0308}
1267 \DeclareUnicodeAccent{\r}{"030A}
1268 \DeclareUnicodeAccent{\H}{"030B}
1269 \DeclareUnicodeAccent{\v}{"030C}
1270 \DeclareUnicodeAccent{\b}{"0332}
1271 \DeclareUnicodeAccent{\d}{"0323}
1272 \DeclareUnicodeAccent{\c}{"0327}
1273 \DeclareUnicodeAccent{\k}{"0328}

    The odd one out:

1274 \DeclareUnicodeCommand\textcommabelow[1]
1275   {\hmode@bgroup\oalign{\null#1\cr\hidewidth\raise-.31ex
1276   \hbox{\check@mathfonts\fontsize\ssf@size\z@
1277   \math@fontsfalse\selectfont,}\hidewidth}\egroup}

1278 \DeclareUnicodeComposite{\^}           {"005E}
1279 \DeclareUnicodeComposite{\~}           {"007E}

1280 \DeclareUnicodeComposite{\`}           {A}{"00C0}
1281 \DeclareUnicodeComposite{\'}           {A}{"00C1}
1282 \DeclareUnicodeComposite{\^}           {A}{"00C2}

```

```

1283 \DeclareUnicodeComposite{\~}      {A}{"00C3}
1284 \DeclareUnicodeComposite{""}     {A}{"00C4}
1285 \DeclareUnicodeComposite{\r}     {A}{"00C5}
1286 \DeclareUnicodeComposite{\c}     {C}{"00C7}
1287 \DeclareUnicodeComposite{\'}     {E}{"00C8}
1288 \DeclareUnicodeComposite{\'}     {E}{"00C9}
1289 \DeclareUnicodeComposite{\^}     {E}{"00CA}
1290 \DeclareUnicodeComposite{""}     {E}{"00CB}
1291 \DeclareUnicodeComposite{\'}     {I}{"00CC}
1292 \DeclareUnicodeComposite{\'}     {I}{"00CD}
1293 \DeclareUnicodeComposite{\^}     {I}{"00CE}
1294 \DeclareUnicodeComposite{""}     {I}{"00CF}
1295 \DeclareUnicodeComposite{\~}     {N}{"00D1}
1296 \DeclareUnicodeComposite{\'}     {O}{"00D2}
1297 \DeclareUnicodeComposite{\'}     {O}{"00D3}
1298 \DeclareUnicodeComposite{\^}     {O}{"00D4}
1299 \DeclareUnicodeComposite{\~}     {O}{"00D5}
1300 \DeclareUnicodeComposite{""}     {O}{"00D6}
1301 \DeclareUnicodeComposite{\'}     {U}{"00D9}
1302 \DeclareUnicodeComposite{\'}     {U}{"00DA}
1303 \DeclareUnicodeComposite{\^}     {U}{"00DB}
1304 \DeclareUnicodeComposite{""}     {U}{"00DC}
1305 \DeclareUnicodeComposite{\'}     {Y}{"00DD}
1306 \DeclareUnicodeComposite{\'}     {a}{"00E0}
1307 \DeclareUnicodeComposite{\'}     {a}{"00E1}
1308 \DeclareUnicodeComposite{\^}     {a}{"00E2}
1309 \DeclareUnicodeComposite{\~}     {a}{"00E3}
1310 \DeclareUnicodeComposite{""}     {a}{"00E4}
1311 \DeclareUnicodeComposite{\r}     {a}{"00E5}
1312 \DeclareUnicodeComposite{\c}     {c}{"00E7}
1313 \DeclareUnicodeComposite{\'}     {e}{"00E8}
1314 \DeclareUnicodeComposite{\'}     {e}{"00E9}
1315 \DeclareUnicodeComposite{\^}     {e}{"00EA}
1316 \DeclareUnicodeComposite{""}     {e}{"00EB}
1317 \DeclareUnicodeComposite{\'}     \i {"00EC}
1318 \DeclareUnicodeComposite{\'}     {i}{"00EC}
1319 \DeclareUnicodeComposite{\'}     \i {"00ED}
1320 \DeclareUnicodeComposite{\'}     {i}{"00ED}
1321 \DeclareUnicodeComposite{\^}     \i {"00EE}
1322 \DeclareUnicodeComposite{\^}     {i}{"00EE}
1323 \DeclareUnicodeComposite{""}     \i {"00EF}
1324 \DeclareUnicodeComposite{""}     {i}{"00EF}
1325 \DeclareUnicodeComposite{\~}     {n}{"00F1}
1326 \DeclareUnicodeComposite{\'}     {o}{"00F2}
1327 \DeclareUnicodeComposite{\'}     {o}{"00F3}
1328 \DeclareUnicodeComposite{\^}     {o}{"00F4}
1329 \DeclareUnicodeComposite{\~}     {o}{"00F5}
1330 \DeclareUnicodeComposite{""}     {o}{"00F6}
1331 \DeclareUnicodeComposite{\'}     {u}{"00F9}
1332 \DeclareUnicodeComposite{\'}     {u}{"00FA}
1333 \DeclareUnicodeComposite{\^}     {u}{"00FB}
1334 \DeclareUnicodeComposite{""}     {u}{"00FC}
1335 \DeclareUnicodeComposite{\'}     {y}{"00FD}
1336 \DeclareUnicodeComposite{""}     {y}{"00FF}

```

```

1337 \DeclareUnicodeComposite{\=} {A}{"0100}
1338 \DeclareUnicodeComposite{\=} {a}{"0101}
1339 \DeclareUnicodeComposite{\u} {A}{"0102}
1340 \DeclareUnicodeComposite{\u} {a}{"0103}
1341 \DeclareUnicodeComposite{\k} {A}{"0104}
1342 \DeclareUnicodeComposite{\k} {a}{"0105}
1343 \DeclareUnicodeComposite{\'} {C}{"0106}
1344 \DeclareUnicodeComposite{\'} {c}{"0107}
1345 \DeclareUnicodeComposite{\^} {C}{"0108}
1346 \DeclareUnicodeComposite{\^} {c}{"0109}
1347 \DeclareUnicodeComposite{\.} {C}{"010A}
1348 \DeclareUnicodeComposite{\.} {c}{"010B}
1349 \DeclareUnicodeComposite{\v} {C}{"010C}
1350 \DeclareUnicodeComposite{\v} {c}{"010D}
1351 \DeclareUnicodeComposite{\v} {D}{"010E}
1352 \DeclareUnicodeComposite{\v} {d}{"010F}
1353 \DeclareUnicodeComposite{\=} {E}{"0112}
1354 \DeclareUnicodeComposite{\=} {e}{"0113}
1355 \DeclareUnicodeComposite{\u} {E}{"0114}
1356 \DeclareUnicodeComposite{\u} {e}{"0115}
1357 \DeclareUnicodeComposite{\.} {E}{"0116}
1358 \DeclareUnicodeComposite{\.} {e}{"0117}
1359 \DeclareUnicodeComposite{\k} {E}{"0118}
1360 \DeclareUnicodeComposite{\k} {e}{"0119}
1361 \DeclareUnicodeComposite{\v} {E}{"011A}
1362 \DeclareUnicodeComposite{\v} {e}{"011B}
1363 \DeclareUnicodeComposite{\^} {G}{"011C}
1364 \DeclareUnicodeComposite{\^} {g}{"011D}
1365 \DeclareUnicodeComposite{\u} {G}{"011E}
1366 \DeclareUnicodeComposite{\u} {g}{"011F}
1367 \DeclareUnicodeComposite{\.} {G}{"0120}
1368 \DeclareUnicodeComposite{\.} {g}{"0121}
1369 \DeclareUnicodeComposite{\c} {G}{"0122}
1370 \DeclareUnicodeComposite{\c} {g}{"0123}
1371 \DeclareUnicodeComposite{\^} {H}{"0124}
1372 \DeclareUnicodeComposite{\^} {h}{"0125}
1373 \DeclareUnicodeComposite{\~} {I}{"0128}
1374 \DeclareUnicodeComposite{\~} \i {"0129}
1375 \DeclareUnicodeComposite{\~} {i}{"0129}
1376 \DeclareUnicodeComposite{\=} {I}{"012A}
1377 \DeclareUnicodeComposite{\=} \i {"012B}
1378 \DeclareUnicodeComposite{\=} {i}{"012B}
1379 \DeclareUnicodeComposite{\u} {I}{"012C}
1380 \DeclareUnicodeComposite{\u} \i {"012D}
1381 \DeclareUnicodeComposite{\u} {i}{"012D}
1382 \DeclareUnicodeComposite{\k} {I}{"012E}
1383 \DeclareUnicodeComposite{\k} \i {"012F}
1384 \DeclareUnicodeComposite{\k} {i}{"012F}
1385 \DeclareUnicodeComposite{\.} {I}{"0130}
1386 \DeclareUnicodeComposite{\^} {J}{"0134}
1387 \DeclareUnicodeComposite{\^} \j {"0135}
1388 \DeclareUnicodeComposite{\^} {j}{"0135}
1389 \DeclareUnicodeComposite{\c} {K}{"0136}
1390 \DeclareUnicodeComposite{\c} {k}{"0137}

```

```

1391 \DeclareUnicodeComposite{\'} {L}{"0139}
1392 \DeclareUnicodeComposite{\'} {l}{"013A}
1393 \DeclareUnicodeComposite{\c} {L}{"013B}
1394 \DeclareUnicodeComposite{\c} {l}{"013C}
1395 \DeclareUnicodeComposite{\v} {L}{"013D}
1396 \DeclareUnicodeComposite{\v} {l}{"013E}
1397 \DeclareUnicodeComposite{\'} {N}{"0143}
1398 \DeclareUnicodeComposite{\'} {n}{"0144}
1399 \DeclareUnicodeComposite{\c} {N}{"0145}
1400 \DeclareUnicodeComposite{\c} {n}{"0146}
1401 \DeclareUnicodeComposite{\v} {N}{"0147}
1402 \DeclareUnicodeComposite{\v} {n}{"0148}
1403 \DeclareUnicodeComposite{=} {O}{"014C}
1404 \DeclareUnicodeComposite{=} {o}{"014D}
1405 \DeclareUnicodeComposite{\u} {O}{"014E}
1406 \DeclareUnicodeComposite{\u} {o}{"014F}
1407 \DeclareUnicodeComposite{\H} {O}{"0150}
1408 \DeclareUnicodeComposite{\H} {o}{"0151}
1409 \DeclareUnicodeComposite{\'} {R}{"0154}
1410 \DeclareUnicodeComposite{\'} {r}{"0155}
1411 \DeclareUnicodeComposite{\c} {R}{"0156}
1412 \DeclareUnicodeComposite{\c} {r}{"0157}
1413 \DeclareUnicodeComposite{\v} {R}{"0158}
1414 \DeclareUnicodeComposite{\v} {r}{"0159}
1415 \DeclareUnicodeComposite{\'} {S}{"015A}
1416 \DeclareUnicodeComposite{\'} {s}{"015B}
1417 \DeclareUnicodeComposite{\^} {S}{"015C}
1418 \DeclareUnicodeComposite{\^} {s}{"015D}
1419 \DeclareUnicodeComposite{\c} {S}{"015E}
1420 \DeclareUnicodeComposite{\c} {s}{"015F}
1421 \DeclareUnicodeComposite{\v} {S}{"0160}
1422 \DeclareUnicodeComposite{\v} {s}{"0161}
1423 \DeclareUnicodeComposite{\c} {T}{"0162}
1424 \DeclareUnicodeComposite{\c} {t}{"0163}
1425 \DeclareUnicodeComposite{\v} {T}{"0164}
1426 \DeclareUnicodeComposite{\v} {t}{"0165}
1427 \DeclareUnicodeComposite{\~} {U}{"0168}
1428 \DeclareUnicodeComposite{\~} {u}{"0169}
1429 \DeclareUnicodeComposite{\=} {U}{"016A}
1430 \DeclareUnicodeComposite{\=} {u}{"016B}
1431 \DeclareUnicodeComposite{\u} {U}{"016C}
1432 \DeclareUnicodeComposite{\u} {u}{"016D}
1433 \DeclareUnicodeComposite{\r} {U}{"016E}
1434 \DeclareUnicodeComposite{\r} {u}{"016F}
1435 \DeclareUnicodeComposite{\H} {U}{"0170}
1436 \DeclareUnicodeComposite{\H} {u}{"0171}
1437 \DeclareUnicodeComposite{\k} {U}{"0172}
1438 \DeclareUnicodeComposite{\k} {u}{"0173}
1439 \DeclareUnicodeComposite{\^} {W}{"0174}
1440 \DeclareUnicodeComposite{\^} {w}{"0175}
1441 \DeclareUnicodeComposite{\^} {Y}{"0176}
1442 \DeclareUnicodeComposite{\^} {y}{"0177}
1443 \DeclareUnicodeComposite{"} {Y}{"0178}
1444 \DeclareUnicodeComposite{\'} {Z}{"0179}

```

```

1445 \DeclareUnicodeComposite{\'} {z}{"017A}
1446 \DeclareUnicodeComposite{\.} {Z}{"017B}
1447 \DeclareUnicodeComposite{\.} {z}{"017C}
1448 \DeclareUnicodeComposite{\v} {Z}{"017D}
1449 \DeclareUnicodeComposite{\v} {z}{"017E}
1450 \DeclareUnicodeComposite{\v} {A}{"01CD}
1451 \DeclareUnicodeComposite{\v} {a}{"01CE}
1452 \DeclareUnicodeComposite{\v} {I}{"01CF}
1453 \DeclareUnicodeComposite{\v} \i {"01D0}
1454 \DeclareUnicodeComposite{\v} {i}{"01D0}
1455 \DeclareUnicodeComposite{\v} {O}{"01D1}
1456 \DeclareUnicodeComposite{\v} {o}{"01D2}
1457 \DeclareUnicodeComposite{\v} {U}{"01D3}
1458 \DeclareUnicodeComposite{\v} {u}{"01D4}

1459 \DeclareUnicodeComposite{\'} \AE{"01FC}
1460 \DeclareUnicodeComposite{\'} {E}{"01FC}
1461 \DeclareUnicodeComposite{\'} \ae{"01FD}
1462 \DeclareUnicodeComposite{\'} {æ}{"01FD}
1463 \DeclareUnicodeComposite{\=} \AE{"01E2}
1464 \DeclareUnicodeComposite{\=} {E}{"01E2}
1465 \DeclareUnicodeComposite{\=} \ae{"01E3}
1466 \DeclareUnicodeComposite{\=} {æ}{"01E3}
1467 \DeclareUnicodeComposite{\v} {G}{"01E6}
1468 \DeclareUnicodeComposite{\v} {g}{"01E7}
1469 \DeclareUnicodeComposite{\v} {K}{"01E8}
1470 \DeclareUnicodeComposite{\v} {k}{"01E9}
1471 \DeclareUnicodeComposite{\k} {O}{"01EA}
1472 \DeclareUnicodeComposite{\k} {o}{"01EB}
1473 \DeclareUnicodeComposite{\v} \j {"01F0}
1474 \DeclareUnicodeComposite{\v} {j}{"01F0}
1475 \DeclareUnicodeComposite{\'} {G}{"01F4}
1476 \DeclareUnicodeComposite{\'} {g}{"01F5}
1477 \DeclareUnicodeComposite{\textcommabelow}{S}{"0218}
1478 \DeclareUnicodeComposite{\textcommabelow}{s}{"0219}
1479 \DeclareUnicodeComposite{\textcommabelow}{T}{"021A}
1480 \DeclareUnicodeComposite{\textcommabelow}{t}{"021B}
1481 \DeclareUnicodeComposite{\=} {Y}{"0232}
1482 \DeclareUnicodeComposite{\=} {y}{"0233}
1483 \DeclareUnicodeComposite{\.} {B}{"1E02}
1484 \DeclareUnicodeComposite{\.} {b}{"1E03}
1485 \DeclareUnicodeComposite{\d} {B}{"1E04}
1486 \DeclareUnicodeComposite{\d} {b}{"1E05}
1487 \DeclareUnicodeComposite{\d} {D}{"1E0C}
1488 \DeclareUnicodeComposite{\d} {d}{"1E0D}
1489 \DeclareUnicodeComposite{\=} {G}{"1E20}
1490 \DeclareUnicodeComposite{\=} {g}{"1E21}
1491 \DeclareUnicodeComposite{\d} {H}{"1E24}
1492 \DeclareUnicodeComposite{\d} {h}{"1E25}
1493 \DeclareUnicodeComposite{\d} {K}{"1E32}
1494 \DeclareUnicodeComposite{\d} {k}{"1E33}
1495 \DeclareUnicodeComposite{\d} {L}{"1E36}
1496 \DeclareUnicodeComposite{\d} {l}{"1E37}
1497 \DeclareUnicodeComposite{\d} {M}{"1E42}

```

```

1498 \DeclareUnicodeComposite{\d}      {m}{"1E43}
1499 \DeclareUnicodeComposite{\d}      {N}{"1E46}
1500 \DeclareUnicodeComposite{\d}      {n}{"1E47}
1501 \DeclareUnicodeComposite{\d}      {R}{"1E5A}
1502 \DeclareUnicodeComposite{\d}      {r}{"1E5B}
1503 \DeclareUnicodeComposite{\d}      {S}{"1E62}
1504 \DeclareUnicodeComposite{\d}      {s}{"1E63}
1505 \DeclareUnicodeComposite{\d}      {T}{"1E6C}
1506 \DeclareUnicodeComposite{\d}      {t}{"1E6D}
1507 \DeclareUnicodeComposite{\d}      {V}{"1E7E}
1508 \DeclareUnicodeComposite{\d}      {v}{"1E7F}
1509 \DeclareUnicodeComposite{\d}      {W}{"1E88}
1510 \DeclareUnicodeComposite{\d}      {w}{"1E89}
1511 \DeclareUnicodeComposite{\d}      {Z}{"1E92}
1512 \DeclareUnicodeComposite{\d}      {z}{"1E93}
1513 \DeclareUnicodeComposite{\d}      {A}{"1EA0}
1514 \DeclareUnicodeComposite{\d}      {a}{"1EA1}
1515 \DeclareUnicodeComposite{\d}      {E}{"1EB8}
1516 \DeclareUnicodeComposite{\d}      {e}{"1EB9}
1517 \DeclareUnicodeComposite{\d}      {I}{"1ECA}
1518 \DeclareUnicodeComposite{\d}      {i}{"1ECB}
1519 \DeclareUnicodeComposite{\d}      {O}{"1ECC}
1520 \DeclareUnicodeComposite{\d}      {o}{"1ECD}
1521 \DeclareUnicodeComposite{\d}      {U}{"1EE4}
1522 \DeclareUnicodeComposite{\d}      {u}{"1EE5}
1523 \DeclareUnicodeComposite{\d}      {Y}{"1EF4}
1524 \DeclareUnicodeComposite{\d}      {y}{"1EF5}
1525 </TU>

```

2 Package files

This file now also contains some packages that provide access to the more specialised encodings.

2.1 The fontenc package

This package allows authors to specify which encodings they will use. For each encoding FOO, the package looks to see if the encoding FOO has already been declared. If it has not, the file `fooenc.def` is loaded. The default encoding is set to be FOO.

In addition the package at the moment contains extra code to extend the `\@uclclist` (list of upper/lower case pairs) for encodings that involve cyrillic characters. THIS IS A TEMPORARY SOLUTION and will not stay this way forever (or so we hope) but right now we are missing a proper interface for this and didn't wanted to rush it.

```

1526 <*package>

```

Here we define a macro that extends the `\@uclclist` if needed and afterwards turns itself in a noop.

```

1527 \def\update@uclc@with@cyrillic{%
1528 \expandafter\def\expandafter\@uclclist\expandafter
1529   {\@uclclist
1530   \cyr\CYRA\cyrahch\CYRABHCH\cyrahchdsc\CYRABHCHDSC\cyrahdze
1531   \CYRABHDZE\cyrahha\CYRABHHA\cyr\CYRAE\cyrb\CYRB\cyrbyus

```



```

1532 \CYRBYUS\cyrc\CYRC\cyrch\CYRCH\cyrchldsc\CYRCHLDSC\cyrchrdsc
1533 \CYRCHRSDSC\cyrchvcrs\CYRCHVCRS\cyrd\CYRD\cyrdelta\CYRDELTA
1534 \cyrdje\CYRDJE\cyrdze\CYRDZE\cyrdzhe\CYRDZHE\cyre\CYRE\cyreps
1535 \CYREPS\cyrerev\CYREREV\cyrery\CYRERY\cyrf\CYRF\cyrfita
1536 \CYRFITA\cyrg\CYRG\cyrgdsc\CYRGDSC\cyrgdschcrs\CYRGDSCHCRS
1537 \cyrgchcrs\CYRGHCRS\cyrgkh\CYRGHK\cyrgup\CYRGUP\cyrh\CYRH
1538 \cyrhds\CYRHDSC\cyrhcrs\CYRHHCRS\cyrhkh\CYRHHK\cyhrdsn
1539 \CYRHRDSN\cyri\CYRI\cyrie\CYRIE\cyrii\CYRII\cyrishrt\CYRISHRT
1540 \cyrishrtdsc\CYRISHRTDSC\cyrizh\CYRIZH\cyrje\CYRJE\cyrk\CYRK
1541 \cyrkbeak\CYRKBEAK\cyrkds\CYRKDSC\cyrkchcrs\CYRKCHCRS\cyrkhh
1542 \CYRKHK\cyrkvcrs\CYRKVCRS\cyr1\CYRL\cyrlldsc\CYRLDSC\cyrllhk
1543 \CYRLHK\cyr1je\CYRLJE\cyrml\CYRM\cyrmdsc\CYRMDSC\cyrmhk\CYRMHK
1544 \cyrn\CYRN\cyrndsc\CYRNDSC\cyrng\CYRNG\cyrnhk\CYRNHK\cyrnje
1545 \CYRNJE\cyrnlhk\CYRNLHK\cyro\CYRO\cyrotld\CYROTLD\cyrp\CYRP
1546 \cyrphk\CYRPHK\cyrq\CYRQ\cyrr\CYRR\cyrrdsc\CYRRDSC\cyrrhk
1547 \CYRRHK\cyrrtick\CYRRTICK\cyrs\CYRS\cyrsacrs\CYRSACRS
1548 \cyrschwa\CYRSCHWA\cyrsdsc\CYRSDSC\cyrsemisftsn\CYRSEMISFTSN
1549 \cyrsftsn\CYRSFTSN\cyrsh\CYRSH\cyrshch\CYRSHCH\cyrshha\CYRSHHA
1550 \cyrt\CYRT\cyrtldsc\CYRTDSC\cyrtetse\CYRTETSE\cyrtshe\CYRTSHE
1551 \cyru\CYRU\cyrushrt\CYRUSHRT\cyrv\CYRV\cyrw\CYRW\cyry\CYRY
1552 \cyrya\CYRYA\cyryat\CYRYAT\cyryhcrs\CYRYHCRS\cyryi\CYRYI\cyryo
1553 \CYRYO\cyryu\CYRYU\cyrz\CYRZ\cyrzds\CYRZDSC\cyrzsh\CYRZH
1554 \cyrzhdsc\CYRZHDSC}%
1555 \let\update@uclc@with@cyrillic\relax
1556 }

```

Here we process each option:

```

1557 \DeclareOption*{%
1558   \let\encodingdefault\CurrentOption

```

From 2020/02/02 release onward we only load the encoding files if they haven't be loaded already. To check this we look at whether `\T@encoding` is already defined. If not, we load it later (indicated by setting the switch `@tempswa` to true) and we always load if we are using an older format (or rather in a rollback situation).

```

1559   \@tempswafalse
1560   \@ifl@t@r\fmtversion{2020/02/02}%
1561     {\expandafter\ifx\csname T@\CurrentOption\endcsname\relax
1562       \@tempswattrue\fi}%
1563     {\@tempswattrue}%

```

Load if necessary:

```

1564   \if@tempswa
1565     \edef\reserved@f{%
1566       \lowercase{\def\noexpand\reserved@f{\CurrentOption enc.def}}}%
1567     \reserved@f
1568     \InputIfFileExists\reserved@f
1569       {\PackageError{fontenc}%
1570         {Encoding file '\reserved@f' not found.%
1571         \MessageBreak
1572         You might have misspelled the name of the encoding
1573         \MessageBreak
1574         or a required support package (e.g., cyrillic) is
1575         \MessageBreak
1576         missing in your installation}%

```

```

1577         {Necessary code for this encoding was not
1578         loaded.\MessageBreak
1579         Thus calling the encoding later on will
1580         produce further error messages.}}%
1581     \let\reserved@f\relax

```

In case the current encoding is one of a list of known cyrillic ones we extend the `\@uclclist`:

```

1582     \expandafter\in@\expandafter{\CurrentOption}%
1583                                     {T2A,T2B,T2C,X2,LCY,OT2}%
1584     \ifin@

```

But only if it hasn't already been extended. This might happen if there are several calls to fontenc loading one of the above encodings. If we don't do this check the `\@uclclist` gets unnecessarily big, slowing down the processing at runtime.

```

1585     \expandafter\in@\expandafter\cyr@\expandafter
1586                                     {\@uclclist}%
1587     \ifin@
1588     \else
1589     \update@uclc@with@cyrillic
1590     \fi
1591 \fi
1592 \fi
1593 }

```

```

1594 \ProcessOptions*

```

We select the new font encoding default (i.e., the last encoding specified in the option list). But this encoding may not work with the current `\f@shape`: e.g., LY1 is not defined for `cmr` and therefore packages switching to LY1 usually also change `\rmdefault`. But that only applies at `\begin{document}` so we get a spurious warning if we use what L^AT_EX previously used:

```

1595 %\fontencoding\encodingdefault\selectfont

```

So instead we do this here:

```

1596 \usefont\encodingdefault\familydefault\seriesdefault\shapedefault

```

To save some space we get rid of the macro extending the `\@uclclist` (might have happened already).

```

1597 \let\update@uclc@with@cyrillic\relax

```

Finally we pretend that the fontenc package wasn't read in. This allows for using it several times, e.g., in a class file and in the preamble (at the cost of not getting any version info). That kind of hackery shows that using a general purpose package just for loading an encoding is not the right kind of interface for setting up encodings — it will get replaced at some point in the future.

```

1598 \let\@elt\relax
1599 \xdef\@fontenc@load@list{\@fontenc@load@list
1600     \@elt{\csname opt@fontenc.sty\endcsname}}
1601 \global\expandafter\let\csname ver@fontenc.sty\expandafter\endcsname
1602     \csname ver@fontenc.sty\endcsname
1603 \global\expandafter\let\csname ver@fontenc.sty\endcsname\relax
1604 \global\expandafter\let\csname opt@fontenc.sty\endcsname\relax
1605 \global\let\@ifl@ter@@\@ifl@ter
1606 \def\@ifl@ter#1#2#3#4#5{\global\let\@ifl@ter\@ifl@ter@@}
1607 </package>

```

File 22

ltxcounts.dtx

1 Counters and Lengths

Commands for defining and using counters. This file defines the following commands. In each case $\langle counter \rangle$ may be $*$ denoting the current counter as set by a previous $\backslash refstepcounter$.

$\backslash newcounter$	To define a new counter.
$\backslash setcounter$	To set the value of counters.
$\backslash addtocounter$	Increase the $\langle counter \rangle$ #1 by the number #2.
$\backslash stepcounter$	Increase the $\langle counter \rangle$ by one.
$\backslash refstepcounter$	Increase the $\langle counter \rangle$ by one, also setting the value used by $\backslash label$.
$\backslash value$	For accessing the value of the counter as a TeX number (as opposed to $\backslash the \langle counter \rangle$ which expands to the <i>printed</i> representation of $\langle counter \rangle$)
$\backslash arabic$	$\backslash arabic \langle counter \rangle$: 1, 2, 3, ...
$\backslash roman$	$\backslash roman \langle counter \rangle$: i, ii, iii, ...
$\backslash Roman$	$\backslash Roman \langle counter \rangle$: I, II, III, ...
$\backslash alph$	$\backslash alph \langle counter \rangle$: a, b, c, ...
$\backslash Alph$	$\backslash Alph \langle counter \rangle$: A, B, C, ...
$\backslash fnsymbol$	$\backslash fnsymbol \langle counter \rangle$: *, †, ‡, ...
$\backslash counterwithin$	$\backslash counterwithin [\langle format \rangle] \langle counter \rangle \langle within-counter \rangle$: Resets $\langle counter \rangle$ whenever $\langle within-counter \rangle$ is stepped. Also redefines $\backslash the \langle counter \rangle$ command to produce $\backslash the \langle within-counter \rangle . \langle format \rangle \langle counter \rangle$ with $\backslash arabic$ as the default for $\langle format \rangle$. Star form omits redefining the print representation. The $*$ alias for the current counter may not be used in either argument.
$\backslash counterwithout$	$\backslash counterwithout [\langle format \rangle] \langle counter \rangle \langle within-counter \rangle$: Removes $\langle counter \rangle$ from the reset list of $\langle within-counter \rangle$. Also redefines $\backslash the \langle counter \rangle$ command to produce $\langle format \rangle \langle counter \rangle$ with $\backslash arabic$ as the default for $\langle format \rangle$. Star form omits redefining the print representation. The $*$ alias for the current counter may not be used in either argument.

¹ $\langle *2kernel \rangle$

1.1 Environment Counter Macros

An environment foo has an associated counter defined by the following control sequences:

`\c@foo` Contains the counter's numerical value. It is defined by `\newcount\foocounter`.

`\thefoo` Macro that expands to the printed value of `\foocounter`. For example, if sections are numbered within chapters, and section headings look like
Section II-3. The Nature of Counters
then `\thesection` might be defined by:
`\def\thesection`
 `{\@Roman{\c@chapter}-\@arabic{\c@section}}`

`\p@foo` Macro that expands to a printed 'reference prefix' of counter `foo`. Any `\ref` to a value created by counter `foo` will produce the expansion of `\p@foo\thefoo` when the `\label` command is executed. See file `ltxref.dtx` for an extension of this mechanism.

`\cl@foo` List of counters to be reset when `foo` stepped. Has format `\@elt{countera}\@elt{counterb}\@elt{counterc}`.

NOTE:

`\thefoo` and `\p@foo` *must* be defined in such a way that `\edef\bar{\thefoo}` or `\edef\bar{\p@foo}` defines `\bar` so that it will evaluate to the counter value at the time of the `\edef`, even after `\foocounter` and any other counters have been changed. This will happen if you use the standard commands `\@arabic`, `\@Roman`, etc.

The following commands are used to define and modify counters.

`\refstepcounter{<foo>}`

Same as `\stepcounter`, but it also defines `\@currentlabel`, `\@currentHref` and `\@currentcounter` and so that a subsequent `\label{<bar>}` command causes `\ref{<bar>}` to generate the current value of counter `<foo>`.

`\@definecounter{<foo>}`

Initializes counter `{<foo>}` (with empty reset list), defines `\p@foo` and `\thefoo` to be null and `\theHfoo` to be `\number\value{foo}`. Also adds `<foo>` to `\cl@ckpt` – the reset list of a dummy counter `@ckpt` used for taking checkpoints for the `\include` system.

`\@addtoreset{<foo>}{<bar>}` : Adds counter `<foo>` to the list of counters `\cl@bar` to be reset when counter `<bar>` is stepped.

`\@removefromreset{<foo>}{<bar>}` : Removes counter `<foo>` to the list of counters `\cl@bar` to be reset when counter `<bar>` is stepped.

`\setcounter` `\setcounter{<foo>}{<val>}` : Globally sets `\foocounter` equal to `<val>`.

```

2 \def\setcounter#1#2{%
3   \@ifundefined{c@#1}%
4     {\@nocounterr{#1}}%
5     {\global\csname c@#1\endcsname#2\relax}}
```

(End of definition for `\setcounter`.)

`\addtocounter` `\addtocounter{<foo>}{<val>}` Globally increments `\foocounter` by `<val>`.

```

6 \def\addtocounter#1#2{%
7   \@ifundefined{c@#1}%
8     {\@nocounterr{#1}}%
9     {\global\advance\csname c@#1\endcsname #2\relax}}
```

(End of definition for `\addtocounter`.)

`\newcounter` `\newcounter{<newctr>}[<oldctr>]` Defines `<newctr>` to be a counter, which is reset when counter `<oldctr>` is stepped. If `<newctr>` already defined produces ‘`c@newctr` already defined’ error.

```

10 \def\newcounter#1{%
11   \expandafter\ifdefinable \csname c@#1\endcsname
12     {\@definecounter{#1}}%
13   \ifnextchar[{\@newctr{#1}}{-}]

```

(End of definition for \newcounter.)

`\value` `\value{<ctr>}` produces the value of counter `<ctr>`, for use with a `\setcounter` or `\addtocounter` command.

```

14 \def\value#1{\csname c@#1\endcsname}

```

(End of definition for \value.)

`\c@*` Make the current counter available as a L^AT_EX counter with name `*`, so `\alph*` returns the current counter as a letter, `\stepcounter{*}` increments the current counter, etc.

```

15 \protected\expandafter
16   \def\csname c@*\endcsname{\value\@currentcounter}

```

(End of definition for \c@.)*

`\@newctr`

```

17 \def\@newctr#1[#2]{%
18   \@ifundefined{c@#2}{\@nocounterr{#2}}{\@addtoreset{#1}{#2}}

```

(End of definition for \@newctr.)

`\stepcounter` `\stepcounterfoo` Globally increments counter `\c@FOO` and resets all subsidiary counters.

```

19 \def\stepcounter#1{%
20   \addtocounter{#1}\@ne
21   \begingroup
22     \let\@elt\@stpelt
23     \csname cl@#1\endcsname
24   \endgroup}

```

(End of definition for \stepcounter.)

`\@stpelt` Rather than resetting the “within” counter to zero we set it to `-1` and then run `\stepcounter` that moves it to `0` and also initiates resetting the next level down.

```

25 \</2kernel>
26 \<latexrelease>\IncludeInRelease{2015/01/01}{\@stpelt}
27 \<latexrelease>                                     {Reset nested counters}%
28 \<*2kernel | latexrelease>
29 \def\@stpelt#1{\global\csname c@#1\endcsname \m@ne\stepcounter{#1}}%
30 \<latexrelease>\EndIncludeInRelease
31 \</2kernel | latexrelease>
32 \<latexrelease>\IncludeInRelease{0000/00/00}{\@stpelt}
33 \<latexrelease>                                     {Reset nested counters}%%
34 \<latexrelease>\def\@stpelt#1{\global\csname c@#1\endcsname \z@}%
35 \<latexrelease>\EndIncludeInRelease
36 \<*2kernel>

```

(End of definition for \@stpelt.)

`\cl@ckpt`

```
37 \def\cl@ckpt{\@elt{page}}
```

(End of definition for `\cl@ckpt`.)

`\@definecounter`

```
38 % \changes{v1.1p}{2024/10/26}{Fully expand counter name in theHfoo commands (gh/1508)}
39 </2ekernel>
40 <latexrelease>\IncludeInRelease{2024/11/01}{\@definecounter}
41 <latexrelease>                                {provide theHfoo commands}%
42 <*2ekernel | latexrelease>
43 \def\@definecounter#1{\expandafter\newcount\csname c@#1\endcsname
44   \setcounter{#1}z@
45   \global\expandafter\let\csname cl@#1\endcsname\@empty
46   \@addtoreset{#1}{@ckpt}%
47   \global\expandafter\let\csname p@#1\endcsname\@empty
48   \expandafter\xdef\csname the#1\endcsname{%
49     \noexpand\the\noexpand\value{#1}}%
```

If `\the#1` is undefined or `\relax` we define it with the standard definition for counters, otherwise we warn. This will catch, for example, that somebody defines a counter named “index” conflicting with the `theindex` environment.

```
50   \expandafter
51   \ifx\csname the#1\endcsname\relax
52     \expandafter
53     \gdef\csname the#1\expandafter\endcsname\expandafter
54       {\expandafter\@arabic\csname c@#1\endcsname}%
55   \else
56     \@latex@warning{Command ‘\string\the#1’ already
57       defined -- not changed}%
58   \fi}
59 <latexrelease>\EndIncludeInRelease
60 </2ekernel | latexrelease>
61 <latexrelease>\IncludeInRelease{0000/00/00}{\@definecounter}
62 <latexrelease>                                {provide theHfoo commands}%%
63 <latexrelease>\def\@definecounter#1{\expandafter\newcount\csname c@#1\endcsname
64   \setcounter{#1}z@
65   \global\expandafter\let\csname cl@#1\endcsname\@empty
66   \global\expandafter\let\csname p@#1\endcsname\@empty
67   \global\expandafter\let\csname p@#1\endcsname\@empty
68   \expandafter
69   \ifx\csname the#1\endcsname\relax
70     \expandafter
71     \gdef\csname the#1\expandafter\endcsname\expandafter
72       {\expandafter\@arabic\csname c@#1\endcsname}%
73   \else
74     \@latex@warning{Command ‘\string\the#1’ already
75       defined -- not changed}%
76   \fi}
77 <latexrelease>\EndIncludeInRelease
78 <*2ekernel>
```

(End of definition for `\@definecounter`.)

`\@addtoreset` If a counter is reset when a parent counter changes it no longer has an unique value across the document. As `\theH<counter>` should be unique this representation is changed to include also the representation of the parent. This is not 100% guaranteed to work but has been used this way by hyperref for many years without causing problems.

```

79 </2ekernel>
80 <latexrelease>\IncludeInRelease{2024/11/01}{\@addtoreset}
81 <latexrelease>                                     {provide theHfoo commands}%
82 <*2ekernel | latexrelease>
83 \def\@addtoreset#1#2{\expandafter\@cons\csname c1@#2\endcsname {{#1}}%
84   \expandafter\xdef\csname theH#1\endcsname{%
85     \expandafter\noexpand\csname theH#2\endcsname.%
86     \noexpand\the\noexpand\value{#1}}%
87 }
88 <latexrelease>\EndIncludeInRelease
89 </2ekernel | latexrelease>
90 <latexrelease>\IncludeInRelease{0000/00/00}{\@addtoreset}
91 <latexrelease>                                     {provide theHfoo commands}%%
92 <latexrelease>\def\@addtoreset#1#2{\expandafter\@cons\csname c1@#2\endcsname {{#1}}}}
93 <latexrelease>\EndIncludeInRelease
94 <*2ekernel>

(End of definition for \@addtoreset.)

95 </2ekernel>

```

`\@removefromreset`

```

96 <latexrelease>\IncludeInRelease{2018-04-01}
97 <latexrelease>                                     {\@removefromreset}{Add interfaces}%
98 <*2ekernel | latexrelease>
99 \def\@removefromreset#1#2{%

```

Even through this is internal and the programmer should know what he/she is doing we test here if counter #2 is defined. If not, the execution would run into a tight loop.

```

100   \ifundefined{c@#2}\relax
101   {\begingroup
102     \expandafter\let\csname c@#1\endcsname\@removefromreset
103     \def\@elt##1{%
104       \expandafter\ifx\csname c@##1\endcsname\@removefromreset
105       \else
106         \noexpand\@elt{##1}%
107       \fi}%
108     \expandafter\xdef\csname c1@#2\endcsname
109     {\csname c1@#2\endcsname}%
110     \endgroup}}

```

(End of definition for \@removefromreset.)

`\@ifbothcounters` Test if arg #1 and #2 are counters and if so execute #3.

```

111 \begingroup\catcode'\*=11 \lowercase{\endgroup
112 \def\@ifbothcounters#1#2#3{%
113   \begingroup\let\c@*\@undefined

```

```

114 \ifundefined{c@#1}{\nocounterr{#1}}%
115   {% else counter is defined
116     \ifundefined{c@#2}{\nocounterr{#2}}%
117     {% else both counter and within are defined
118       #3}}%
119   \endgroup}}

```

(End of definition for \@ifbothcounters.)

```

120 </2ekernel | latexrelease>
121 <latexrelease>\EndIncludeInRelease
122 <latexrelease>\IncludeInRelease{0000-00-00}
123 <latexrelease>           {\@removefromreset}{Add interfaces}%
124 <latexrelease>\let \@removefromreset \undefined
125 <latexrelease>\let \@ifbothcounters \undefined
126 <latexrelease>\EndIncludeInRelease
127 <*2ekernel>

```

`\counterwithout` New implementation using xparse and supporting an optional format argument.

`\counterwithin`

```

128 </2ekernel>
129 <*2ekernel | latexrelease>
130 <latexrelease>\IncludeInRelease{2021/11/15}%
131 <latexrelease>           {\counterwithout}{counter without/within}%
132 \NewDocumentCommand \counterwithout {s0{\arabic}mm}{%
133   \@ifbothcounters{#3}{#4}{%
134     \@removefromreset{#3}{#4}%
135     \IfBooleanF #1%
136       {\expandafter
137         \gdef\csname the#3\endcsname {#2{#3}}}%
138   }%
139 }
140 \NewDocumentCommand \counterwithin {s0{\arabic}mm}{%
141   \@ifbothcounters{#3}{#4}{%
142     \@addtoreset{#3}{#4}%
143     \IfBooleanF #1%
144     {\expandafter
145       \gdef\csname the#3\expandafter\endcsname
146       \expandafter
147         {\csname the#4\endcsname .#2{#3}}}%
148   }%
149 }

```

```

150 </2ekernel | latexrelease>
151 <latexrelease>\EndIncludeInRelease
152 <latexrelease>\IncludeInRelease{2018-04-01}
153 <latexrelease>           {\counterwithout}{counter without/within}%
154 <latexrelease>
155 <latexrelease>\def\counterwithout {\@ifstar\counterwithout@s\counterwithout@x}
156 <latexrelease>\def\counterwithout@s#1#2{%
157 <latexrelease>   \@ifbothcounters{#1}{#2}{\@removefromreset{#1}{#2}}
158 <latexrelease>\def\counterwithout@x#1#2{%
159 <latexrelease>   \@ifbothcounters{#1}{#2}%
160 <latexrelease>     {\@removefromreset{#1}{#2}%
161 <latexrelease>     \expandafter

```



```

162 <latexrelease> \gdef\csname the#1\expandafter\endcsname\expandafter
163 <latexrelease> {\expandafter
164 <latexrelease> \@arabic\csname c@#1\endcsname}}
165 <latexrelease>
166 <latexrelease>\def\counterwithin{\@ifstar\counterwithin@s\counterwithin@x}
167 <latexrelease>\def\counterwithin@s#1#2{%
168 <latexrelease> \@ifbothcounters{#1}{#2}{\@addtoreset{#1}{#2}}}

169 <latexrelease>\def\counterwithin@x#1#2{%
170 <latexrelease> \@ifbothcounters{#1}{#2}%
171 <latexrelease> {\@addtoreset{#1}{#2}%
172 <latexrelease> \expandafter
173 <latexrelease> \gdef\csname the#1\expandafter\endcsname\expandafter
174 <latexrelease> {\csname the#2\expandafter\endcsname\expandafter
175 <latexrelease> .\expandafter
176 <latexrelease> \@arabic\csname c@#1\endcsname}}
177 <latexrelease>
178 <latexrelease>\EndIncludeInRelease
179 <latexrelease>\IncludeInRelease{0000-00-00}
180 <latexrelease> {\counterwithout}{counter without/within}%
181 <latexrelease>\let \counterwithout \undefined
182 <latexrelease>\let \counterwithout@s \undefined
183 <latexrelease>\let \counterwithout@x \undefined
184 <latexrelease>\let \counterwithin \undefined
185 <latexrelease>\let \counterwithin@s \undefined
186 <latexrelease>\let \counterwithin@x \undefined
187 <latexrelease>\EndIncludeInRelease
188 <*2ekernel>

```

(End of definition for \counterwithout and \counterwithin.)

Numbering commands for definitions of `\theCOUNTER` and `\list` arguments.
All commands can now be used in text and math mode.

`\arabic` Representation of `<counter>` as arabic numerals. Changed 29 Apr 86 to make it print the obvious thing it COUNTER not positive.

```
189 \def\arabic#1{\expandafter\@arabic\csname c@#1\endcsname}
```

(End of definition for \arabic.)

`\roman` Representation of `<counter>` as lower-case Roman numerals.

```
190 \def\roman#1{\expandafter\@roman\csname c@#1\endcsname}
```

(End of definition for \roman.)

`\Roman` Representation of `<counter>` as upper-case Roman numerals.

```
191 \def\Roman#1{\expandafter\@Roman\csname c@#1\endcsname}
```

(End of definition for \Roman.)

`\alph` Representation of `<counter>` as a lower-case letter: 1 = a, 2 = b, etc.

```
192 \def\alph#1{\expandafter\@alph\csname c@#1\endcsname}
```

(End of definition for \alph.)

`\Alph` Representation of $\langle counter \rangle$ as an upper-case letter: 1 = A, 2 = B, etc.
193 `\def\Alph#1{\expandafter\@Alph\csname c@#1\endcsname}`
(End of definition for \Alph.)

`\fnsymbol` Representation of $\langle COUNTER \rangle$ as a footnote symbol: 1 = *, 2 = †, etc.
194 `\def\fnsymbol#1{\expandafter\@fnsymbol\csname c@#1\endcsname}`
(End of definition for \fnsymbol.)

`\@arabic` `\@arabic\F00counter` Representation of `\F00counter` as arabic numerals.
195 `\def\@arabic#1{\number #1} %% changed 29 Apr 86`
(End of definition for \@arabic.)

`\@roman` `\@roman\F00counter` Representation of `\F00counter` as lower-case Roman numerals.
196 `\def\@roman#1{\romannumeral #1}`
(End of definition for \@roman.)

`\@Roman` `\@Roman\F00counter` Representation of `\F00counter` as upper-case Roman numerals.
197 `\def\@Roman#1{\expandafter\@slowromancap\romannumeral #1@}`
(End of definition for \@Roman.)

`\@slowromancap` Fully expandable macro to change a roman number to uppercase.
198 `\def\@slowromancap#1{\ifx @#1% then terminate`
199 `\else`
200 `\if i#1I\else\if v#1V\else\if x#1X\else\if l#1L\else\if`
201 `c#1C\else\if d#1D\else \if m#1M\else#1\fi\fi\fi\fi\fi\fi`
202 `\expandafter\@slowromancap`
203 `\fi`
204 `}`
(End of definition for \@slowromancap.)

`\@alph` `\@alph\F00counter` Representation of `\F00counter` as a lower-case letter: 1 = a, 2 = b, etc.
205 `\def\@alph#1{%`
206 `\ifcase#1\or a\or b\or c\or d\or e\or f\or g\or h\or i\or j\or`
207 `k\or l\or m\or n\or o\or p\or q\or r\or s\or t\or u\or v\or w\or x\or`
208 `y\or z\else\@ctrerr\fi}`
(End of definition for \@alph.)

`\@Alph` `\@Alph\F00counter` Representation of `\F00counter` as an upper-case letter: 1 = A, 2 = B, etc.
209 `\def\@Alph#1{%`
210 `\ifcase#1\or A\or B\or C\or D\or E\or F\or G\or H\or I\or J\or`
211 `K\or L\or M\or N\or O\or P\or Q\or R\or S\or T\or U\or V\or W\or X\or`
212 `Y\or Z\else\@ctrerr\fi}`
(End of definition for \@Alph.)

`\@fnsymbol` Typesetting old fashioned footnote symbols. This can be done both in text or math mode now.

This macro is another example of an ever recurring problem in \TeX : Determining if something is text-mode or math-mode. It is imperative for the decision between text and math to be delayed until the actual typesetting is done as the code in question may go through an `\edef` or `\write` where an `\ifmode` test would be executed prematurely. Hence in the implementation below, `\@fnsymbol` is not robust in itself but the parts doing the actual typesetting are.

In the case of `\@fnsymbol` we make use of the robust command `\TextOrMath` which takes two arguments and typesets the first if in text-mode and the second if in math-mode. Note that in order for this command to make the correct decision, it must insert a `\relax` token if run under regular \TeX , which ruins any kerning between the preceding characters and whatever awaits typesetting. If you use $e\TeX$ as engine for $L\TeX$ (as recommended) this unfortunate side effect is not present.

```

213 </2ekernel>
214 <latexrelease>\IncludeInRelease{2015/01/01}{\@fnsymbol}{Use \TextOrMath}%
215 <*2ekernel | latexrelease>
216 \def\@fnsymbol#1{%
217   \ifcase#1\or \TextOrMath\textasteriskcentered *\or
218   \TextOrMath \textdagger \dagger\or
219   \TextOrMath \textdaggerdbl \ddagger \or
220   \TextOrMath \textsection \mathsection\or
221   \TextOrMath \textparagraph \mathparagraph\or
222   \TextOrMath \textbardbl \|\or
223   \TextOrMath {\textasteriskcentered\textasteriskcentered}{**}\or
224   \TextOrMath {\textdagger\textdagger}{\dagger\dagger}\or
225   \TextOrMath {\textdaggerdbl\textdaggerdbl}{\ddagger\ddagger}\else
226   \@ctrerr \fi
227 }%
228 </2ekernel | latexrelease>
229 <latexrelease>\EndIncludeInRelease
230 <latexrelease>\IncludeInRelease{0000/00/00}{\@fnsymbol}{Use \TextOrMath}%
231 <latexrelease>\def\@fnsymbol#1{\ensuremath{%
232 <latexrelease>   \ifcase#1\or *\or \dagger\or \ddagger\or \mathsection\or
233 <latexrelease>     \mathparagraph\or \|\or **\or \dagger\dagger
234 <latexrelease>     \or \ddagger\ddagger \else\@ctrerr\fi}}%
235 <latexrelease>\EndIncludeInRelease
236 <*2ekernel>

```

(End of definition for \@fnsymbol.)

`\TextOrMath` When using regular \TeX , we make this command robust so that it always selects the correct branch in an `\ifmode` switch with the usual disadvantage of ruining kerning. For the application we use it for here that shouldn't matter. The alternative would be to mimic `\IeC` from `inputenc` but then it will have the disadvantage of choosing the wrong branch if appearing at the beginning of an alignment cell. However, users of $e\TeX$ will be pleasantly surprised to get the best of both worlds and no bad side effects.

First some code for checking if we are running $e\TeX$ but making sure not to permanently turn `\protected` into `\relax`.

```

237 </2ekernel>
238 <latexrelease>\IncludeInRelease{2015/01/01}{\TextOrMath}{\TextOrMath}%
239 <*2ekernel | latexrelease>

```

```

240 \begingroup\expandafter\expandafter\expandafter\endgroup
241 \expandafter\ifx\csname protected\endcsname\relax

```

In case of ordinary T_EX we define `\TextOrMath` as a robust command but make sure it always grabs its arguments. If we didn't do this it might very well gobble spaces in the input stream.

```

242 \DeclareRobustCommand\TextOrMath{%
243   \ifmmode \expandafter\@secondoftwo
244   \else    \expandafter\@firstoftwo \fi}
245 \protected@edef\TextOrMath#1#2{\TextOrMath{#1}{#2}}
246 \else

```

For eT_EX the situation is similar. The robust macro is a hidden one so that we again avoid problems of gobbling spaces in the input.

```

247 \protected\expandafter\def\csname TextOrMath\space\endcsname{%
248   \ifmmode \expandafter\@secondoftwo
249   \else    \expandafter\@firstoftwo \fi}
250 \edef\TextOrMath#1#2{%
251   \expandafter\noexpand\csname TextOrMath\space\endcsname
252   {#1}{#2}}
253 \fi
254 </2ekernel | latexrelease>
255 <latexrelease>\EndIncludeInRelease
256 <latexrelease>\IncludeInRelease{0000/00/00}{\TextOrMath}{\TextOrMath}%
257 <latexrelease>\let\TextOrMath\@undefined
258 <latexrelease>\EndIncludeInRelease
259 <*2ekernel>

```

(End of definition for \TextOrMath.)

```

260 </2ekernel>

```

File 23

ltnlength.dtx

1 Lengths

```
\newlength Declare #1 to be a new length command.
\setlength Set the length command, #1, to the value #2.
\addtolength Increase the value of the length command, #1, by the value #2.
\settowidth Set the length, #1 to the width of a box containing #2.
\settoheight Set the length, #1 to the height of a box containing #2.
\settodepth Set the length, #1 to the depth of a box containing #2.

1 <*2ekernel>
2 \message{lengths,}

\newlength
3 \def\newlength#1{\@ifdefinable#1{\newskip#1}}
(End of definition for \newlength.)

\setlength
4 </2ekernel>
5 <latexrelease>\IncludeInRelease{2015/01/01}%
6 <latexrelease> \setlength}{Using \setlength with \dimen0}%
7 <*2ekernel | latexrelease>
8 \def\setlength#1#2{#1 #2\relax}
9 </2ekernel | latexrelease>
10 <latexrelease>\EndIncludeInRelease
11 <latexrelease>\IncludeInRelease{0000/00/00}%
12 <latexrelease> \setlength}{Using \setlength with \dimen0}%
13 <latexrelease>\def\setlength#1#2{#1#2\relax}
14 <latexrelease>\EndIncludeInRelease
15 <*2ekernel>
(End of definition for \setlength.)

\addtolength \relax added 24 Mar 86
16 \def\addtolength#1#2{\advance#1 #2\relax}
(End of definition for \addtolength.)

\settoheight The obvious analogs of \settowidth.
\settodepth
\settowidth
17 </2ekernel>
18 <*2ekernel | latexrelease>
\@settodim
19 <latexrelease>\IncludeInRelease{2024/11/01}%
20 <latexrelease> {\@settodim}{suspend tagging}%
21 \def\@settodim#1#2#3{\setbox\@tempboxa\hbox
22 {\SuspendTagging{\@settodim}#3\ResumeTagging{\@settodim}}#2#1\@tempboxa}
Clear the memory afterwards (which might be a lot).
23 \setbox\@tempboxa\box\voidb@x}
24 </2ekernel | latexrelease>
25 <latexrelease>\EndIncludeInRelease
```

```

26 <latexrelease>\IncludeInRelease{0000/00/00}%
27 <latexrelease>          {\@settodim}{suspend tagging}%
28 <latexrelease>\def\@settodim#1#2#3{\setbox\@tempboxa\hbox{##3}}#2#1\@tempboxa
29 <latexrelease>          \setbox\@tempboxa\box\voidb@x}
30 <latexrelease>\EndIncludeInRelease
31 <*/2kernel>

32 \DeclareRobustCommand\settoheight{\@settodim\ht}
33 \DeclareRobustCommand\settodepth {\@settodim\dp}
34 \DeclareRobustCommand\settowidth {\@settodim\wd}

```

(End of definition for \settoheight and others.)

\@settopoint This macro takes the contents of the skip register that is supplied as its argument and removes the fractional part to make it a whole number of points. This can be used in class files to avoid values like 345.4666666pt when calculating a dimension.

```

35 \def\@settopoint#1{\divide#1\p@\multiply#1\p@}
36 </2kernel>

```

(End of definition for \@settopoint.)

File 24

ltfssbas.dtx

This file contains the main implementation of the ‘low level’ font selection commands. See other parts of the L^AT_EX distribution, or *The L^AT_EX Companion* for higher level documentation of the L^AT_EX ‘New’ Font Selection Scheme.

Warning: The macro documentation is still basically the documentation from the first NFSS release and therefore in some cases probably not completely accurate.

1 Preliminary macros

We define a number of macros that will be used later.

```
1 <*2kernel>
2 \message{NFSS base,}
```

`\@nomath` `\@nomath` is used by most macros that will have no effect in math mode. It issues a warning message.

```
3 \def\@nomath#1{\relax\ifmmode
4   \@font@warning{Command \noexpand#1invalid in math mode}\fi}
```

(End of definition for `\@nomath`.)

`\no@alphabet@error` The macro `\no@alphabet@error` is called whenever the user requests a math *alphabet* that is not available in the current *version*. In math mode an error message is produced otherwise the command keeps silent. The argument is the name of the control sequence that identifies the math *alphabet*. The `\relax` at the beginning is necessary to prevent T_EX from scanning too far in certain situations.

```
5 \gdef\no@alphabet@error#1{\relax \ifmmode
6   \@latex@error{Math\space alphabet\space identifier\space
7     \noexpand#1is\space undefined\space in\space math\space
8     version\space ‘\math@version’}%
9   {Your\space requested\space math\space alphabet\space
10    is\space undefined\space in\space the\space current\space
11    math\space version.^^JCheck\space the\space spelling\space
12    or\space use\space the\space \noexpand\SetMathAlphabet\space
13    command.}
14   \fi}
```

(End of definition for `\no@alphabet@error`.)

`\new@mathgroup` We also give a new name to `\newfam` and `\fam` to avoid verbal confusion (see the introduction).³²

`\mathgroup`

```
15 %\def\new@mathgroup{\alloc@8\mathgroup\chardef\sixt@n}
16 \let\mathgroup\fam
17 %\let\newfam\new@mathgroup
18 \@onlypreamble\new@mathgroup
```

(End of definition for `\new@mathgroup` and `\mathgroup`.)

³²For the same reason it seems advisable to `\let\fam` and `\newfam` equal to `\relax`, but this is commented out to retain compatibility to existing style files.

2 Macros for setting up the tables

`\DeclareFontShape` The macro `\DeclareFontShape` takes 6 arguments:

```
19 \def\DeclareFontShape{\begingroup
```

First we restore the catcodes of all characters used in the syntax.

```
20 \nfss@catcodes
```

We use `\expandafter \endgroup` to restore catcode in case something goes wrong with the argument parsing (suggested by Tim Van Zandt)

```
21 \expandafter\endgroup
```

```
22 \DeclareFontShape@}
```

(End of definition for `\DeclareFontShape`.)

`\DeclareFontShape@`

```
23 </2ekernel>
```

```
24 <*2ekernel | latexrelease>
```

```
25 <latexrelease>\IncludeInRelease{2020/02/02}%
```

```
26 <latexrelease> \DeclareFontShape@}{Maybe drop one m}%
```

```
27 \def\DeclareFontShape@#1#2#3#4#5#6{%
```

```
28 \expandafter\ifx\csname #1+#2\endcsname\relax
```

```
29 \@latex@error{Font family ‘#1+#2’ unknown}\@eha
```

```
30 \else
```

If the series value is incorrectly specified with an extra “m”, e.g., “mc” instead of just “c”, drop the surplus “m” but keep the “m” if it is by its own. In that case also issue a warning that the declaration needs correction.

For this we compare the given value #3 with one where we may have dropped an “m”. If nothing has changes, fine. Otherwise there was a wrong value which is now corrected in `\reserved@b` so we use that and also issue a warning.

```
31 \edef\reserved@a{#3}%
```

```
32 \series@maybe@drop@one@m\reserved@a\reserved@b
```

```
33 \ifx\reserved@a\reserved@b\else
```

```
34 \@latex@note{Font shape #1/#2/#3/#4 has incorrect series
```

```
35 value ‘#3’.\MessageBreak It should not contain an ‘m’!
```

```
36 Please correct it.\MessageBreak Found}%
```

```
37 \fi
```

```
38 \expandafter
```

```
39 \xdef\csname#1/#2/\reserved@b/#4\endcsname
```

```
40 {\expandafter\noexpand\csname #5\endcsname}%
```

```
41 %
```

Most of the time #6 is empty so using `\let` to `\@empty` saves on space compared to using `\def`. That’s really one of the old space saving techniques and probably not necessary these days.

```
42 \def\reserved@a{#6}%
```

```
43 \global
```

```
44 \expandafter\let\csname#5\endcsname\expandafter\endcsname
```

```
45 \ifx\reserved@a\@empty
```

```
46 \@empty
```

```
47 \else
```

```
48 \reserved@a
```

```
49 \fi
```

```
50 \fi
```

```
51 }
```



```

52 </2ekernel | latexrelease>
53 <latexrelease>\EndIncludeInRelease
54 <latexrelease>\IncludeInRelease{0000/00/00}%
55 <latexrelease>          {\DeclareFontShape@}{Maybe drop one m}%
56 <latexrelease>
57 <latexrelease>\def\DeclareFontShape@#1#2#3#4#5#6{%
58 <latexrelease>  \expandafter\ifx\csname #1+#2\endcsname\relax
59 <latexrelease>    \@latex@error{Font family '#1+#2' unknown}\@eha
60 <latexrelease>  \else
61 <latexrelease>    \expandafter
62 <latexrelease>      \xdef\csname#1/#2/#3/#4\endcsname{\expandafter\noexpand
63 <latexrelease>                                     \csname #5\endcsname}%
64 <latexrelease>    \def\reserved@a{#6}%
65 <latexrelease>    \global
66 <latexrelease>    \expandafter\let\csname#5\endcsname\expandafter\endcsname
67 <latexrelease>    \ifx\reserved@a\@empty
68 <latexrelease>      \@empty
69 <latexrelease>    \else
70 <latexrelease>      \reserved@a
71 <latexrelease>    \fi
72 <latexrelease>  \fi
73 <latexrelease> }
74 <latexrelease>\EndIncludeInRelease
75 <*2ekernel>

```

(End of definition for \DeclareFontShape@.)

`\DeclareFixedFont` Define a direct font switch that avoids all overhead.

```

76 \def\DeclareFixedFont#1#2#3#4#5#6{%
77   \begingroup
78     \math@fontsfalse
79     \every@math@size{}%
80     \fontsize{#6}\z@
81     \usefont{#2}{#3}{#4}{#5}%
82     \global\expandafter\let\expandafter#1\the\font
83   \endgroup
84 }

```

(End of definition for \DeclareFixedFont.)

`\do@subst@correction`

```

85 \def\do@subst@correction{%
86   \xdef\subst@correction{%
87     \font@name
88     \global\expandafter\font
89     \csname \curr@fontshape/\f@size\endcsname
90     \noexpand\fontname\font
91     \relax}%

```

Calling `\subst@correction` after the current group means calling it after we have loaded the substitution font which is done inside a group.

```

92     \aftergroup\subst@correction
93 }

```

(End of definition for \do@subst@correction.)

`\DeclareFontFamily`

```
94 \def\DeclareFontFamily#1#2#3{%  
If we want fast checking for the encoding scheme we can just check for \T@. being defined.
```

```
95 % \@tempswafalse  
96 % \def\reserved@b{#1}%  
97 % \def\cdp@elt##1##2##3##4{\def\reserved@c{##1}%  
98 % \ifx\reserved@b\reserved@c \@tempswatrue\fi}%  
99 % \cdp@list  
100 % \if@tempswa  
101 \ifundefined{T@#1}%  
102 {%  
103 \@latex@error{Encoding scheme ‘#1’ unknown}\@eha  
104 }%  
105 {%
```

Now we have to define the macro `\langle#1\rangle+\langle#2\rangle` to contain `#3`. But since most of the time `#3` will be empty we use `\let` in a tricky way rather than a simple `\def` since this will save internal memory. We store the argument `#3` in a temporary macro `\reserved@a`.

```
106 \def\reserved@a{#3}%
```

We compare `\reserved@a` with `\@empty`. If these two are the same we `\let` the ‘extra’ macro equal to `\@empty` which is not the same as doing a `\let` to `\reserved@a` — the latter would blow one extra memory location rather than reusing the one from `\@empty`.

```
107 \global  
108 \expandafter\let\csname #1+#2\expandafter\endcsname  
109 \ifx \reserved@a\@empty  
110 \@empty  
111 \else \reserved@a  
112 \fi  
113 }%  
114 }
```

(End of definition for `\DeclareFontFamily`.)

`\cdp@list` We initialize the code page list to be empty.

```
115 \let\cdp@list\@empty  
116 \@onlypreamble\cdp@list
```

(End of definition for `\cdp@list`.)

`\cdp@elt`

```
117 \let\cdp@elt\relax  
118 \@onlypreamble\cdp@elt
```

(End of definition for `\cdp@elt`.)

`\DeclareFontEncoding`

```
119 \def\DeclareFontEncoding{%
```

First we start with ignoring all blanks and newlines since every surplus space in the second or third argument will come out in a weird place in the document.

```

120 \begingroup
121 \nfss@catcodes
122 \expandafter\endgroup
123 \DeclareFontEncoding@}
124 \@onlypreamble\DeclareFontEncoding

125 \def\DeclareFontEncoding@#1#2#3{%
126 \expandafter
127 \ifx\csname T@#1\endcsname\relax
128 \def\cdp@elt{\noexpand\cdp@elt}%
129 \xdef\cdp@list{\cdp@list\cdp@elt{#1}%
130 \{default@family\}\{default@series\}%
131 \{default@shape\}}%

```

To support encoding dependent commands (like accents) we initialise the command `\(encoding)`-cmd to be `\@changed@cmd`. (See `ltoutenc.dtx` for details.)

```

132 \expandafter\let\csname#1-cmd\endcsname\@changed@cmd
133 \else
134 \@font@info{Redefining font encoding #1}%
135 \fi

136 \global\@namedef{T@#1}{#2}%
137 \global\@namedef{M@#1}{\default@M#3}%

```

Keep a record of the last encoding being declared:

```

138 \xdef\LastDeclaredEncoding{#1}%
139 }
140 \@onlypreamble\DeclareFontEncoding@

```

(End of definition for \DeclareFontEncoding.)

`\LastDeclaredEncoding` The last encoding being declared by `\DeclareFontEncoding`.

```

141 \def\LastDeclaredEncoding{}

```

(End of definition for \LastDeclaredEncoding.)

`\DeclareFontSubstitution`

```

142 \def\DeclareFontSubstitution#1#2#3#4{%
143 \expandafter
144 \ifx\csname T@#1\endcsname\relax
145 \@latex@error{Encoding scheme ‘#1’ unknown}\@eha
146 \else
147 \begingroup

```

We loop through the `\cdp@list` and rebuild it anew in `\toks@` thereby replacing the defaults for the encoding in question with the new defaults. It is important to store the encoding to test against expanded in `\reserved@a` since it might just be `\LastDeclaredEncoding` that is passed as `#1`.

```

148 \edef\reserved@a{#1}%
149 \toks@{}%
150 \def\cdp@elt##1##2##3##4{%
151 \def\reserved@b{##1}%
152 \ifx\reserved@a\reserved@b

```

Here we use the new defaults but we use `##1` (i.e., the encoding name already stored previously) since we know that it is expanded.

```
153         \addto@hook\toks@\c@p@elt{##1}{##2}{##3}{##4}%
154         \else
```

If `\reserved@a` and `\reserved@b` differ then we simply copy from the old list to the new.

```
155         \addto@hook\toks@\c@p@elt{##1}{##2}{##3}{##4}%
156         \fi}%
157         \c@p@list
158         \xdef\c@p@list{\the\toks@}%
159     \endgroup
160     \global
161     \@namedef{D@#1}{%
162         \def\default@family{##2}%
163         \def\default@series{##3}%
164         \def\default@shape{##4}%
165         }%
166     \fi
167 }
168 \@onlypreamble\DeclareFontSubstitution
```

(End of definition for `\DeclareFontSubstitution`.)

`\DeclareFontEncodingDefaults`

```
169 \def\DeclareFontEncodingDefaults#1#2{%
170     \ifx\relax#1\else
171         \ifx\default@T\@empty\else
172             \@font@info{Overwriting encoding scheme text defaults}%
173             \fi
174             \gdef\default@T{#1}%
175         \fi
176         \ifx\relax#2\else
177             \ifx\default@M\@empty\else
178                 \@font@info{Overwriting encoding scheme math defaults}%
179                 \fi
180                 \gdef\default@M{#2}%
181             \fi
182         }
183     \@onlypreamble\DeclareFontEncodingDefaults
```

(End of definition for `\DeclareFontEncodingDefaults`.)

```
\default@T
\default@M
```

```
184 \let\default@T\@empty
185 \let\default@M\@empty
```

(End of definition for `\default@T` and `\default@M`.)

`\DeclareEncodingSubset` The declaration takes 3 mandatory arguments: an *encoding* for which a subsetting is wanted (currently always `TS1`, and most likely forever), the *font family* for which we declare the subset and finally the *subset* number, with a value between 0 (all of the encoding is supported) and 9 (many glyphs are missing).

For `TS1` the numbers have been chosen in a way that most fonts can be fairly correctly categorized, but the default settings are always conservative, that is they may claim that fewer glyphs are supported than there actually are.

As these days many font families are set up to end in `-LF` (lining figures), `-OsF` (oldstyle figures), etc. the declaration supports a shortcut: if the *font family* name ends in `-*` then the star gets replaced by these common ending, e.g.,

```
\DeclareEncodingSubset{TS1}{Alegreya-*}{2}
```

is the same as writing

```
\DeclareEncodingSubset{TS1}{Alegreya-LF}{2}
\DeclareEncodingSubset{TS1}{Alegreya-OsF}{2}
\DeclareEncodingSubset{TS1}{Alegreya-TLF}{2}
\DeclareEncodingSubset{TS1}{Alegreya-TOsF}{2}
```

If only some are needed then one can define them individually but in many cases all four are wanted, hence the shortcut.

The coding of the declaration has no error checking as it is mostly for internal use.

```
186 \def\DeclareEncodingSubset#1#2{%
187   \DeclareEncodingSubset@aux{#1}#2*\DeclareEncodingSubset@aux
188 }
189 \def\DeclareEncodingSubset@aux#1#2*#3\DeclareEncodingSubset@aux#4{%
if #3 is empty then there was no star, otherwise we define all four variants.
190   \expandafter\ifx\expandafter X\detokenize{#3}X%
191     \@DeclareEncodingSubset{#1}{#2}{#4}%
192   \else
193     \@DeclareEncodingSubset{#1}{#2LF}{#4}%
194     \@DeclareEncodingSubset{#1}{#2TLF}{#4}%
195     \@DeclareEncodingSubset{#1}{#2OsF}{#4}%
196     \@DeclareEncodingSubset{#1}{#2ToSF}{#4}%
197   \fi
198 }
```

The subset info is stored in a command with the name `\family:subset` so if that already exists we change otherwise declare a subset.

```
199 \def\@DeclareEncodingSubset#1#2#3{%
200   \ifundefined{#1:#2}%
201     {\@font@info{Setting #2 sub-encoding to #1/#3}}%
202     {\@font@info{Changing #2 sub-encoding to #1/#3}}%
```

This declaration should be usable in `.fd` files and therefore has to make its definition globally, because such files can get loaded in random places.

```
203   \global\@namedef{#1:#2}{#3}
```

(End of definition for `\DeclareEncodingSubset`.)

`\CheckEncodingSubset` The command `\CheckEncodingSubset` will check if the current font family has the right encoding subset to typeset a certain command. It takes five arguments as follows: first argument is either `\UseTextSymbol`, `\UseTextAccent` depending on whether or not the symbol is a text symbol or a text accent.

The second argument is the encoding from which this symbol should be fetched.

The third argument is either a fake accessor command or an error message. the code in that argument (if ever executed) receives two arguments: `#2` and `#5` of `\CheckEncodingSubset`.

Argument four is the subset encoding id to test against: if this value is higher than the subset id of the current font family then we typeset the symbol, i.e., execute `#1{#2}#5` otherwise it runs `#3#5`, e.g., to produce an error message or fake the glyph somehow.

Argument five is the symbol or accent command that is being checked.

For usage examples see definitions in `ltxtextcomp.dtx`.

```

204 </2ekernel>
205 < *2ekernel | latexrelease >
206 < latexrelease > \IncludeInRelease{2025/06/01}%
207 < latexrelease >          {\CheckEncodingSubset}{preload .fd file}%
208 \def\CheckEncodingSubset#1#2#3#4#5{%

```

If the `.fd` is not yet loaded we make an attempt to load it now because it may contain a `\DeclareEncodingSubset` declaration for the font.

```

209   \expandafter\ifx\csname #2:\f@family\endcsname\relax
210     \LoadFontDefinitionFile{#2}\f@family
211   \fi
212   \ifnum #4>%
213     \expandafter\ifx\csname #2:\f@family\endcsname\relax
214       0\csname #2:?\endcsname
215     \else
216       \csname #2:\f@family\endcsname
217     \fi
218   \relax
219   \expandafter\@firstoftwo
220 \else
221   \expandafter\@secondoftwo
222 \fi
223   {#1{#2}}{#3}%
224   #5%
225 }
226 </2ekernel | latexrelease >
227 < latexrelease > \EndIncludeInRelease
228 < latexrelease > \IncludeInRelease{0000/00/00}%
229 < latexrelease >          {\CheckEncodingSubset}{preload .fd file}%
230 < latexrelease >
231 < latexrelease > \def\CheckEncodingSubset#1#2#3#4#5{%
232 < latexrelease >   \ifnum #4>%
233 < latexrelease >     \expandafter\ifx\csname #2:\f@family\endcsname\relax
234 < latexrelease >       0\csname #2:?\endcsname
235 < latexrelease >     \else
236 < latexrelease >       \csname #2:\f@family\endcsname
237 < latexrelease >     \fi
238 < latexrelease >   \relax
239 < latexrelease >   \expandafter\@firstoftwo
240 < latexrelease > \else
241 < latexrelease >   \expandafter\@secondoftwo
242 < latexrelease > \fi
243 < latexrelease >   {#1{#2}}{#3}%
244 < latexrelease >   #5%
245 < latexrelease > }
246 < latexrelease > \EndIncludeInRelease
247 < *2ekernel >

```

(End of definition for \CheckEncodingSubset.)

`\DeclarePreloadSizes`

```
248 \def\DeclarePreloadSizes#1#2#3#4#5{%
249 \ifundefined{T@#1}%
250 {\@latex@error{Encoding scheme ‘#1’ unknown}\@eha}%
251 }
```

Don't know at the moment what this group here does!

```
252 \begingroup
```

We define a macro `\reserved@f`³³ that grabs the next *size* and loads the corresponding font. This is done by delimiting `\reserved@f`'s only argument by the token `,` (comma).

```
253 \def\reserved@f##1,{%
```

The end of the list will be detected when there are no more elements, i.e. when `\reserved@f`'s argument is empty. The trick used here is explained in Appendix D of the `TEXbook`: if the argument is empty the `\if` will select the first clause and `\let \reserved@f` equal to `\relax`. (We use the `>` character here since it cannot appear in font file names.)

```
254 \if>##1>%
255 \let\reserved@f\relax
256 \else
```

Otherwise, we define `\font@name` appropriately and call `\pickup@font` to do the work. Note that the requested `\curr@fontshape` combination must have been defined, or you will get an error. The definition of `\font@name` is carried out globally to be consistent with the rest of the code in this file.

```
257 \xdef\font@name{\csname#1/#2/#3/#4/##1\endcsname}%
258 \pickup@font
```

Now we forget the name of the font just loaded. More precisely, we set the corresponding control sequence to `\relax`. This means that later on, when the font is first used, the macro `\define@newfont` is called again to execute the 'extra' macro for this font.

```
259 \global\expandafter\let\font@name\relax
260 \fi
```

Finally we call `\reserved@f` again to process the next *size*. If `\reserved@f` was `\let` equal to `\relax` this will end the macro.

```
261 \reserved@f}%
```

We finish with reinserting the list of sizes after the `\reserved@f` macro and appending an empty element so that the end of the list is recognized properly.

```
262 \reserved@f#5,,%
263 \endgroup
264 }%
265 }
266 \@onlypreamble\DeclarePreloadSizes
```

(End of definition for `\DeclarePreloadSizes`.)

`\ifmath@fonts` We need a switch to decide if we have to switch math fonts. For this purpose we provide `\ifmath@fonts` that can be set to true or false by the `\S@...` macros depending on if math fonts are provided for this size or not. The default is of course to switch all fonts.

```
267 \newif\ifmath@fonts \math@fontstrue
```

³³We cannot use `\@tempa` since it is needed in `\pickup@font`.

(End of definition for `\ifmath@fonts`.)

`\DeclareMathSizes` `\DeclareMathSizes` takes the text size, math text size, math script size, and math
`\DeclareMathSizes*` scriptscript size as arguments and defines the right `\S@...` macro.

```
268 \def\DeclareMathSizes{%
269   \ifstar{\@DeclareMathSizes\math@fontsfalse}%
270   {\@DeclareMathSizes{}}
271 \@onlypreamble\DeclareMathSizes
```

(End of definition for `\DeclareMathSizes` and `\DeclareMathSizes*`.)

`\@DeclareMathSizes` This modification by Michael J. Downes on `comp.text.tex` on 2002/10/17 allows the user
to have settings such as

```
\DeclareMathSizes{9.5dd}{9.5dd}{7.4dd}{6.6dd}.
```

```
272 \</2kernel>
273 \<latexrelease>\IncludeInRelease{2015/01/01}{\@DeclareMathSizes}%
274 \<latexrelease>           {Arbitrary units in \DeclareMathSizes}%
275 \<*2kernel | latexrelease>
276 \def\@DeclareMathSizes #1#2#3#4#5{%
277   \@defaultunits\dimen@ #2pt\relax\@nnil
278   \if $#3$%
279     \expandafter\let\csname S@\strip@pt\dimen@\endcsname\math@fontsfalse
280   \else
281     \@defaultunits\dimen@ii #3pt\relax\@nnil
282     \@defaultunits\@tempdima #4pt\relax\@nnil
283     \@defaultunits\@tempdimb #5pt\relax\@nnil
284     \toks@{#1}%
285     \expandafter\xdef\csname S@\strip@pt\dimen@\endcsname{%
286       \gdef\noexpand\tf@size{\strip@pt\dimen@ii}%
287       \gdef\noexpand\sfs@size{\strip@pt\@tempdima}%
288       \gdef\noexpand\ssf@size{\strip@pt\@tempdimb}%
289       \the\toks@
290     }%
291     \fi
292   }%
293 \</2kernel | latexrelease>
294 \<latexrelease>\EndIncludeInRelease
295 \<latexrelease>\IncludeInRelease{0000/00/00}{\@DeclareMathSizes}%
296 \<latexrelease>           {Arbitrary units in \DeclareMathSizes}%
297 \<latexrelease>\def\@DeclareMathSizes#1#2#3#4#5{%
298 \<latexrelease>   \@defaultunits\dimen@#2pt\relax\@nnil
299 \<latexrelease>   \if $#3$%
300 \<latexrelease>     \expandafter \let
301 \<latexrelease>       \csname S@\strip@pt\dimen@\endcsname
302 \<latexrelease>       \math@fontsfalse
303 \<latexrelease>   \else
304 \<latexrelease>     \expandafter \gdef
305 \<latexrelease>       \csname S@\strip@pt\dimen@\endcsname
306 \<latexrelease>         {\gdef\tf@size{#3}\gdef\sfs@size{#4}%
307 \<latexrelease>           \gdef\ssf@size{#5}%
308 \<latexrelease>         #1%
309 \<latexrelease>       }%
310 \<latexrelease>   \fi}%
311 \<latexrelease>\EndIncludeInRelease
```



```

312 (*2kernel)
313 \@onlypreamble\@DeclareMathSizes
(End of definition for \@DeclareMathSizes.)

```

3 Selecting a new font

3.1 Macros for the user

`\fontencoding` As we said in the introduction a font is described by four parameters. We first define `\f@encoding` macros to specify the wanted *family*, *series*, or *shape*. These are simply recorded in internal macros `\f@family`, `\f@series`, and `\f@shape`, resp. We use `\edef`'s so that the arguments can also be macros.

```

314 \DeclareRobustCommand\fontencoding[1]{%
315   \expandafter\ifx\csname T@#1\endcsname\relax
316   \latex@error{Encoding scheme '#1' unknown}\@eha
317   \else
318   \edef\f@encoding{#1}%
319   \ifx\cf@encoding\f@encoding

```

If the new encoding is the same as the old encoding we have nothing to do. However, in case we had a sequence of several encoding changes without a `\selectfont` in-between we can save processing by making sure that `\enc@update` is `\relax`.

```

320   \let\enc@update\relax
321   \else

```

If current and new encoding differ we define the macro `\enc@update` to contain all updates necessary at `\selectfont` time.

```

322   \let\enc@update\@@enc@update
323   \fi
324   \fi
325 }

```

(End of definition for `\fontencoding` and `\f@encoding`.)

`\@@enc@update`

```

326 \def\@@enc@update{%

```

When `\@@enc@update` is executed `\f@encoding` holds the encoding name for the new encoding and `\cf@encoding` the name of the last active encoding.

We start by setting the init command for encoding dependent macros to `\@changed@cmd`.

```

327   \expandafter
328   \let
329   \csname\cf@encoding -cmd\endcsname
330   \@changed@cmd

```

Then we turn the one for the new encoding to `\@current@cmd` (see `ltoutenc.dtx` for further explanations).

```

331   \expandafter
332   \let
333   \csname\f@encoding-cmd\endcsname
334   \@current@cmd

```

We execute the default settings `\default@T`, followed by the one for the new encoding.

```
335     \default@T
336     \csname T@f@encoding\endcsname
```

Finally we change the default substitution values, disable `\enc@update` and make `\f@encoding` officially the current encoding.

```
337     \csname D@f@encoding\endcsname
338     \let\enc@update\relax
339     \let\cf@encoding\f@encoding
340 }
```

(End of definition for `\@enc@update`.)

`\enc@update` The default action in `\selectfont` is to do nothing.

```
341 \let\enc@update\relax
```

(End of definition for `\enc@update`.)

`\fontfamily`

`\f@family`

```
342 \DeclareRobustCommand\fontfamily[1]{\edef\f@family{#1}}
```

`\fontseries`

There are now defined later (and differently).

`\f@series`

```
343 %\DeclareRobustCommand\fontseries[1]{\edef\f@series{#1}}
```

`\fontshape`

```
344 %\DeclareRobustCommand\fontshape [1]{\edef\f@shape{#1}}
```

`\f@shape`

(End of definition for `\fontfamily` and others.)

`\usefont`

Some handy abbreviation if you want to get some particular font in the current size. If also the size should change one has to issue a `\fontsize` command first.

`\fontencoding` needs to do some setup work so we call that, but instead of calling `\fontfamily`, `\fontseries` and `\fontshape` it earlier versions of this code did, we now set `\f@family`, etc. directly. If we would call `\fontseries` or `\fontshape` as it was done in the past, they would now interact with the existing series and shape which is not desired if we intend to use an explicit font shape!

```
345 </2kernel>
346 <*2kernel | latexrelease>
347 <latexrelease> \IncludeInRelease{2021/06/01}%
348 <latexrelease>           {\usefont}{Force font face}%
349 \DeclareRobustCommand\usefont[4]{\fontencoding{#1}%
350   \edef\f@family{#2}%
351   \set@target@series{#3}%
352   \edef\f@shape{#4}%
```

Any earlier `\fontseries`, etc. should be canceled and we should switch unconditionally to the requested font face so we drop any code that may have been stored in `\delayed@f@adjustment`.

```
353   \let\delayed@f@adjustment\@empty
354   \selectfont
355   \ignorespaces}
356 </2kernel | latexrelease>
357 <latexrelease> \EndIncludeInRelease
358 <latexrelease> \IncludeInRelease{2020/02/02}%
359 <latexrelease>           {\usefont}{Drop m in usefont}%
360 <latexrelease>
361 <latexrelease> \DeclareRobustCommand\usefont[4]{\fontencoding{#1}%
```

```

362 <latexrelease> \edef\f@family{#2}%
363 <latexrelease> \set@target@series{#3}%
364 <latexrelease> \edef\f@shape{#4}\selectfont
365 <latexrelease> \ignorespaces}
366 <latexrelease>
367 <latexrelease>\EndIncludeInRelease
368 <latexrelease>\IncludeInRelease{0000/00/00}%
369 <latexrelease> \usefont}{Drop m in usefont}%
370 <latexrelease>
371 <latexrelease>\DeclareRobustCommand\usefont[4]{\fontencoding{#1}%
372 <latexrelease> \edef\f@family{#2}%
373 <latexrelease> \edef\f@series{#3}%
374 <latexrelease> \edef\f@shape{#4}\selectfont
375 <latexrelease> \ignorespaces}
376 <latexrelease>
377 <latexrelease>\EndIncludeInRelease
378 <*2ekernel)

```

(End of definition for \usefont.)

\linespread The command `\linespread` changes the current `\baselinestretch` by calling `\set@fontsize`. The values for `\f@size` and `\f@baselineskip` will be left unchanged.

```

379 \DeclareRobustCommand\linespread[1]
380   {\set@fontsize{#1}\f@size\f@baselineskip}

```

(End of definition for \linespread.)

\fontsize We also define a macro that allows to specify a size. In this case, however, we also need the value of `\baselineskip`. As the first argument to `\set@fontsize` we pass the current value of `\baselinestretch`. This will either match the internal value (in which case nothing changes) or it will be an updated value due to a user change of that macro using `\renewcommand`. If we would pass the internal `\f@linespread` such a change would be effectively overwritten by a size change.

```

381 \DeclareRobustCommand\fontsize[2]
382   {\set@fontsize\baselinestretch{#1}{#2}}

```

(End of definition for \fontsize.)

\f@linespread This macro holds the current internal value for `\baselinestretch`.

```

383 \let\f@family\@empty
384 \let\f@series\@empty
385 \let\f@shape\@empty
386 \let\f@size\@empty
387 \let\f@baselineskip\@empty
388 \let\f@linespread\@empty

```

(End of definition for \f@linespread.)

\cf@encoding

```

389 \let\f@encoding\@empty
390 \let\cf@encoding\@empty

```

(End of definition for \cf@encoding.)

`\@defaultunits` The function `\@defaultunits` when wrapped around a dimen or skip assignment supplies default units. Usage:

```
\@defaultunits\dimen@=#1pt\relax\@nnil
```

Note: the `\relax` is *important*. Other units can be substituted for the ‘pt’ if desired.

We use `\remove@to@nnil` as an auxiliary macros for `\@defaultunits`. It just has to gobble the supplied default unit ‘pt’ or whatever, if it wasn’t used in the assignment.

```
391 \def\@defaultunits{\afterassignment\remove@to@nnil}
```

(End of definition for `\@defaultunits`.)

`\strip@pt` This macro strips the characters pt produced by using `\the` on a dimen register.

```
\rem@pt
392 \begingroup
393   \catcode‘P=12
394   \catcode‘T=12
395   \lowercase{
396     \def\x{\def\rem@pt##1.##2PT{##1\ifnum##2>\z@.##2\fi}}
397   \expandafter\endgroup\x
398 \def\strip@pt{\expandafter\rem@pt\the}
```

(End of definition for `\strip@pt` and `\rem@pt`.)

`\mathversion` `\math@version` `\mathversion` takes the math *version* name as argument, defines `\math@version` appropriately and switches to the font selected forcing a call to `\glb@settings` if the *version* is known to the system.

```
399 \DeclareRobustCommand\mathversion[1]
400   {\@nomath\mathversion
401     \expandafter\ifx\csname mv@#1\endcsname\relax
402     \@latex@error{Math version ‘#1’ is not defined}\@eha\else
```

If there has been a frozen math version reset unconditionally to it if we are at `\@math@level` one, so that in the typical case of one bold symbol within the normal version this doesn’t allocate an additional math alphabet. If the nesting is deeper we do nothing, which means alphabet allocations accumulate until the end of the formula. One could do slightly better but that would mean keeping track of the allocations on all levels severately and this is likely to be overkill in nearly all situations.

```
403     \ifcsname mv@#1@frozen\endcsname
404     \ifnum \@math@level = \@ne
405       \unconditionally@reset@math@version {#1}%
406     \fi
407   \fi
408   \edef\math@version{#1}%
```

We need to force a math font setup both now and at the point where we return to the previous math version. Forcing a math font setup can simply be done by setting `\glb@currsz` to an invalid value since this will trigger the setup when the formula starts.

```
409     \gdef\glb@currsz{}
```

When the scope of the current `\mathversion` ends we need to restore the old setup. However this time we need to force it directly at least if we are inside math, otherwise we could wait. Another way to enhance this code here is to do the setting only if the version really has changed after all. This might be interesting in case of `amstext` and `boldsymbol`.

```
410         \aftergroup\glb@settings
411         \fi}
```

Resetting the math version unconditionally means that we have to copy the frozen version to `\mv@#1` and also reset the counter `\c@mv@#1` to the number of math alphabets allocated in the frozen version.

```
412 \ExplSyntaxOn
413 \cs_new_protected:Npn \unconditionally@reset@math@version #1 {
414   \cs_gset_eq:cc { mv@#1 }{ mv@#1@frozen }
415   \int_gset:cn { c@mv@#1 }{ \tl_use:c {g__nfss_frozen_mv_ #1 _tl} }
416 }
417 \ExplSyntaxOff
```

(End of definition for `\mathversion` and `\math@version`.)

If \TeX would support a hook just before the end of a formula (opposite of `\everymath` so to speak) the implementation of the algorithm would be much simpler because in that case we would set up the correct math fonts at this point without having to worry about incorrect settings due to nesting. The same would be true if in \LaTeX the use of `$` (as the primitive \TeX command) would be impossible and instead only a higher-level interface would be available. Note that this does not mean that a `$` couldn't be the short-hand for starting and stopping that higher-level interface, it only means that the direct \TeX function must be hidden.

Anyway, since we don't have this and won't have it in $\LaTeX 2_\epsilon$ we need to implement it in a somewhat slower way.

We test for the current math font setup on entry of a formula, i.e., on the hooks `\everymath` and `\everydisplay`. But since these hooks may contain user data we provide ourselves with an internal version of these hooks which stays frozen.

```
\frozen@everymath New internal names for \everymath and \everydisplay.
\frozen@everydisplay 418 \let\frozen@everymath\everymath
419 \let\frozen@everydisplay\everydisplay
```

(End of definition for `\frozen@everymath` and `\frozen@everydisplay`.)

```
\everymath Now we provide now user hooks that will be called in the frozen internals.
\everydisplay 420 \newtoks\everymath
421 \newtoks\everydisplay
```

(End of definition for `\everymath` and `\everydisplay`.)

```
\@math@level This counter records the nesting level of math within math.
422 \newcount\@math@level
```

(End of definition for `\@math@level`.)

`\frozen@everydisplay` Now we define the behaviour of the frozen hooks: first check the math setup then call the user hook.

The check code may push tokens after the math formula with `\aftergroup` and they would prevent a `$$` from dropping following spaces. We therefore use a switch to be set as the first thing after the group so that following code can determine if there was a display or some inline math (in the latter case we better not drop spaces). After setting the switch we also have to place `\ignorespaces` because setting the switch may be the only thing that happens after the display. The issue with handling of spaces was found in 2022, but it is really a bug fix for the code added in 2021/11.

```
423 </2ekernel>
424 <latexrelease>\IncludeInRelease{2021/11/15}
425 <latexrelease> {\frozen@everydisplay}{Handle spaces after math}%
426 <*2ekernel | latexrelease>
427 \frozen@everydisplay = {%
428   \aftergroup\@ignoretrue \aftergroup\ignorespaces
```

Record that we entered another math level.

```
429   \advance\@math@level\@ne
430   \check@mathfonts
431   \the\everydisplay}
```

(End of definition for \frozen@everydisplay.)

`\frozen@everymath` The frozen code for inline math is similar, except that here we do not want to drop following spaces.

```
432 \frozen@everymath = {%
433   \aftergroup\@ignorefalse
```

Record that we entered another math level.

```
434   \advance\@math@level\@ne
435   \check@mathfonts
436   \the\everymath}
```

(End of definition for \frozen@everymath.)

```
437 </2ekernel | latexrelease>
438 <latexrelease>\EndIncludeInRelease
439 <latexrelease>\IncludeInRelease{2020/10/01}
440 <latexrelease> {\frozen@everydisplay}{Handle spaces after math}%
441 <latexrelease>
442 <latexrelease>\frozen@everydisplay = {\check@mathfonts
443 <latexrelease>                                     \the\everydisplay}
444 <latexrelease>\frozen@everymath = {\check@mathfonts
445 <latexrelease>                                     \the\everymath}
446 <latexrelease>
447 <latexrelease>\EndIncludeInRelease
448 <*2ekernel>
```

`\curr@math@size` This holds locally the current math size.

```
449 \let\curr@math@size\@empty
```

(End of definition for \curr@math@size.)

3.2 Macros for loading fonts

`\pickup@font` The macro `\pickup@font` which is used in `\selectfont` is very simple: if the font name is undefined (i.e. not known yet) it calls `\define@newfont` to load it.

```
450 \def\pickup@font{%
451     \expandafter \ifx \font@name \relax
452         \define@newfont
453     \fi}
```

(End of definition for `\pickup@font`.)

`\split@name` `\pickup@font` assumes that `\font@name` is set but it is sometimes called when `\f@family`, `\f@series`, `\f@shape`, or `\f@size` may have the wrong settings (see, e.g., the definition of `\getanddefine@fonts`). Therefore we need a macro to extract font *family*, *series*, *shape*, and *size* from the font name. To this end we define `\split@name` which takes the font name as a list of characters of `\catcode 12` (without the backslash at the beginning) delimited by the special control sequence `\@nil`. This is not very complicated: we first ensure that `/` has the right `\catcode`

```
454 {\catcode'\/=12
```

and define `\split@name` so that it will define our private `\f@encoding`, `\f@family`, `\f@series`, `\f@shape`, and `\f@size` macros.

```
455 \gdef\split@name#1/#2/#3/#4/#5\@nil{\def\f@encoding{#1}%
456                                     \def\f@family{#2}%
457                                     \def\f@series{#3}%
458                                     \def\f@shape{#4}%
459                                     \def\f@size{#5}}}
```

(End of definition for `\split@name`.)

`\curr@fontshape` Abbreviation which may get removed again for speed.

```
460 \def\curr@fontshape{\f@encoding/\f@family/\f@series/\f@shape}
```

(End of definition for `\curr@fontshape`.)

`\define@newfont` Now we can tackle the problem of defining a new font.

```
461 \def\define@newfont{%
```

We have already mentioned that the token list that `\split@name` will get as argument must not start with a backslash. To reach this goal we will set the `\escapechar` to `-1` so that the `\string` primitive will not generate an escape character. To keep this change local we open a group. We use `\begingroup` for this purpose since `\define@newfont` might be called in math mode, and an empty `\bgroup... \egroup` would add an empty Ord atom to the math list and thus affect the spacing.

Also locally redefine `\typeout` so that ‘No file ...fd’ Warnings become Font Info message just sent to the log file.

```
462     \begingroup
463     \let\typeout\@font@info
464     \escapechar\m@ne
```

Then we extract *encoding scheme*, *family*, *series*, *shape*, and *size* from the font name. Note the four `\expandafter`’s so that `\font@name` is expanded first, then `\string`, and finally `\split@name`.

```
465     \expandafter\expandafter\expandafter
466     \split@name\expandafter\string\font@name\@nil
```

If the `\curr@fontshape` combination is not available, (i.e. undefined) we call the macro `\wrong@fontshape` to take care of this case. Otherwise `\extract@font` will load the external font for us.

```

467 %   \expandafter\ifx
468 %       \csname\curr@fontshape\endcsname \relax
469 %   \try@load@fontshape % try always
470 %   \fi
471 %   \expandafter\ifx
472 %       \csname\curr@fontshape\endcsname \relax
473 %   \wrong@fontshape\else

```

To allow substitution we call the `curr@fontshape` macro which usually will expand to `\relax` but may hold code for substitution (see `\subst@fontshape` definition).

```

474 %   \csname\curr@fontshape\endcsname
475 %   \extract@font\fi

```

We are nearly finished and must only restore the `\escapechar` by closing the group.

```

476 \endgroup}

477 \def\try@load@fontshape{%
478   \expandafter
479   \ifx\csname \f@encoding+\f@family\endcsname\relax
480     \@font@info{Trying to load font information for
481       \f@encoding+\f@family}%

```

We predefine this combination to be `\@empty` which means that next time we don't try again unnecessary in case we don't find a `.fd` file. If the file contains a `\DeclareFontFamily` command than this setting will be overwritten.

```

482   \global\expandafter\let
483   \csname\f@encoding+\f@family\endcsname\@empty

```

Set the catcodes used in the syntax, but do it only once (this will be restored at the end of the font loading group).

```

484   \nfss@catcodes
485   \let\nfss@catcodes\relax

```

For increased portability make the external filename monospace, but look for the (old style) mixed case filename if the first attempt fails.

On any monospace system this means that the file is looked for twice which takes up time and string space, but at least for this release Check for both names to give people time to re-install their private `fd` files with lowercase names.

```

486   \edef\reserved@a{%
487     \lowercase{%
488       \noexpand\InputIfFileExists{\f@encoding\f@family.fd}}%
489   \reserved@a\relax
490     {\@input@\f@encoding\f@family.fd}}%
491 \fi}

```

(End of definition for `\define@newfont`.)

`\nfss@catcodes` This macro should contain the standard `\catcode` assignments to all characters which are used in the commands found in an `.fd` file and which might have special `\catcodes` in the middle of a document. If necessary, this list can be extended in a package file using a suitable number of `\expandafter`, i.e.,


```
\expandafter\def\expandafter\nfss@catcodes
  \expandafter{\nfss@catcodes <additional settings>}
```

Note, that this macro might get executed several times since it is also called by `\DeclareFontShape`, thus it probably should not be misused as a general purpose hook.

```
492 \def\nfss@catcodes{%
```

We start by making `@` a letter and ignoring all blanks and newlines.

```
493   \makeatletter
494   \catcode'\ 9%
495   \catcode'\^~I9%
496   \catcode'\^~M9%
```

Then we set up `\`, `{`, `}`, `#` and `%` in case an `.fd` file is loaded during a verbatim environment.

```
497   \catcode'\\\z@
498   \catcode'\{\@ne
499   \catcode'\}\tw@
500   \catcode'\#6%
501   \catcode'\^7%
502   \catcode'\%14%
```

Then we make sure that the important syntax parts have the right `\catcode`.

```
503   \@makeother\<%
504   \@makeother\>%
505   \@makeother\*%
506   \@makeother\.%
507   \@makeother\-%
508   \@makeother\/%
509   \@makeother\[%
510   \@makeother\]%
511   \@makeother\'%
512   \@makeother\'%
513   \@makeother\"%
514 }
```

(End of definition for `\nfss@catcodes`.)

`\LoadFontDefinitionFile` Load and `.fd` files for some encoding and family (if it exists).

```
515 </2ekernel>
516 <*2ekernel | latexrelease>
517 <latexrelease>\IncludeInRelease{2020/02/02}%
518 <latexrelease>          {\LoadFontDefinitionFile}{Loading .fd files}%
519 \def\LoadFontDefinitionFile#1#2{%
520   \begingroup
521   \edef\f@encoding{#1}%
522   \edef\f@family{#2}%
523   \try@load@fontshape
524   \endgroup
525 }
526 </2ekernel | latexrelease>
527 <latexrelease>\EndIncludeInRelease
528 <latexrelease>\IncludeInRelease{0000/00/00}%
529 <latexrelease>          {\LoadFontDefinitionFile}{Loading .fd files}%
530 <latexrelease>
```

```

531 <latexrelease>\let\LoadFontDefinitionFile\@undefined
532 <latexrelease>\EndIncludeInRelease
533 <*2ekernel>

```

(End of definition for \LoadFontDefinitionFile.)

\DeclareFontFamilySubstitution The idea for this macro is stolen from the `substitutefont` package by Günter Milde, with some modifications and a new name.

Its purpose is to provide characters in a special encoding that are not available in the current font family to be taken from a different family that is visually compatible (or not if you choose badly). For example, you can match the GFS Didot Greek characters with T_EX Gyre Pagella (Palatino) by specifying

```
\DeclareFontFamilySubstitution{LGR}{qpl}{udidot}
```

This way if you ask for the LGR encoding in for the `qpl` family you get the characters from the `udidot` family substituted.

We need to ensure that the macro is defined with `\nfss@catcodes` in force (not quite sure why at the moment to be honest).

```

534 </2ekernel>
535 <*2ekernel | latexrelease>
536 <latexrelease>\IncludeInRelease{2020/02/02}%
537 <latexrelease>      {\DeclareFontFamilySubstitution}{Provide family substitution}%
538 \begingroup
539 \nfss@catcodes
540 \gdef\DeclareFontFamilySubstitution#1#2#3{%

```

We only provide a set of silent substitutions. The package also (re)declared the family, but this is incorrect in my eyes and it is better to handle that differently.

Of course the families may still need loading at this point and so we arrange for this. Otherwise we might run into trouble because the necessary `\DeclareFontFamily` has not been seen.

```

541   \LoadFontDefinitionFile{#1}{#2}%
542   \LoadFontDefinitionFile{#1}{#3}%
543   \DeclareFontShape{#1}{#2}{m}{it}{<->ssub * #3/m/it}{}%
544   \DeclareFontShape{#1}{#2}{m}{n}{<->ssub * #3/m/n}{}%
545   \DeclareFontShape{#1}{#2}{m}{sc}{<->ssub * #3/m/sc}{}%
546   \DeclareFontShape{#1}{#2}{m}{sl}{<->ssub * #3/m/sl}{}%

```

These days a few more shapes might be around, so we declare those too. If they don't exist then after the first substitution normal fallbacks will happen.

```

547   \DeclareFontShape{#1}{#2}{m}{sw}{<->ssub * #3/m/sw}{}%
548   \DeclareFontShape{#1}{#2}{m}{scit}{<->ssub * #3/m/scit}{}%
549   \DeclareFontShape{#1}{#2}{m}{scsl}{<->ssub * #3/m/scsl}{}%

```

Same game with `b` and `bx`, for other weights you are on your own:

```

550   \DeclareFontShape{#1}{#2}{b}{it}{<->ssub * #3/b/it}{}%
551   \DeclareFontShape{#1}{#2}{b}{n}{<->ssub * #3/b/n}{}%
552   \DeclareFontShape{#1}{#2}{b}{scit}{<->ssub * #3/b/scit}{}%
553   \DeclareFontShape{#1}{#2}{b}{scsl}{<->ssub * #3/b/scsl}{}%
554   \DeclareFontShape{#1}{#2}{b}{sc}{<->ssub * #3/b/sc}{}%
555   \DeclareFontShape{#1}{#2}{b}{sl}{<->ssub * #3/b/sl}{}%
556   \DeclareFontShape{#1}{#2}{b}{sw}{<->ssub * #3/b/sw}{}%
557   \DeclareFontShape{#1}{#2}{bx}{it}{<->ssub * #3/bx/it}{}%

```

```

558 \DeclareFontShape{#1}{#2}{bx}{n}{<->ssub * #3/bx/n}{}%
559 \DeclareFontShape{#1}{#2}{bx}{scit}{<->ssub * #3/bx/scit}{}%
560 \DeclareFontShape{#1}{#2}{bx}{scsl}{<->ssub * #3/bx/scsl}{}%
561 \DeclareFontShape{#1}{#2}{bx}{sc}{<->ssub * #3/bx/sc}{}%
562 \DeclareFontShape{#1}{#2}{bx}{sl}{<->ssub * #3/bx/sl}{}%
563 \DeclareFontShape{#1}{#2}{bx}{sw}{<->ssub * #3/bx/sw}{}%
564 }
565 \endgroup
566 </2ekernel | latexrelease>
567 <latexrelease>\EndIncludeInRelease
568 <latexrelease>\IncludeInRelease{0000/00/00}%
569 <latexrelease>      {\DeclareFontFamilySubstitution}{Provide family substitution}%
570 <latexrelease>
571 <latexrelease>\let\DeclareFontFamilySubstitution\@undefined
572 <latexrelease>\EndIncludeInRelease
573 <*2ekernel>

```

(End of definition for \DeclareFontFamilySubstitution.)

\DeclareErrorFont Declare the last resort shape! We assume that in this fontshape there is a 10pt font but it doesn't really matter. We only loose one macro name if the assumption is false. But at least the font should be there!

```

574 </2ekernel>
575 <*2ekernel | latexrelease>
576 <latexrelease>\IncludeInRelease{2019/10/01}%
577 <latexrelease>      {\DeclareErrorFont}{No side effects please}%
578 \def\DeclareErrorFont#1#2#3#4#5{%
579     \xdef\error@fontshape{%
580         \noexpand\expandafter\noexpand\split@name\noexpand\string
581         \expandafter\noexpand\csname#1/#2/#3/#4/#5\endcsname
582         \noexpand\@nil}%

```

Initialize all those internal variables which may or may not have values in the first seconds of NFSS' bootstrapping process. Later on such values will be updated when an encoding is selected, etc.

We definitely don't want to set `\f@encoding`; we can set all the others since if they are left "blank" any selection would grab "error default values" as well. However, this probably should go also—and now it did.

```

583 %     \gdef\f@encoding{#1}%
584     \gdef\default@family{#2}%
585     \gdef\default@series{#3}%
586     \gdef\default@shape{#4}%
587 }
588 </2ekernel | latexrelease>
589 <latexrelease>\EndIncludeInRelease
590 <latexrelease>\IncludeInRelease{0000/00/00}%
591 <latexrelease>      {\DeclareErrorFont}{No side effects please}%
592 <latexrelease>
593 <latexrelease>\def\DeclareErrorFont#1#2#3#4#5{%
594 <latexrelease>     \xdef\error@fontshape{%
595 <latexrelease>         \noexpand\expandafter\noexpand\split@name\noexpand\string
596 <latexrelease>         \expandafter\noexpand\csname#1/#2/#3/#4/#5\endcsname
597 <latexrelease>         \noexpand\@nil}%
598 <latexrelease>     \gdef\default@family{#2}%

```

```

599 <latexrelease>      \gdef\default@series{#3}%
600 <latexrelease>      \gdef\default@shape{#4}%
601 <latexrelease>      \global\let\f@family\default@family
602 <latexrelease>      \global\let\f@series\default@series
603 <latexrelease>      \global\let\f@shape\default@shape
604 <latexrelease>      \gdef\f@size{#5}%
605 <latexrelease>      \gdef\f@baselineskip{#5pt}%
606 <latexrelease>}
607 <latexrelease>\EndIncludeInRelease
608 <*2ekernel)
609 \@onlypreamble\DeclareErrorFont

```

(End of definition for \DeclareErrorFont.)

`\wrong@fontshape` Before we come to the macro `\extract@font` we have to take care of unknown `\curr@fontshape` combinations. The general strategy is to issue a warning and to try a default *shape*, then a default *series*, and finally a default *family*. If this last one also fails T_EX will go into an infinite loop. But if the defaults are set incorrectly one deserves nothing else!

```

610 </2ekernel)
611 <latexrelease>\IncludeInRelease{2015/01/01}{\wrong@fontshape}%
612 <latexrelease>      {Font substitution in preamble}%
613 <*2ekernel | latexrelease)
614 \def\wrong@fontshape{%
615   \csname D@f@encoding\endcsname % install defaults if in math

```

We remember the wanted `\curr@fontshape` combination which we will need in a moment.

```

616   \edef\reserved@a{\csname\curr@fontshape\endcsname}%
617   \ifx\last@fontshape\reserved@a
618     \errmessage{Corrupted NFSS tables}%
619     \error@fontshape
620   \else

```

Then we warn the user about the mess and set the shape to its default.

```

621   \let\f@shape\default@shape

```

If the combination is not known, try the default *series*.

```

622   \expandafter\ifx\csname\curr@fontshape\endcsname\relax
623     \let\f@series\default@series

```

If this is still undefined, try the default *family*. Otherwise give up. We never try to change the encoding scheme!

```

624   \expandafter
625     \ifx\csname\curr@fontshape\endcsname\relax
626     \let\f@family\default@family

```

If we change the font family and we are in the preamble then the corresponding `.fd` file may not been loaded yet. Therefore we try this now. Otherwise equating the requested font shape with the finally selected fontshape below will fail and can result in “NFSS tables corrupted”. After begin document that will not happen as all `.fd` files involved in substitution are loaded at `\begin{document}`.

```

627       \begingroup
628         \try@load@fontshape
629       \endgroup
630     \fi \fi
631 \fi

```

At this point a valid `\curr@fontshape` combination must have been found. We inform the user about this fact.

The `\expandafter\string` here stops \TeX adding the space that it usually puts after command names in messages. The similar construction with `\@undefined` just produces ‘undefined’, but saves a few tokens.

`\@wrong@font@char` is locally redefined in `\UseTextSymbol` from its normal (empty) definition, to report the symbol generating the font switch.

```
632     \font@warning{Font shape ‘\expandafter\string\reserved@a’
633                 \expandafter@gobble\string\@undefined\MessageBreak
634                 using ‘\curr@fontshape’ instead\@wrong@font@char}%
635     \global\let\last@fontshape\reserved@a
```

We change `\@defaultsubs` to produce a warning at the end of the document. The macro `\@defaultsubs` is initially `\relax` but gets changed here if some default font substitution happens. It is then executed in `\enddocument`.

```
636     \gdef\@defaultsubs{%
637         \font@warning{Some font shapes were not available, defaults
638                     substituted.\@gobbletwo}}%
```

If we substitute a `\curr@fontshape` combination by the default one we don’t want the warning to be printed out whenever this (unknown) combination is used. Therefore we globally `\let` the macro corresponding to the wanted combination equal to its substitution. This requires the use of four `\expandafter`’s since `\csname... \endcsname` has to be expanded before `\reserved@a` (i.e. the requested combination), and this must happen before the `\let` is executed.

```
639     \global\expandafter\expandafter\expandafter\let
640         \expandafter\reserved@a
641         \csname\curr@fontshape\endcsname
```

Now we can redefine `\font@name` accordingly. This *must* be done globally since it might occur in the group opened by `\define@newfont`. If we would this definition were local the closing `\endgroup` there would restore the old meaning of `\font@name` and then switch to the wrong font at the end of `\selectfont` although the correct font was loaded.

```
642     \xdef\font@name{%
643         \csname\curr@fontshape/\f@size\endcsname}%
```

The last thing this macro does is to call `\pickup@font` again to load the font if it is not defined yet. At this point this code will loop endlessly if the defaults are not well defined.

```
644     \pickup@font}
645     </2ekernel | latexrelease>
646     <latexrelease>\EndIncludeInRelease
647     <latexrelease>\IncludeInRelease{0000/00/00}{\wrong@fontshape}%
648     <latexrelease>           {Font substitution in preamble}%
649     <latexrelease>\def\wrong@fontshape{%
650     <latexrelease>     \csname D@f@encoding\endcsname
651     <latexrelease>     \edef\reserved@a{\csname\curr@fontshape\endcsname}%
652     <latexrelease>     \ifx\last@fontshape\reserved@a
653     <latexrelease>       \errmessage{Corrupted NFSS tables}%
654     <latexrelease>       \error@fontshape
655     <latexrelease>     \else
656     <latexrelease>       \let\f@shape\default@shape
657     <latexrelease>       \expandafter\ifx\csname\curr@fontshape\endcsname\relax
658     <latexrelease>         \let\f@series\default@series
659     <latexrelease>         \expandafter
```

```

660 <latexrelease>          \ifx\curname\curr@fontshape\endcurname\relax
661 <latexrelease>          \let\f@family\default@family
662 <latexrelease>          \fi \fi
663 <latexrelease> \fi
664 <latexrelease>          \@font@warning{Font shape
665 <latexrelease>          ‘\expandafter\string\reserved@a’
666 <latexrelease>          \expandafter\@gobble\string\@undefined
667 <latexrelease>          \MessageBreak
668 <latexrelease>          using ‘\curr@fontshape’ instead\@wrong@font@char}%
669 <latexrelease> \global\let\last@fontshape\reserved@a
670 <latexrelease> \gdef\@defaultsubs{%
671 <latexrelease>          \@font@warning{Some font shapes were not available,
672 <latexrelease>          defaults substituted.\@gobbletwo}}%
673 <latexrelease> \global\expandafter\expandafter\expandafter\let
674 <latexrelease>          \expandafter\reserved@a
675 <latexrelease>          \curname\curr@fontshape\endcurname
676 <latexrelease> \xdef\font@name{%
677 <latexrelease>          \curname\curr@fontshape/\f@size\endcurname}%
678 <latexrelease> \pickup@font}
679 <latexrelease>\EndIncludeInRelease
680 (*2ekernel)

```

(End of definition for \wrong@fontshape.)

\@wrong@font@char Normally empty but redefined in \UseTextSymbol so that the Font shape undefined message can refer to the symbol causing the problem.

```
681 \let\@wrong@font@char\@empty
```

(End of definition for \@wrong@font@char.)

\@@defaultsubs See above.

```
\@defaultsubs 682 \let\@defaultsubs\relax
```

(End of definition for \@@defaultsubs and \@defaultsubs.)

\strip@prefix In \extract@font we will need a way to recover the replacement text of a macro. This is done by the primitive \meaning together with the macro \strip@prefix (for the details see appendix D of the T_EXbook, p. 382).

```
683 \def\strip@prefix#1>{\}
```

(End of definition for \strip@prefix.)

4 Assigning math fonts to *versions*

\install@mathalphabet This is just another name for \gdef but we can redefine it if necessary later on.

```
684 \let\install@mathalphabet\gdef
```

(End of definition for \install@mathalphabet.)

\math@fonts

```
685 \let\math@fonts\@empty
```

(End of definition for \math@fonts.)

`\select@group` `\select@group` has four arguments: the new $\langle\text{math alphabet identifier}\rangle$ (a control sequence), the $\langle\text{math group number}\rangle$, the extra macro for math mode and the `\curr@fontshape` definition macro name. We first check if we are in math mode.

```
686 %\def\select@group#1#2#3{\relax\ifmmode
```

We do these things locally using `\begingroup` instead of `\bgroup` to avoid the appearance of an empty Ord atom on the math list.

```
687 % \begingroup
```

We set the math fonts for the *family* in question by calling `\getanddefine@fonts` in the correct environment.

```
688 % \escapechar\m@ne
```

```
689 % \getanddefine@fonts{\csname c@mv@math@version\endcsname}#3%
```

We globally select the math fonts...

```
690 % \globaldefs\@ne \math@fonts
```

... and close the group to restore `\globaldefs` and `\escapechar`.

```
691 % \endgroup
```

As long as no *size* or *version* change occurs the $\langle\text{math alphabet identifier}\rangle$ should simply switch to the installed *math group* instead of calling `\select@group` unnecessarily. So we globally redefine the first argument (the new $\langle\text{math alphabet identifier}\rangle$) to expand into a `\mathgroup` switch and then select this *alphabet*. Note that this redefinition will be overwritten by the next call to a *version* macro. The original code for the end of `\select@group` was

```
\gdef#1{#3\mathgroup #2}#1\fi}
```

i.e. first redefining the $\langle\text{math alphabet identifier}\rangle$ and then calling the new definition to switch to the wanted $\langle\text{math group}\rangle$. Now we define the $\langle\text{math alphabet identifier}\rangle$ as a call to the `\use@mathgroup` command.

```
692 % \xdef#1{\noexpand\use@mathgroup\noexpand#2%
```

```
693 % {\number\csname c@mv@math@version\endcsname}}%
```

But this is not sufficient, as we learned the hard way. The problem here is that the loading of the fonts that comprise the alphabet identifier `#1`, as well as the necessary math font assignments is deferred until it is used. This is OK so far, but if the fonts are switched within the current formula (which may happen if a sub-formula is a box that contains a math version switch) the font assignments for `#1` are not restored unless `#1` is used again. This is disastrous since TeX sees the wrong fonts at the end of the math formula, when it converts the math list into a horizontal list.

This is taken into account as follows: When a math alphabet identifier is used for the first time in a certain version it modifies the corresponding macro `\mv@<version>` so that it calls `\getanddefine@fonts` directly in future as well. We use the macro `\extract@alph@from@version` to do this. It takes the math alphabet identifier `#1` and the math version macro as arguments.

```
694 % \expandafter\extract@alph@from@version
```

```
695 % \csname mv@math@version\expandafter\endcsname
```

```
696 % \expandafter{\number\csname c@mv@math@version\endcsname}%
```

```
697 % #1%
```

```
698 % \stepcounter{mv@math@version}%
```

Finally, it is not possible to simply call the new definition since we have an argument (the third argument of `\use@mathgroup` or more exactly the argument of `\math@egroup` if the `margid` option is in force) which would swallow our closing `\fi`. So we use the `\expandafter` technique to remove the `\fi` before the `\use@mathgroup` is expanded.

```
699 %\expandafter #1\fi}
```

(End of definition for `\select@group`.)

`\extract@alph@from@version`

We proceed to the definition of the macro `\extract@alph@from@version`. As stated above, it takes a math alphabet identifier and a math version macro (e.g. `\mv@normal`) as its arguments.

```
700 \def\extract@alph@from@version#1#2#3{%
```

To extract and replace the definition of math alphabet identifier `#3` in macro `#1` we have to recall how this definition looks like: Somewhere in the replacement text of `#1` there is the sequence

```
\install@mathalphabet<math alphabet identifier> #3{%
  <Definitions for >#3}
```

Hence, the first thing we do is to extract the tokens preceding this definitions, the definition itself, and the tokens following it. To this end we define one auxiliary macro `\reserved@a`.

```
701 \def\reserved@a##1\install@mathalphabet#3##2##3\@nil{%
```

When `\reserved@a` is expanded, it will have the tokens preceding the definition in question in its first argument (`##1`), the following tokens in its third argument (`##3`), and the replacement text for the math alphabet identifier `#3` in its second argument. (`##2`). This is then recorded for later use in a temporary macro `\reserved@b`.

```
702 \def\reserved@b{##2}%
```

Additionally, we define a macro `\reserved@c` to reconstruct the definitions for the math version in question from the tokens that will remain unchanged (`##1` and `##3`) and the yet to build new definitions for the math alphabet identifier `#3`.

```
703 \def\reserved@c###1{\gdef#1{##1###1##3}}%
```

Then we execute our auxiliary macro.

```
704 \expandafter\reserved@a#1\@nil
```

OK, so now we have to build the new definition for `#3`. To do so, we first extract the interesting parts out of the old one. The old definition looks like:

```
\select@group<math alphabet identifier>
  <math group number><math extra part>
<curr@fontshape definition>
```

So we define a new temporary macro `\reserved@a` that extracts these parts.

```
705 \def\reserved@a\select@group#3##1##2\@nil{%
```

This macro can now directly rebuild the math version definition by calling `\reserved@c`:

```
706 \reserved@c{%
707 \getanddefine@fonts{#2}##2%
708 \install@mathalphabet#3{%
709 \relax\ifmmode \else \non@alpherr#3\fi
710 \use@mathgroup##1{#2}}%
```


In addition it defines the alphabet the way it should be used from now on.

```
711     \gdef#3{\relax\ifmmode \else \non@alpherr#3\fi
712         \use@mathgroup##1{#2}}}%
```

Finally, we only have to call this macro `\reserved@a` on the old definitions recorded in `\reserved@b`:

```
713     \expandafter\reserved@a\reserved@b\@nil
714     }
```

(End of definition for \extract@alph@from@version.)

`\math@bgroup` Here are the default definitions for `\math@bgroup` and `\math@egroup`. We use `\bgroup` instead of `\begingroup` to avoid ‘leaking out’ of style changes. This has the side effect of always producing mathord atoms.

```
715 \let\math@bgroup\bgroup
716 \def\math@egroup#1{#1\egroup}
```

(End of definition for \math@bgroup and \math@egroup.)

`\calculate@math@sizes` Here is the default definition for `\calculate@math@sizes` a more elaborate interface is under testing in `mthscale.sty`.

```
717 \gdef\calculate@math@sizes{%
718     \@font@info{Calculating\space math\space sizes\space for\space
719         size\space <\f@size>}%
720     \dimen@f@size \p@
721     \@tempdimb \defaultscritratio \dimen@
722     \dimen@ \defaultscritscritratio \dimen@
723     \expandafter\xdef\csname S@\f@size\endcsname{%
724         \gdef\noexpand\tf@size{\f@size}%
725         \gdef\noexpand\sf@size{\strip@pt\@tempdimb}%
726         \gdef\noexpand\ssf@size{\strip@pt\dimen@}%
727         \noexpand\math@fontstrue}}
```

(End of definition for \calculate@math@sizes.)

`\defaultscritratio` The default ratio for math sizes is:
`\defaultscritscritratio` 1 to `\defaultscritratio` to `\defaultscritscritratio`.
 By default this is 1 to .7 to .5.

```
728 \def\defaultscritratio{.7}
729 \def\defaultscritscritratio{.5}
```

(End of definition for \defaultscritratio and \defaultscritscritratio.)

`\noaccents@` If we don’t have a definition for `\noaccents@` we provide a dummy.

```
730 \ifx\noaccents@\@undefined
731     \let\noaccents@\@empty
732 \fi
```

(End of definition for \noaccents@.)

`\showhyphens` The `\showhyphens` command must be redefined since the version in `plain.tex` uses `\tenrm`. We have also made some further adjustments for its use in L^AT_EX.

```

733 </2ekernel>
734 <latexrelease>\IncludeInRelease{2017/01/01}{\showhyphens}%
735 <latexrelease>                {XeTeX support for \showhyphens}%
736 <*2ekernel | latexrelease>
737 \ifx\XeTeXcharclass\undefined
Version for engines other than XeTEX.
738 \DeclareRobustCommand\showhyphens[1]{%
739   \setbox0\vbox{%
740     \color@begingroup
741     \everypar{}%
742     \parfillskip\z@skip\hsize\maxdimen
743     \normalfont
744     \pretolerance\m@ne\tolerance\m@ne\hbadness\z@\showboxdepth\z@\ #1%
745     \color@endgroup}}

```

746 `\else`

XeT_EX version. When using system fonts XeT_EX reports consecutive runs of characters as a single item in box logging, which means the standard `\showhyphens` does not work. This version typesets the text into a narrow box to force hyphenation and then reconstructs a horizontal list with explicit hyphens to generate the display. Note that the `lmr` OpenType font is forced, this works even if the characters are not in the font as hyphenation is attempted due to the width of the space and hyphen character. It would generate spurious Missing Character warnings in the log, these are however suppressed from the terminal and log output by ensuring that `\tracinglostchars` is locally zero.

```

747 \DeclareRobustCommand\showhyphens[1]{%
748   \setbox0\vbox{%
749     \usefont{TU}{lmr}{m}{n}%
750     \hsize 1sp %
751     \hbadness\@M
752     \hfuzz\maxdimen
753     \tracingonline\z@
754     \tracinglostchars\z@
755     \everypar={}%
756     \leftskip\z@skip
757     \rightskip\z@skip
758     \parfillskip\z@skip
759     \hyphenpenalty=-\@M
760     \pretolerance\m@ne
761     \interlinepenalty\z@
762     \clubpenalty\z@
763     \widowpenalty\z@
764     \brokenpenalty1127 %
765     \setbox\z@\hbox{}%
766     \noindent
767     \hskip\z@skip
768     #1%
769     \par

```

Note here we stop the loop if made no progress, non-removable items may mean that we can not process the whole list (which would be testable as `\lastnodetype=-1`).

770 `\loop`

```

771 \tempwafalse
772 \ifnum\lastnodetype=11\unskip\tempwatruer\fi
773 \ifnum\lastnodetype=12\unkern\tempwatruer\fi
774 \ifnum\lastnodetype=13 %
775 \count@\lastpenalty
776 \unpenalty\tempwatruer
777 \fi
778 \ifnum\lastnodetype=\@ne
779 \setbox\tw@\lastbox\tempwatruer
780 \setbox0\hbox{\unhbox\tw@\unskip\unskip\unpenalty
781 \ifnum\count@=1127 \else\ \fi
782 \unhbox0}%
783 \count@\z@
784 \fi
785 \if@tempswa
786 \repeat
787 \hbadness\z@
788 \hsize\maxdimen
789 \showboxdepth\z@
790 \tolerance\m@ne
791 \hyphenpenalty\z@
792 \noindent\unhbox\z@
793 }}
794 \fi
795 </2ekernel | latexrelease>
796 <latexrelease>\EndIncludeInRelease
797 <latexrelease>\IncludeInRelease{0000/00/00}{\showhyphens}%
798 <latexrelease> {XeTeX support for \showhyphens}%
799 <latexrelease>\gdef\showhyphens#1{%
800 <latexrelease> \setbox0\vbox{%
801 <latexrelease> \color@begingroup
802 <latexrelease> \everypar{ }%
803 <latexrelease> \parfillskip\z@skip\hsize\maxdimen
804 <latexrelease> \normalfont
805 <latexrelease> \pretolerance\m@ne\tolerance\m@ne
806 <latexrelease> \hbadness\z@\showboxdepth\z@\ #1%
807 <latexrelease> \color@endgroup}}
808 <latexrelease>\EndIncludeInRelease
809 <*2ekernel>

```

(End of definition for \showhyphens.)

\addto@hook We need a macro to add tokens to a hook.

```
810 \long\def\addto@hook#1#2{#1\expandafter{\the#1#2}}
```

(End of definition for \addto@hook.)

\@vpt

```
811 \def\@vpt{5}
```

(End of definition for \@vpt.)

\@vipt

```
812 \def\@vipt{6}
```

(End of definition for \@vipt.)

\@viipt

813 \def\@viipt{7}

(End of definition for \@viipt.)

\@viipt

814 \def\@viipt{8}

(End of definition for \@viipt.)

\@ixpt

815 \def\@ixpt{9}

(End of definition for \@ixpt.)

\@xpt

816 \def\@xpt{10}

(End of definition for \@xpt.)

\@xipt

817 \def\@xipt{10.95}

(End of definition for \@xipt.)

\@xiipt

818 \def\@xiipt{12}

(End of definition for \@xiipt.)

\@xivpt

819 \def\@xivpt{14.4}

(End of definition for \@xivpt.)

\@xvipt

820 \def\@xvipt{17.28}

(End of definition for \@xvipt.)

\@xxpt

821 \def\@xxpt{20.74}

(End of definition for \@xxpt.)

\@xxvpt

822 \def\@xxvpt{24.88}

(End of definition for \@xxvpt.)

823 $\langle /2\text{kernel} \rangle$

File 25

ltfssaxes.dtx

This file contains the implementation for handling extra axes by splitting the series and the shape values into sub-categories. See other parts of the L^AT_EX distribution, or *The L^AT_EX Companion* for higher level documentation of the L^AT_EX Font Selection Scheme.

```
1 <2ekernel>\message{NFSS axes,}
```

Everything in the this file got introduced 2020/02/02, so we use large rollback chunks, only interrupted if necessary.

```
2 <*2ekernel | latexrelease>
```

```
3 <latexrelease>\IncludeInRelease{2025/06/01}%
```

```
4 <latexrelease> {\DeclareFontSeriesChangeRule}{Series change rules}%
```

1 Changing the font series

In the original NFSS implementation the series was a single attribute stored in `\f@series` and so one always had to specify both weight and width together. This meant that it was impossible to typeset a paragraph in a condensed font and inside have a few words in bold weight (but still condensed) without doing this manually by requesting `\fontseries{bc}\selectfont`.

The new implementation now works differently, by looking at both the current value of `\f@series` and the requested new series; these are then used to select a new series value. Thus, if the current series is `c` and we ask for `b` we now get `bc`.

This is done by consulting a simple lookup table. This table is configurable (though most likely that flexibility will seldom if ever be needed). Adding or changing entries in this table is done with `\DeclareFontSeriesChangeRule`.

1.1 The series lookup table

`\DeclareFontSeriesChangeRule`

The macro `\DeclareFontSeriesChangeRule` defines entries in a simple database (implemented as a set of commands) that define mappings from the current series and a requested new series to a result series (and additionally offers an alternative if the desired one is unavailable):

```
#1 current \f@series
```

```
#2 requested new series
```

```
#3 result (provided this series exists in the given font family)
```

```
#4 alternative result (if #3 does not exist)
```

If an `.fd` file has its own substitution rules then `#3` exist and thus `#4` is not applied.

If there is no matching database entry, or if neither the result nor the alternative result exists in this font family, then the requested new series is used (which then may trigger substitutions later on).

```
5 \def\DeclareFontSeriesChangeRule#1#2#3#4{%
```

```
6   \@namedef{series@#1@#2}{#3}#4}}
```

(End of definition for \DeclareFontSeriesChangeRule.)

1.2 Mapping rules for series changes

The rules set up use explicit series values not `\.default` indirections; my current feeling is that this is in fact better.

With 9 weight values (from `ul` to `ub`) and 9 width values (from `uc` to `ux`), this table is now rather large (far more than 1000 entries), but, on the other hand, the table doesn't change and accessing rules is fast when using a table implemented in this way.

We could alternatively split the axis and maintain weight and width separately, but that would take more processing time and would not allow for setting up explicit exceptions nicely (not sure that this would ever get used though).

Design considerations for mapping entries:

- We make `m` reset both weight and width (as this is how it always worked). To reset just the width `?m` is provided and to reset just the weight `m?` is provided.
- We support “`m<width>`” and “`<weight>m`”, e.g., `mec` to mean “go to medium weight and extra-condensed width”. At the end of the process we automatically drop any leftover `m` in the series name (unless it is just a single `m`).
- If there is no table entry then the requested series is used unconditionally. This means that we usually do not need entries where the second argument (the requested series) and the third argument (the result series) are identical (unless we want to use the fourth argument to specify an alternative result series). In particular, this means:
 - Any request for `m` needs no entry, i.e., there are no entries which have `m` as the second argument.
 - Any request to set both weight and width (e.g., `sbx` or `ulc`) needs no entry. For that reason, there are no entries which have a weight+width as second argument (except for cases involving `bx`, see below). In particular, this is also true for cases involving `m`, e.g., `bm` (bold medium width) which automatically gets reduced to `b`, or `mc` (medium weight condensed) which becomes `c` as a result.

There is one exception: we do have rules for a request for `bx`; this is because the Computer Modern fonts (or related families like Latin Modern) need `bx` as their bold series default (`\bfdefault`), since they only have a `b` series for a restricted set of font faces. Many other font families have only `b` but no `bx`, so we have explicit rules that fall back from `bx` to `b`.

- For each combination of a non-`m` `<weight>` and a non-`m` `<width>`, there are 19 entries which have “`<weight><width>`” as first argument: 8 of them have the weight values from `ul` to `ub` except `m` as second argument, another 8 have the width values from `uc` to `ux` except `m` as second argument, and (further down in this file) there is one entry which has `bx` as second argument and two further entries which have `m?` and `?m` as second argument. Rules which have `m` or a weight+width (other than `bx`) as second argument aren't needed (see above).
- For each non-`m` `<weight>`, there are at least 11 entries which have “`<weight>`” as first argument: 8 of them have the width values from `uc` to `ux` except `m` as second argument, and (further down in this file) there is one entry which has `bx` as second argument and two further entries which have `m?` and `?m` as second argument. Rules which have a single weight value as second argument aren't needed because the

second and third arguments would then be identical. In some cases, there are extra rules which make use of the fourth argument to specify an alternative result series.

- Similarly, for each non-*m* (*width*), there are at least 11 entries which have “*<width>*” as first argument: 8 of them have the weight values from *ul* to *ub* except *m* as second argument, and (further down in this file) there is one entry which has *bx* as second argument and two further entries which have *m?* and *?m* as second argument. Rules which have a single width value as second argument aren’t needed because the second and third arguments would then be identical. In some cases, there are extra rules which make use of the fourth argument to specify an alternative result series.
- Only a few entries have “alternative” values.

The idea is that you don’t want the normal substitution to kick in because that would reset the shape first and it may be better to stay with *b* when a change to *c* is requested and *bc* doesn’t exist, than to go to first change the shape to *n* and then find that *bc/n* doesn’t exist either and thus ending up with *m/n*.

We mainly do this when *sb*, *eb*, or *ub* is requested but can’t be fulfilled. In that case it is better to try to stay with some sort of bold rather than ending up with *m*. There are some other cases where an “alternative” value is specified; these are explained below in the appropriate places.

```

7 \DeclareFontSeriesChangeRule {uluc}{ul} {uluc} {}
8 \DeclareFontSeriesChangeRule {uluc}{el} {eluc} {}
9 \DeclareFontSeriesChangeRule {uluc}{l} {luc} {}
10 \DeclareFontSeriesChangeRule {uluc}{sl} {sluc} {}
11 \DeclareFontSeriesChangeRule {uluc}{sb} {sbuc} {buc}
12 \DeclareFontSeriesChangeRule {uluc}{b} {buc} {}
13 \DeclareFontSeriesChangeRule {uluc}{eb} {ebuc} {buc}
14 \DeclareFontSeriesChangeRule {uluc}{ub} {ubuc} {buc}
15 \DeclareFontSeriesChangeRule {uluc}{uc} {uluc} {}
16 \DeclareFontSeriesChangeRule {uluc}{ec} {ulec} {}
17 \DeclareFontSeriesChangeRule {uluc}{c} {ulc} {}
18 \DeclareFontSeriesChangeRule {uluc}{sc} {ulsc} {}
19 \DeclareFontSeriesChangeRule {uluc}{sx} {ulsx} {}
20 \DeclareFontSeriesChangeRule {uluc}{x} {ulx} {}
21 \DeclareFontSeriesChangeRule {uluc}{ex} {ulex} {}
22 \DeclareFontSeriesChangeRule {uluc}{ux} {ulux} {}

23 \DeclareFontSeriesChangeRule {ulec}{ul} {ulec} {}
24 \DeclareFontSeriesChangeRule {ulec}{el} {elec} {}
25 \DeclareFontSeriesChangeRule {ulec}{l} {lec} {}
26 \DeclareFontSeriesChangeRule {ulec}{sl} {slec} {}
27 \DeclareFontSeriesChangeRule {ulec}{sb} {sbec} {bec}
28 \DeclareFontSeriesChangeRule {ulec}{b} {bec} {}
29 \DeclareFontSeriesChangeRule {ulec}{eb} {ebec} {bec}
30 \DeclareFontSeriesChangeRule {ulec}{ub} {ubec} {bec}
31 \DeclareFontSeriesChangeRule {ulec}{uc} {uluc} {}
32 \DeclareFontSeriesChangeRule {ulec}{ec} {ulec} {}
33 \DeclareFontSeriesChangeRule {ulec}{c} {ulc} {}
34 \DeclareFontSeriesChangeRule {ulec}{sc} {ulsc} {}
35 \DeclareFontSeriesChangeRule {ulec}{sx} {ulsx} {}
36 \DeclareFontSeriesChangeRule {ulec}{x} {ulx} {}
37 \DeclareFontSeriesChangeRule {ulec}{ex} {ulex} {}
38 \DeclareFontSeriesChangeRule {ulec}{ux} {ulux} {}

```

```

39 \DeclareFontSeriesChangeRule {ulc}{ul} {ulc} {}
40 \DeclareFontSeriesChangeRule {ulc}{el} {elc} {}
41 \DeclareFontSeriesChangeRule {ulc}{l} {lc} {}
42 \DeclareFontSeriesChangeRule {ulc}{sl} {slc} {}
43 \DeclareFontSeriesChangeRule {ulc}{sb} {sbc} {bc}
44 \DeclareFontSeriesChangeRule {ulc}{b} {bc} {}
45 \DeclareFontSeriesChangeRule {ulc}{eb} {ebc} {bc}
46 \DeclareFontSeriesChangeRule {ulc}{ub} {ubc} {bc}
47 \DeclareFontSeriesChangeRule {ulc}{uc} {uluc} {}
48 \DeclareFontSeriesChangeRule {ulc}{ec} {ulec} {}
49 \DeclareFontSeriesChangeRule {ulc}{c} {ulc} {}
50 \DeclareFontSeriesChangeRule {ulc}{sc} {ulsc} {}
51 \DeclareFontSeriesChangeRule {ulc}{sx} {ulsx} {}
52 \DeclareFontSeriesChangeRule {ulc}{x} {ulx} {}
53 \DeclareFontSeriesChangeRule {ulc}{ex} {ulex} {}
54 \DeclareFontSeriesChangeRule {ulc}{ux} {ulux} {}

55 \DeclareFontSeriesChangeRule {ulsc}{ul} {ulsc} {}
56 \DeclareFontSeriesChangeRule {ulsc}{el} {elsc} {}
57 \DeclareFontSeriesChangeRule {ulsc}{l} {lsc} {}
58 \DeclareFontSeriesChangeRule {ulsc}{sl} {slsc} {}
59 \DeclareFontSeriesChangeRule {ulsc}{sb} {sbsc} {bsc}
60 \DeclareFontSeriesChangeRule {ulsc}{b} {bsc} {}
61 \DeclareFontSeriesChangeRule {ulsc}{eb} {ebsc} {bsc}
62 \DeclareFontSeriesChangeRule {ulsc}{ub} {ubsc} {bsc}
63 \DeclareFontSeriesChangeRule {ulsc}{uc} {uluc} {}
64 \DeclareFontSeriesChangeRule {ulsc}{ec} {ulec} {}
65 \DeclareFontSeriesChangeRule {ulsc}{c} {ulc} {}
66 \DeclareFontSeriesChangeRule {ulsc}{sc} {ulsc} {}
67 \DeclareFontSeriesChangeRule {ulsc}{sx} {ulsx} {}
68 \DeclareFontSeriesChangeRule {ulsc}{x} {ulx} {}
69 \DeclareFontSeriesChangeRule {ulsc}{ex} {ulex} {}
70 \DeclareFontSeriesChangeRule {ulsc}{ux} {ulux} {}

71 \DeclareFontSeriesChangeRule {ul}{uc} {uluc} {}
72 \DeclareFontSeriesChangeRule {ul}{ec} {ulec} {}
73 \DeclareFontSeriesChangeRule {ul}{c} {ulc} {}
74 \DeclareFontSeriesChangeRule {ul}{sc} {ulsc} {}
75 \DeclareFontSeriesChangeRule {ul}{sx} {ulsx} {}
76 \DeclareFontSeriesChangeRule {ul}{x} {ulx} {}
77 \DeclareFontSeriesChangeRule {ul}{ex} {ulex} {}
78 \DeclareFontSeriesChangeRule {ul}{ux} {ulux} {}
79 \DeclareFontSeriesChangeRule {ul}{sb} {sb} {b}
80 \DeclareFontSeriesChangeRule {ul}{eb} {eb} {b}
81 \DeclareFontSeriesChangeRule {ul}{ub} {ub} {b}

82 \DeclareFontSeriesChangeRule {ulsx}{ul} {ulsx} {}
83 \DeclareFontSeriesChangeRule {ulsx}{el} {elsx} {}
84 \DeclareFontSeriesChangeRule {ulsx}{l} {lsx} {}
85 \DeclareFontSeriesChangeRule {ulsx}{sl} {slsx} {}
86 \DeclareFontSeriesChangeRule {ulsx}{sb} {sbsx} {bsx}
87 \DeclareFontSeriesChangeRule {ulsx}{b} {bsx} {}
88 \DeclareFontSeriesChangeRule {ulsx}{eb} {ebsx} {bsx}
89 \DeclareFontSeriesChangeRule {ulsx}{ub} {ubsx} {bsx}
90 \DeclareFontSeriesChangeRule {ulsx}{uc} {uluc} {}
91 \DeclareFontSeriesChangeRule {ulsx}{ec} {ulec} {}

```



```

92 \DeclareFontSeriesChangeRule {ulsx}{c} {ulc} {}
93 \DeclareFontSeriesChangeRule {ulsx}{sc} {ulsc} {}
94 \DeclareFontSeriesChangeRule {ulsx}{sx} {ulsx} {}
95 \DeclareFontSeriesChangeRule {ulsx}{x} {ulx} {}
96 \DeclareFontSeriesChangeRule {ulsx}{ex} {ulex} {}
97 \DeclareFontSeriesChangeRule {ulsx}{ux} {ulux} {}

98 \DeclareFontSeriesChangeRule {ulx}{ul} {ulx} {}
99 \DeclareFontSeriesChangeRule {ulx}{el} {elx} {}
100 \DeclareFontSeriesChangeRule {ulx}{l} {lx} {}
101 \DeclareFontSeriesChangeRule {ulx}{sl} {slx} {}
102 \DeclareFontSeriesChangeRule {ulx}{sb} {sbx} {bx}
103 \DeclareFontSeriesChangeRule {ulx}{b} {bx} {}
104 \DeclareFontSeriesChangeRule {ulx}{eb} {ebx} {bx}
105 \DeclareFontSeriesChangeRule {ulx}{ub} {ubx} {bx}
106 \DeclareFontSeriesChangeRule {ulx}{uc} {uluc} {}
107 \DeclareFontSeriesChangeRule {ulx}{ec} {ulec} {}
108 \DeclareFontSeriesChangeRule {ulx}{c} {ulc} {}
109 \DeclareFontSeriesChangeRule {ulx}{sc} {ulsc} {}
110 \DeclareFontSeriesChangeRule {ulx}{sx} {ulsx} {}
111 \DeclareFontSeriesChangeRule {ulx}{x} {ulx} {}
112 \DeclareFontSeriesChangeRule {ulx}{ex} {ulex} {}
113 \DeclareFontSeriesChangeRule {ulx}{ux} {ulux} {}

114 \DeclareFontSeriesChangeRule {ulex}{ul} {ulex} {}
115 \DeclareFontSeriesChangeRule {ulex}{el} {elex} {}
116 \DeclareFontSeriesChangeRule {ulex}{l} {lex} {}
117 \DeclareFontSeriesChangeRule {ulex}{sl} {slex} {}
118 \DeclareFontSeriesChangeRule {ulex}{sb} {sbex} {bex}
119 \DeclareFontSeriesChangeRule {ulex}{b} {bex} {}
120 \DeclareFontSeriesChangeRule {ulex}{eb} {ebex} {bex}
121 \DeclareFontSeriesChangeRule {ulex}{ub} {ubex} {bex}
122 \DeclareFontSeriesChangeRule {ulex}{uc} {uluc} {}
123 \DeclareFontSeriesChangeRule {ulex}{ec} {ulec} {}
124 \DeclareFontSeriesChangeRule {ulex}{c} {ulc} {}
125 \DeclareFontSeriesChangeRule {ulex}{sc} {ulsc} {}
126 \DeclareFontSeriesChangeRule {ulex}{sx} {ulsx} {}
127 \DeclareFontSeriesChangeRule {ulex}{x} {ulx} {}
128 \DeclareFontSeriesChangeRule {ulex}{ex} {ulex} {}
129 \DeclareFontSeriesChangeRule {ulex}{ux} {ulux} {}

130 \DeclareFontSeriesChangeRule {ulux}{ul} {ulux} {}
131 \DeclareFontSeriesChangeRule {ulux}{el} {elux} {}
132 \DeclareFontSeriesChangeRule {ulux}{l} {lux} {}
133 \DeclareFontSeriesChangeRule {ulux}{sl} {slux} {}
134 \DeclareFontSeriesChangeRule {ulux}{sb} {sbux} {bux}
135 \DeclareFontSeriesChangeRule {ulux}{b} {bux} {}
136 \DeclareFontSeriesChangeRule {ulux}{eb} {ebux} {bux}
137 \DeclareFontSeriesChangeRule {ulux}{ub} {ubux} {bux}
138 \DeclareFontSeriesChangeRule {ulux}{uc} {uluc} {}
139 \DeclareFontSeriesChangeRule {ulux}{ec} {ulec} {}
140 \DeclareFontSeriesChangeRule {ulux}{c} {ulc} {}
141 \DeclareFontSeriesChangeRule {ulux}{sc} {ulsc} {}
142 \DeclareFontSeriesChangeRule {ulux}{sx} {ulsx} {}
143 \DeclareFontSeriesChangeRule {ulux}{x} {ulx} {}
144 \DeclareFontSeriesChangeRule {ulux}{ex} {ulex} {}

```

```

145 \DeclareFontSeriesChangeRule {ulux}{ux} {ulux} {}
146 \DeclareFontSeriesChangeRule {eluc}{ul} {uluc} {}
147 \DeclareFontSeriesChangeRule {eluc}{el} {eluc} {}
148 \DeclareFontSeriesChangeRule {eluc}{l} {luc} {}
149 \DeclareFontSeriesChangeRule {eluc}{sl} {sluc} {}
150 \DeclareFontSeriesChangeRule {eluc}{sb} {sbuc} {buc}
151 \DeclareFontSeriesChangeRule {eluc}{b} {buc} {}
152 \DeclareFontSeriesChangeRule {eluc}{eb} {ebuc} {buc}
153 \DeclareFontSeriesChangeRule {eluc}{ub} {ubuc} {buc}
154 \DeclareFontSeriesChangeRule {eluc}{uc} {eluc} {}
155 \DeclareFontSeriesChangeRule {eluc}{ec} {elec} {}
156 \DeclareFontSeriesChangeRule {eluc}{c} {elc} {}
157 \DeclareFontSeriesChangeRule {eluc}{sc} {elsc} {}
158 \DeclareFontSeriesChangeRule {eluc}{sx} {elsx} {}
159 \DeclareFontSeriesChangeRule {eluc}{x} {elx} {}
160 \DeclareFontSeriesChangeRule {eluc}{ex} {elcx} {}
161 \DeclareFontSeriesChangeRule {eluc}{ux} {elux} {}

162 \DeclareFontSeriesChangeRule {elec}{ul} {ulec} {}
163 \DeclareFontSeriesChangeRule {elec}{el} {elec} {}
164 \DeclareFontSeriesChangeRule {elec}{l} {lec} {}
165 \DeclareFontSeriesChangeRule {elec}{sl} {slec} {}
166 \DeclareFontSeriesChangeRule {elec}{sb} {sbec} {bec}
167 \DeclareFontSeriesChangeRule {elec}{b} {bec} {}
168 \DeclareFontSeriesChangeRule {elec}{eb} {ebec} {bec}
169 \DeclareFontSeriesChangeRule {elec}{ub} {ubec} {bec}
170 \DeclareFontSeriesChangeRule {elec}{uc} {eluc} {}
171 \DeclareFontSeriesChangeRule {elec}{ec} {elec} {}
172 \DeclareFontSeriesChangeRule {elec}{c} {elc} {}
173 \DeclareFontSeriesChangeRule {elec}{sc} {elsc} {}
174 \DeclareFontSeriesChangeRule {elec}{sx} {elsx} {}
175 \DeclareFontSeriesChangeRule {elec}{x} {elx} {}
176 \DeclareFontSeriesChangeRule {elec}{ex} {elcx} {}
177 \DeclareFontSeriesChangeRule {elec}{ux} {elux} {}

178 \DeclareFontSeriesChangeRule {elc}{ul} {ulc} {}
179 \DeclareFontSeriesChangeRule {elc}{el} {elc} {}
180 \DeclareFontSeriesChangeRule {elc}{l} {lc} {}
181 \DeclareFontSeriesChangeRule {elc}{sl} {slc} {}
182 \DeclareFontSeriesChangeRule {elc}{sb} {sbc} {bc}
183 \DeclareFontSeriesChangeRule {elc}{b} {bc} {}
184 \DeclareFontSeriesChangeRule {elc}{eb} {ebc} {bc}
185 \DeclareFontSeriesChangeRule {elc}{ub} {ubc} {bc}
186 \DeclareFontSeriesChangeRule {elc}{uc} {eluc} {}
187 \DeclareFontSeriesChangeRule {elc}{ec} {elec} {}
188 \DeclareFontSeriesChangeRule {elc}{c} {elc} {}
189 \DeclareFontSeriesChangeRule {elc}{sc} {elsc} {}
190 \DeclareFontSeriesChangeRule {elc}{sx} {elsx} {}
191 \DeclareFontSeriesChangeRule {elc}{x} {elx} {}
192 \DeclareFontSeriesChangeRule {elc}{ex} {elcx} {}
193 \DeclareFontSeriesChangeRule {elc}{ux} {elux} {}

194 \DeclareFontSeriesChangeRule {elsc}{ul} {ulsc} {}
195 \DeclareFontSeriesChangeRule {elsc}{el} {elsc} {}
196 \DeclareFontSeriesChangeRule {elsc}{l} {lsc} {}
197 \DeclareFontSeriesChangeRule {elsc}{sl} {slsc} {}

```

```

198 \DeclareFontSeriesChangeRule {elsc}{sb} {sbsc} {bsc}
199 \DeclareFontSeriesChangeRule {elsc}{b} {bsc} {}
200 \DeclareFontSeriesChangeRule {elsc}{eb} {ebsc} {bsc}
201 \DeclareFontSeriesChangeRule {elsc}{ub} {ubsc} {bsc}
202 \DeclareFontSeriesChangeRule {elsc}{uc} {eluc} {}
203 \DeclareFontSeriesChangeRule {elsc}{ec} {elec} {}
204 \DeclareFontSeriesChangeRule {elsc}{c} {elc} {}
205 \DeclareFontSeriesChangeRule {elsc}{sc} {elsc} {}
206 \DeclareFontSeriesChangeRule {elsc}{sx} {elsx} {}
207 \DeclareFontSeriesChangeRule {elsc}{x} {elx} {}
208 \DeclareFontSeriesChangeRule {elsc}{ex} {elex} {}
209 \DeclareFontSeriesChangeRule {elsc}{ux} {elux} {}

210 \DeclareFontSeriesChangeRule {el}{uc} {eluc} {}
211 \DeclareFontSeriesChangeRule {el}{ec} {elec} {}
212 \DeclareFontSeriesChangeRule {el}{c} {elc} {}
213 \DeclareFontSeriesChangeRule {el}{sc} {elsc} {}
214 \DeclareFontSeriesChangeRule {el}{sx} {elsx} {}
215 \DeclareFontSeriesChangeRule {el}{x} {elx} {}
216 \DeclareFontSeriesChangeRule {el}{ex} {elex} {}
217 \DeclareFontSeriesChangeRule {el}{ux} {elux} {}
218 \DeclareFontSeriesChangeRule {el}{sb} {sb} {b}
219 \DeclareFontSeriesChangeRule {el}{eb} {eb} {b}
220 \DeclareFontSeriesChangeRule {el}{ub} {ub} {b}

221 \DeclareFontSeriesChangeRule {elsx}{ul} {ulsx} {}
222 \DeclareFontSeriesChangeRule {elsx}{el} {elsx} {}
223 \DeclareFontSeriesChangeRule {elsx}{l} {lsx} {}
224 \DeclareFontSeriesChangeRule {elsx}{sl} {slsx} {}
225 \DeclareFontSeriesChangeRule {elsx}{sb} {sbsx} {bsx}
226 \DeclareFontSeriesChangeRule {elsx}{b} {bsx} {}
227 \DeclareFontSeriesChangeRule {elsx}{eb} {ebsx} {bsx}
228 \DeclareFontSeriesChangeRule {elsx}{ub} {ubsx} {bsx}
229 \DeclareFontSeriesChangeRule {elsx}{uc} {eluc} {}
230 \DeclareFontSeriesChangeRule {elsx}{ec} {elec} {}
231 \DeclareFontSeriesChangeRule {elsx}{c} {elc} {}
232 \DeclareFontSeriesChangeRule {elsx}{sc} {elsc} {}
233 \DeclareFontSeriesChangeRule {elsx}{sx} {elsx} {}
234 \DeclareFontSeriesChangeRule {elsx}{x} {elx} {}
235 \DeclareFontSeriesChangeRule {elsx}{ex} {elex} {}
236 \DeclareFontSeriesChangeRule {elsx}{ux} {elux} {}

237 \DeclareFontSeriesChangeRule {elx}{ul} {ulx} {}
238 \DeclareFontSeriesChangeRule {elx}{el} {elx} {}
239 \DeclareFontSeriesChangeRule {elx}{l} {lx} {}
240 \DeclareFontSeriesChangeRule {elx}{sl} {slx} {}
241 \DeclareFontSeriesChangeRule {elx}{sb} {sbx} {bx}
242 \DeclareFontSeriesChangeRule {elx}{b} {bx} {}
243 \DeclareFontSeriesChangeRule {elx}{eb} {ebx} {bx}
244 \DeclareFontSeriesChangeRule {elx}{ub} {ubx} {bx}
245 \DeclareFontSeriesChangeRule {elx}{uc} {eluc} {}
246 \DeclareFontSeriesChangeRule {elx}{ec} {elec} {}
247 \DeclareFontSeriesChangeRule {elx}{c} {elc} {}
248 \DeclareFontSeriesChangeRule {elx}{sc} {elsc} {}
249 \DeclareFontSeriesChangeRule {elx}{sx} {elsx} {}
250 \DeclareFontSeriesChangeRule {elx}{x} {elx} {}

```

```

251 \DeclareFontSeriesChangeRule {elx}{ex} {elex} {}
252 \DeclareFontSeriesChangeRule {elx}{ux} {elux} {}

253 \DeclareFontSeriesChangeRule {elex}{ul} {ulex} {}
254 \DeclareFontSeriesChangeRule {elex}{el} {elex} {}
255 \DeclareFontSeriesChangeRule {elex}{l} {lex} {}
256 \DeclareFontSeriesChangeRule {elex}{sl} {slex} {}
257 \DeclareFontSeriesChangeRule {elex}{sb} {sbex} {bex}
258 \DeclareFontSeriesChangeRule {elex}{b} {bex} {}
259 \DeclareFontSeriesChangeRule {elex}{eb} {ebex} {bex}
260 \DeclareFontSeriesChangeRule {elex}{ub} {ubex} {bex}
261 \DeclareFontSeriesChangeRule {elex}{uc} {eluc} {}
262 \DeclareFontSeriesChangeRule {elex}{ec} {elec} {}
263 \DeclareFontSeriesChangeRule {elex}{c} {elc} {}
264 \DeclareFontSeriesChangeRule {elex}{sc} {elsc} {}
265 \DeclareFontSeriesChangeRule {elex}{sx} {elsx} {}
266 \DeclareFontSeriesChangeRule {elex}{x} {elx} {}
267 \DeclareFontSeriesChangeRule {elex}{ex} {elex} {}
268 \DeclareFontSeriesChangeRule {elex}{ux} {elux} {}

269 \DeclareFontSeriesChangeRule {elux}{ul} {ulux} {}
270 \DeclareFontSeriesChangeRule {elux}{el} {elux} {}
271 \DeclareFontSeriesChangeRule {elux}{l} {lux} {}
272 \DeclareFontSeriesChangeRule {elux}{sl} {slux} {}
273 \DeclareFontSeriesChangeRule {elux}{sb} {sbux} {bux}
274 \DeclareFontSeriesChangeRule {elux}{b} {bux} {}
275 \DeclareFontSeriesChangeRule {elux}{eb} {ebux} {bux}
276 \DeclareFontSeriesChangeRule {elux}{ub} {ubux} {bux}
277 \DeclareFontSeriesChangeRule {elux}{uc} {eluc} {}
278 \DeclareFontSeriesChangeRule {elux}{ec} {elec} {}
279 \DeclareFontSeriesChangeRule {elux}{c} {elc} {}
280 \DeclareFontSeriesChangeRule {elux}{sc} {elsc} {}
281 \DeclareFontSeriesChangeRule {elux}{sx} {elsx} {}
282 \DeclareFontSeriesChangeRule {elux}{x} {elx} {}
283 \DeclareFontSeriesChangeRule {elux}{ex} {elex} {}
284 \DeclareFontSeriesChangeRule {elux}{ux} {elux} {}

285 \DeclareFontSeriesChangeRule {luc}{ul} {uluc} {}
286 \DeclareFontSeriesChangeRule {luc}{el} {eluc} {}
287 \DeclareFontSeriesChangeRule {luc}{l} {luc} {}
288 \DeclareFontSeriesChangeRule {luc}{sl} {sluc} {}
289 \DeclareFontSeriesChangeRule {luc}{sb} {sbuc} {buc}
290 \DeclareFontSeriesChangeRule {luc}{b} {buc} {}
291 \DeclareFontSeriesChangeRule {luc}{eb} {ebuc} {buc}
292 \DeclareFontSeriesChangeRule {luc}{ub} {ubuc} {buc}
293 \DeclareFontSeriesChangeRule {luc}{uc} {luc} {}
294 \DeclareFontSeriesChangeRule {luc}{ec} {lec} {}
295 \DeclareFontSeriesChangeRule {luc}{c} {lc} {}
296 \DeclareFontSeriesChangeRule {luc}{sc} {lsc} {}
297 \DeclareFontSeriesChangeRule {luc}{sx} {lsx} {}
298 \DeclareFontSeriesChangeRule {luc}{x} {lx} {}
299 \DeclareFontSeriesChangeRule {luc}{ex} {lex} {}
300 \DeclareFontSeriesChangeRule {luc}{ux} {lux} {}

301 \DeclareFontSeriesChangeRule {lec}{ul} {ulec} {}
302 \DeclareFontSeriesChangeRule {lec}{el} {elec} {}
303 \DeclareFontSeriesChangeRule {lec}{l} {lec} {}

```

```

304 \DeclareFontSeriesChangeRule {lec}{sl} {sllec} {}
305 \DeclareFontSeriesChangeRule {lec}{sb} {sbec} {bec}
306 \DeclareFontSeriesChangeRule {lec}{b} {bec} {}
307 \DeclareFontSeriesChangeRule {lec}{eb} {ebec} {bec}
308 \DeclareFontSeriesChangeRule {lec}{ub} {ubec} {bec}
309 \DeclareFontSeriesChangeRule {lec}{uc} {luc} {}
310 \DeclareFontSeriesChangeRule {lec}{ec} {lec} {}
311 \DeclareFontSeriesChangeRule {lec}{c} {lc} {}
312 \DeclareFontSeriesChangeRule {lec}{sc} {lsc} {}
313 \DeclareFontSeriesChangeRule {lec}{sx} {lsx} {}
314 \DeclareFontSeriesChangeRule {lec}{x} {lx} {}
315 \DeclareFontSeriesChangeRule {lec}{ex} {lex} {}
316 \DeclareFontSeriesChangeRule {lec}{ux} {lux} {}

317 \DeclareFontSeriesChangeRule {lc}{ul} {ulc} {}
318 \DeclareFontSeriesChangeRule {lc}{el} {elc} {}
319 \DeclareFontSeriesChangeRule {lc}{l} {lc} {}
320 \DeclareFontSeriesChangeRule {lc}{sl} {slc} {}
321 \DeclareFontSeriesChangeRule {lc}{sb} {sbc} {bc}
322 \DeclareFontSeriesChangeRule {lc}{b} {bc} {}
323 \DeclareFontSeriesChangeRule {lc}{eb} {ebc} {bc}
324 \DeclareFontSeriesChangeRule {lc}{ub} {ubc} {bc}
325 \DeclareFontSeriesChangeRule {lc}{uc} {luc} {}
326 \DeclareFontSeriesChangeRule {lc}{ec} {lec} {}
327 \DeclareFontSeriesChangeRule {lc}{c} {lc} {}
328 \DeclareFontSeriesChangeRule {lc}{sc} {lsc} {}
329 \DeclareFontSeriesChangeRule {lc}{sx} {lsx} {}
330 \DeclareFontSeriesChangeRule {lc}{x} {lx} {}
331 \DeclareFontSeriesChangeRule {lc}{ex} {lex} {}
332 \DeclareFontSeriesChangeRule {lc}{ux} {lux} {}

333 \DeclareFontSeriesChangeRule {lsc}{ul} {ulsc} {}
334 \DeclareFontSeriesChangeRule {lsc}{el} {elsc} {}
335 \DeclareFontSeriesChangeRule {lsc}{l} {lsc} {}
336 \DeclareFontSeriesChangeRule {lsc}{sl} {slsc} {}
337 \DeclareFontSeriesChangeRule {lsc}{sb} {sbsc} {bsc}
338 \DeclareFontSeriesChangeRule {lsc}{b} {bsc} {}
339 \DeclareFontSeriesChangeRule {lsc}{eb} {ebsc} {bsc}
340 \DeclareFontSeriesChangeRule {lsc}{ub} {ubsc} {bsc}
341 \DeclareFontSeriesChangeRule {lsc}{uc} {luc} {}
342 \DeclareFontSeriesChangeRule {lsc}{ec} {lec} {}
343 \DeclareFontSeriesChangeRule {lsc}{c} {lc} {}
344 \DeclareFontSeriesChangeRule {lsc}{sc} {lsc} {}
345 \DeclareFontSeriesChangeRule {lsc}{sx} {lsx} {}
346 \DeclareFontSeriesChangeRule {lsc}{x} {lx} {}
347 \DeclareFontSeriesChangeRule {lsc}{ex} {lex} {}
348 \DeclareFontSeriesChangeRule {lsc}{ux} {lux} {}

349 \DeclareFontSeriesChangeRule {l}{uc} {luc} {}
350 \DeclareFontSeriesChangeRule {l}{ec} {lec} {}
351 \DeclareFontSeriesChangeRule {l}{c} {lc} {}
352 \DeclareFontSeriesChangeRule {l}{sc} {lsc} {}
353 \DeclareFontSeriesChangeRule {l}{sx} {lsx} {}
354 \DeclareFontSeriesChangeRule {l}{x} {lx} {}
355 \DeclareFontSeriesChangeRule {l}{ex} {lex} {}
356 \DeclareFontSeriesChangeRule {l}{ux} {lux} {}

```

```

357 \DeclareFontSeriesChangeRule {l}{sb} {sb} {b}
358 \DeclareFontSeriesChangeRule {l}{eb} {eb} {b}
359 \DeclareFontSeriesChangeRule {l}{ub} {ub} {b}

360 \DeclareFontSeriesChangeRule {lsx}{ul} {ulsx} {}
361 \DeclareFontSeriesChangeRule {lsx}{el} {elsx} {}
362 \DeclareFontSeriesChangeRule {lsx}{l} {lsx} {}
363 \DeclareFontSeriesChangeRule {lsx}{sl} {slsx} {}
364 \DeclareFontSeriesChangeRule {lsx}{sb} {sbsx} {bsx}
365 \DeclareFontSeriesChangeRule {lsx}{b} {bsx} {}
366 \DeclareFontSeriesChangeRule {lsx}{eb} {ebsx} {bsx}
367 \DeclareFontSeriesChangeRule {lsx}{ub} {ubsx} {bsx}
368 \DeclareFontSeriesChangeRule {lsx}{uc} {luc} {}
369 \DeclareFontSeriesChangeRule {lsx}{ec} {lec} {}
370 \DeclareFontSeriesChangeRule {lsx}{c} {lc} {}
371 \DeclareFontSeriesChangeRule {lsx}{sc} {lsc} {}
372 \DeclareFontSeriesChangeRule {lsx}{sx} {lsx} {}
373 \DeclareFontSeriesChangeRule {lsx}{x} {lx} {}
374 \DeclareFontSeriesChangeRule {lsx}{ex} {lex} {}
375 \DeclareFontSeriesChangeRule {lsx}{ux} {lux} {}

376 \DeclareFontSeriesChangeRule {lx}{ul} {ulx} {}
377 \DeclareFontSeriesChangeRule {lx}{el} {elx} {}
378 \DeclareFontSeriesChangeRule {lx}{l} {lx} {}
379 \DeclareFontSeriesChangeRule {lx}{sl} {slx} {}
380 \DeclareFontSeriesChangeRule {lx}{sb} {sbx} {bx}
381 \DeclareFontSeriesChangeRule {lx}{b} {bx} {}
382 \DeclareFontSeriesChangeRule {lx}{eb} {ebx} {bx}
383 \DeclareFontSeriesChangeRule {lx}{ub} {ubx} {bx}
384 \DeclareFontSeriesChangeRule {lx}{uc} {luc} {}
385 \DeclareFontSeriesChangeRule {lx}{ec} {lec} {}
386 \DeclareFontSeriesChangeRule {lx}{c} {lc} {}
387 \DeclareFontSeriesChangeRule {lx}{sc} {lsc} {}
388 \DeclareFontSeriesChangeRule {lx}{sx} {lsx} {}
389 \DeclareFontSeriesChangeRule {lx}{x} {lx} {}
390 \DeclareFontSeriesChangeRule {lx}{ex} {lex} {}
391 \DeclareFontSeriesChangeRule {lx}{ux} {lux} {}

392 \DeclareFontSeriesChangeRule {lex}{ul} {ulex} {}
393 \DeclareFontSeriesChangeRule {lex}{el} {lelex} {}
394 \DeclareFontSeriesChangeRule {lex}{l} {lex} {}
395 \DeclareFontSeriesChangeRule {lex}{sl} {slex} {}
396 \DeclareFontSeriesChangeRule {lex}{sb} {sbex} {bex}
397 \DeclareFontSeriesChangeRule {lex}{b} {bex} {}
398 \DeclareFontSeriesChangeRule {lex}{eb} {ebex} {bex}
399 \DeclareFontSeriesChangeRule {lex}{ub} {ubex} {bex}
400 \DeclareFontSeriesChangeRule {lex}{uc} {luc} {}
401 \DeclareFontSeriesChangeRule {lex}{ec} {lec} {}
402 \DeclareFontSeriesChangeRule {lex}{c} {lc} {}
403 \DeclareFontSeriesChangeRule {lex}{sc} {lsc} {}
404 \DeclareFontSeriesChangeRule {lex}{sx} {lsx} {}
405 \DeclareFontSeriesChangeRule {lex}{x} {lx} {}
406 \DeclareFontSeriesChangeRule {lex}{ex} {lex} {}
407 \DeclareFontSeriesChangeRule {lex}{ux} {lux} {}

408 \DeclareFontSeriesChangeRule {lux}{ul} {fulux} {}
409 \DeclareFontSeriesChangeRule {lux}{el} {felux} {}

```

```

410 \DeclareFontSeriesChangeRule {lux}{l} {lux} {}
411 \DeclareFontSeriesChangeRule {lux}{sl} {slux} {}
412 \DeclareFontSeriesChangeRule {lux}{sb} {sbux} {bux}
413 \DeclareFontSeriesChangeRule {lux}{b} {bux} {}
414 \DeclareFontSeriesChangeRule {lux}{eb} {ebux} {bux}
415 \DeclareFontSeriesChangeRule {lux}{ub} {ubux} {bux}
416 \DeclareFontSeriesChangeRule {lux}{uc} {luc} {}
417 \DeclareFontSeriesChangeRule {lux}{ec} {lec} {}
418 \DeclareFontSeriesChangeRule {lux}{c} {lc} {}
419 \DeclareFontSeriesChangeRule {lux}{sc} {lsc} {}
420 \DeclareFontSeriesChangeRule {lux}{sx} {lsx} {}
421 \DeclareFontSeriesChangeRule {lux}{x} {lx} {}
422 \DeclareFontSeriesChangeRule {lux}{ex} {lex} {}
423 \DeclareFontSeriesChangeRule {lux}{ux} {lux} {}

424 \DeclareFontSeriesChangeRule {sluc}{ul} {uluc} {}
425 \DeclareFontSeriesChangeRule {sluc}{el} {eluc} {}
426 \DeclareFontSeriesChangeRule {sluc}{l} {luc} {}
427 \DeclareFontSeriesChangeRule {sluc}{sl} {sluc} {}
428 \DeclareFontSeriesChangeRule {sluc}{sb} {sbuc} {buc}
429 \DeclareFontSeriesChangeRule {sluc}{b} {buc} {}
430 \DeclareFontSeriesChangeRule {sluc}{eb} {ebuc} {buc}
431 \DeclareFontSeriesChangeRule {sluc}{ub} {ubuc} {buc}
432 \DeclareFontSeriesChangeRule {sluc}{uc} {sluc} {}
433 \DeclareFontSeriesChangeRule {sluc}{ec} {slec} {}
434 \DeclareFontSeriesChangeRule {sluc}{c} {slc} {}
435 \DeclareFontSeriesChangeRule {sluc}{sc} {slsc} {}
436 \DeclareFontSeriesChangeRule {sluc}{sx} {slsx} {}
437 \DeclareFontSeriesChangeRule {sluc}{x} {slx} {}
438 \DeclareFontSeriesChangeRule {sluc}{ex} {slex} {}
439 \DeclareFontSeriesChangeRule {sluc}{ux} {slux} {}

440 \DeclareFontSeriesChangeRule {slec}{ul} {ulec} {}
441 \DeclareFontSeriesChangeRule {slec}{el} {elec} {}
442 \DeclareFontSeriesChangeRule {slec}{l} {lec} {}
443 \DeclareFontSeriesChangeRule {slec}{sl} {slec} {}
444 \DeclareFontSeriesChangeRule {slec}{sb} {sbec} {bec}
445 \DeclareFontSeriesChangeRule {slec}{b} {bec} {}
446 \DeclareFontSeriesChangeRule {slec}{eb} {ebec} {bec}
447 \DeclareFontSeriesChangeRule {slec}{ub} {ubec} {bec}
448 \DeclareFontSeriesChangeRule {slec}{uc} {sluc} {}
449 \DeclareFontSeriesChangeRule {slec}{ec} {slec} {}
450 \DeclareFontSeriesChangeRule {slec}{c} {slc} {}
451 \DeclareFontSeriesChangeRule {slec}{sc} {slsc} {}
452 \DeclareFontSeriesChangeRule {slec}{sx} {slsx} {}
453 \DeclareFontSeriesChangeRule {slec}{x} {slx} {}
454 \DeclareFontSeriesChangeRule {slec}{ex} {slex} {}
455 \DeclareFontSeriesChangeRule {slec}{ux} {slux} {}

456 \DeclareFontSeriesChangeRule {slc}{ul} {ulc} {}
457 \DeclareFontSeriesChangeRule {slc}{el} {elc} {}
458 \DeclareFontSeriesChangeRule {slc}{l} {lc} {}
459 \DeclareFontSeriesChangeRule {slc}{sl} {slc} {}
460 \DeclareFontSeriesChangeRule {slc}{sb} {sbc} {bc}
461 \DeclareFontSeriesChangeRule {slc}{b} {bc} {}
462 \DeclareFontSeriesChangeRule {slc}{eb} {ebc} {bc}

```

```

463 \DeclareFontSeriesChangeRule {slc}{ub} {ubc} {bc}
464 \DeclareFontSeriesChangeRule {slc}{uc} {sluc} {}
465 \DeclareFontSeriesChangeRule {slc}{ec} {slec} {}
466 \DeclareFontSeriesChangeRule {slc}{c} {slc} {}
467 \DeclareFontSeriesChangeRule {slc}{sc} {slsc} {}
468 \DeclareFontSeriesChangeRule {slc}{sx} {slsx} {}
469 \DeclareFontSeriesChangeRule {slc}{x} {slx} {}
470 \DeclareFontSeriesChangeRule {slc}{ex} {slex} {}
471 \DeclareFontSeriesChangeRule {slc}{ux} {slux} {}

472 \DeclareFontSeriesChangeRule {slsc}{ul} {ulsc} {}
473 \DeclareFontSeriesChangeRule {slsc}{el} {elsc} {}
474 \DeclareFontSeriesChangeRule {slsc}{l} {lsc} {}
475 \DeclareFontSeriesChangeRule {slsc}{sl} {slsc} {}
476 \DeclareFontSeriesChangeRule {slsc}{sb} {sbsc} {bsc}
477 \DeclareFontSeriesChangeRule {slsc}{b} {bsc} {}
478 \DeclareFontSeriesChangeRule {slsc}{eb} {ebsc} {bsc}
479 \DeclareFontSeriesChangeRule {slsc}{ub} {ubsc} {bsc}
480 \DeclareFontSeriesChangeRule {slsc}{uc} {sluc} {}
481 \DeclareFontSeriesChangeRule {slsc}{ec} {slec} {}
482 \DeclareFontSeriesChangeRule {slsc}{c} {slc} {}
483 \DeclareFontSeriesChangeRule {slsc}{sc} {slsc} {}
484 \DeclareFontSeriesChangeRule {slsc}{sx} {slsx} {}
485 \DeclareFontSeriesChangeRule {slsc}{x} {slx} {}
486 \DeclareFontSeriesChangeRule {slsc}{ex} {slex} {}
487 \DeclareFontSeriesChangeRule {slsc}{ux} {slux} {}

488 \DeclareFontSeriesChangeRule {sl}{uc} {sluc} {}
489 \DeclareFontSeriesChangeRule {sl}{ec} {slec} {}
490 \DeclareFontSeriesChangeRule {sl}{c} {slc} {}
491 \DeclareFontSeriesChangeRule {sl}{sc} {slsc} {}
492 \DeclareFontSeriesChangeRule {sl}{sx} {slsx} {}
493 \DeclareFontSeriesChangeRule {sl}{x} {slx} {}
494 \DeclareFontSeriesChangeRule {sl}{ex} {slex} {}
495 \DeclareFontSeriesChangeRule {sl}{ux} {slux} {}
496 \DeclareFontSeriesChangeRule {sl}{sb} {sb} {b}
497 \DeclareFontSeriesChangeRule {sl}{eb} {eb} {b}
498 \DeclareFontSeriesChangeRule {sl}{ub} {ub} {b}

499 \DeclareFontSeriesChangeRule {slsx}{ul} {ulsx} {}
500 \DeclareFontSeriesChangeRule {slsx}{el} {elsx} {}
501 \DeclareFontSeriesChangeRule {slsx}{l} {lsx} {}
502 \DeclareFontSeriesChangeRule {slsx}{sl} {slsx} {}
503 \DeclareFontSeriesChangeRule {slsx}{sb} {sbsx} {bsx}
504 \DeclareFontSeriesChangeRule {slsx}{b} {bsx} {}
505 \DeclareFontSeriesChangeRule {slsx}{eb} {ebsx} {bsx}
506 \DeclareFontSeriesChangeRule {slsx}{ub} {ubsx} {bsx}
507 \DeclareFontSeriesChangeRule {slsx}{uc} {sluc} {}
508 \DeclareFontSeriesChangeRule {slsx}{ec} {slec} {}
509 \DeclareFontSeriesChangeRule {slsx}{c} {slc} {}
510 \DeclareFontSeriesChangeRule {slsx}{sc} {slsc} {}
511 \DeclareFontSeriesChangeRule {slsx}{sx} {slsx} {}
512 \DeclareFontSeriesChangeRule {slsx}{x} {slx} {}
513 \DeclareFontSeriesChangeRule {slsx}{ex} {slex} {}
514 \DeclareFontSeriesChangeRule {slsx}{ux} {slux} {}
515 \DeclareFontSeriesChangeRule {slx}{ul} {ulx} {}

```



```

516 \DeclareFontSeriesChangeRule {slx}{el} {elx} {}
517 \DeclareFontSeriesChangeRule {slx}{l} {lx} {}
518 \DeclareFontSeriesChangeRule {slx}{sl} {slx} {}
519 \DeclareFontSeriesChangeRule {slx}{sb} {sbx} {bx}
520 \DeclareFontSeriesChangeRule {slx}{b} {bx} {}
521 \DeclareFontSeriesChangeRule {slx}{eb} {ebx} {bx}
522 \DeclareFontSeriesChangeRule {slx}{ub} {ubx} {bx}
523 \DeclareFontSeriesChangeRule {slx}{uc} {sluc} {}
524 \DeclareFontSeriesChangeRule {slx}{ec} {slec} {}
525 \DeclareFontSeriesChangeRule {slx}{c} {slc} {}
526 \DeclareFontSeriesChangeRule {slx}{sc} {slsc} {}
527 \DeclareFontSeriesChangeRule {slx}{sx} {slsx} {}
528 \DeclareFontSeriesChangeRule {slx}{x} {slx} {}
529 \DeclareFontSeriesChangeRule {slx}{ex} {slex} {}
530 \DeclareFontSeriesChangeRule {slx}{ux} {slux} {}

531 \DeclareFontSeriesChangeRule {slex}{ul} {ulex} {}
532 \DeclareFontSeriesChangeRule {slex}{el} {ellex} {}
533 \DeclareFontSeriesChangeRule {slex}{l} {llex} {}
534 \DeclareFontSeriesChangeRule {slex}{sl} {slex} {}
535 \DeclareFontSeriesChangeRule {slex}{sb} {sbex} {bex}
536 \DeclareFontSeriesChangeRule {slex}{b} {bex} {}
537 \DeclareFontSeriesChangeRule {slex}{eb} {ebex} {bex}
538 \DeclareFontSeriesChangeRule {slex}{ub} {ubex} {bex}
539 \DeclareFontSeriesChangeRule {slex}{uc} {sluc} {}
540 \DeclareFontSeriesChangeRule {slex}{ec} {slec} {}
541 \DeclareFontSeriesChangeRule {slex}{c} {slc} {}
542 \DeclareFontSeriesChangeRule {slex}{sc} {slsc} {}
543 \DeclareFontSeriesChangeRule {slex}{sx} {slsx} {}
544 \DeclareFontSeriesChangeRule {slex}{x} {slx} {}
545 \DeclareFontSeriesChangeRule {slex}{ex} {slex} {}
546 \DeclareFontSeriesChangeRule {slex}{ux} {slux} {}

547 \DeclareFontSeriesChangeRule {slux}{ul} {ulux} {}
548 \DeclareFontSeriesChangeRule {slux}{el} {elux} {}
549 \DeclareFontSeriesChangeRule {slux}{l} {lux} {}
550 \DeclareFontSeriesChangeRule {slux}{sl} {slux} {}
551 \DeclareFontSeriesChangeRule {slux}{sb} {sbux} {bux}
552 \DeclareFontSeriesChangeRule {slux}{b} {bux} {}
553 \DeclareFontSeriesChangeRule {slux}{eb} {ebux} {bux}
554 \DeclareFontSeriesChangeRule {slux}{ub} {ubux} {bux}
555 \DeclareFontSeriesChangeRule {slux}{uc} {sluc} {}
556 \DeclareFontSeriesChangeRule {slux}{ec} {slec} {}
557 \DeclareFontSeriesChangeRule {slux}{c} {slc} {}
558 \DeclareFontSeriesChangeRule {slux}{sc} {slsc} {}
559 \DeclareFontSeriesChangeRule {slux}{sx} {slsx} {}
560 \DeclareFontSeriesChangeRule {slux}{x} {slx} {}
561 \DeclareFontSeriesChangeRule {slux}{ex} {slex} {}
562 \DeclareFontSeriesChangeRule {slux}{ux} {slux} {}

563 \DeclareFontSeriesChangeRule {uc}{ul} {uluc} {}
564 \DeclareFontSeriesChangeRule {uc}{el} {eluc} {}
565 \DeclareFontSeriesChangeRule {uc}{l} {luc} {}
566 \DeclareFontSeriesChangeRule {uc}{sl} {sluc} {}
567 \DeclareFontSeriesChangeRule {uc}{sb} {sbuc} {buc}
568 \DeclareFontSeriesChangeRule {uc}{b} {buc} {}

```

```

569 \DeclareFontSeriesChangeRule {uc}{eb} {ebuc} {buc}
570 \DeclareFontSeriesChangeRule {uc}{ub} {ubuc} {buc}

571 \DeclareFontSeriesChangeRule {ec}{ul} {ulec} {}
572 \DeclareFontSeriesChangeRule {ec}{el} {elec} {}
573 \DeclareFontSeriesChangeRule {ec}{l} {lec} {}
574 \DeclareFontSeriesChangeRule {ec}{sl} {slec} {}
575 \DeclareFontSeriesChangeRule {ec}{sb} {sbec} {bec}
576 \DeclareFontSeriesChangeRule {ec}{b} {bec} {}
577 \DeclareFontSeriesChangeRule {ec}{eb} {ebec} {bec}
578 \DeclareFontSeriesChangeRule {ec}{ub} {ubec} {bec}

```

There are a number of font families that implement condensed series, but often only **c** and **bc**. Therefore, if we see a weight or width change request that can't be fulfilled we try to stay with **c** or **bc**.

```

579 \DeclareFontSeriesChangeRule {c}{ul} {ulc} {c}
580 \DeclareFontSeriesChangeRule {c}{el} {elc} {c}
581 \DeclareFontSeriesChangeRule {c}{l} {lc} {c}
582 \DeclareFontSeriesChangeRule {c}{sl} {slc} {c}
583 \DeclareFontSeriesChangeRule {c}{sb} {sbc} {bc}
584 \DeclareFontSeriesChangeRule {c}{b} {bc} {}
585 \DeclareFontSeriesChangeRule {c}{eb} {ebc} {bc}
586 \DeclareFontSeriesChangeRule {c}{ub} {ubc} {bc}
587 \DeclareFontSeriesChangeRule {c}{uc} {uc} {c}
588 \DeclareFontSeriesChangeRule {c}{ec} {ec} {c}
589 \DeclareFontSeriesChangeRule {c}{sc} {sc} {c}

590 \DeclareFontSeriesChangeRule {sc}{ul} {ulsc} {}
591 \DeclareFontSeriesChangeRule {sc}{el} {elsc} {}
592 \DeclareFontSeriesChangeRule {sc}{l} {lsc} {}
593 \DeclareFontSeriesChangeRule {sc}{sl} {slsc} {}
594 \DeclareFontSeriesChangeRule {sc}{sb} {sbsc} {bsc}
595 \DeclareFontSeriesChangeRule {sc}{b} {bsc} {}
596 \DeclareFontSeriesChangeRule {sc}{eb} {ebsc} {bsc}
597 \DeclareFontSeriesChangeRule {sc}{ub} {ubsc} {bsc}

598 \DeclareFontSeriesChangeRule {m}{sb} {sb} {b}
599 \DeclareFontSeriesChangeRule {m}{eb} {eb} {b}
600 \DeclareFontSeriesChangeRule {m}{ub} {ub} {b}

```

This special rule normally does nothing since nearly every font implements the **b** bold series. The exception are Computer Modern and Latin Modern and fonts based on them. They usually only have **bx**, but then they normally provide an `.fd` file declaration mapping from **b** to **bx** and thus pretend that **b** exists. But in case any of them does not, we offer an alternative result and switch to **bx** if **b** can't be found.

```

601 \DeclareFontSeriesChangeRule {m}{b} {b} {bx}

602 \DeclareFontSeriesChangeRule {sx}{ul} {ulsx} {}
603 \DeclareFontSeriesChangeRule {sx}{el} {elsx} {}
604 \DeclareFontSeriesChangeRule {sx}{l} {lsx} {}
605 \DeclareFontSeriesChangeRule {sx}{sl} {slsx} {}
606 \DeclareFontSeriesChangeRule {sx}{sb} {sbsx} {bsx}
607 \DeclareFontSeriesChangeRule {sx}{b} {bsx} {}
608 \DeclareFontSeriesChangeRule {sx}{eb} {ebsx} {bsx}
609 \DeclareFontSeriesChangeRule {sx}{ub} {ubsx} {bsx}

```

```

610 \DeclareFontSeriesChangeRule {x}{ul}    {ulx} {}
611 \DeclareFontSeriesChangeRule {x}{el}    {elx} {}
612 \DeclareFontSeriesChangeRule {x}{l}     {lx} {}
613 \DeclareFontSeriesChangeRule {x}{sl}    {slx} {}
614 \DeclareFontSeriesChangeRule {x}{sb}    {sbx} {bx}
615 \DeclareFontSeriesChangeRule {x}{b}     {bx} {}
616 \DeclareFontSeriesChangeRule {x}{eb}    {ebx} {bx}
617 \DeclareFontSeriesChangeRule {x}{ub}    {ubx} {bx}

618 \DeclareFontSeriesChangeRule {ex}{ul}   {ulex} {}
619 \DeclareFontSeriesChangeRule {ex}{el}   {elex} {}
620 \DeclareFontSeriesChangeRule {ex}{l}    {lex} {}
621 \DeclareFontSeriesChangeRule {ex}{sl}   {slex} {}
622 \DeclareFontSeriesChangeRule {ex}{sb}   {sbex} {bex}
623 \DeclareFontSeriesChangeRule {ex}{b}    {bex} {}
624 \DeclareFontSeriesChangeRule {ex}{eb}   {ebex} {bex}
625 \DeclareFontSeriesChangeRule {ex}{ub}   {ubex} {bex}

626 \DeclareFontSeriesChangeRule {ux}{ul}   {ulux} {}
627 \DeclareFontSeriesChangeRule {ux}{el}   {elux} {}
628 \DeclareFontSeriesChangeRule {ux}{l}    {lux} {}
629 \DeclareFontSeriesChangeRule {ux}{sl}   {slux} {}
630 \DeclareFontSeriesChangeRule {ux}{sb}   {sbux} {bux}
631 \DeclareFontSeriesChangeRule {ux}{b}    {bux} {}
632 \DeclareFontSeriesChangeRule {ux}{eb}   {ebux} {bux}
633 \DeclareFontSeriesChangeRule {ux}{ub}   {ubux} {bux}

634 \DeclareFontSeriesChangeRule {sbuc}{ul} {uluc} {}
635 \DeclareFontSeriesChangeRule {sbuc}{el} {eluc} {}
636 \DeclareFontSeriesChangeRule {sbuc}{l}  {luc} {}
637 \DeclareFontSeriesChangeRule {sbuc}{sl} {sluc} {}
638 \DeclareFontSeriesChangeRule {sbuc}{sb} {sbuc} {}
639 \DeclareFontSeriesChangeRule {sbuc}{b}  {buc} {}
640 \DeclareFontSeriesChangeRule {sbuc}{eb} {ebuc} {buc}
641 \DeclareFontSeriesChangeRule {sbuc}{ub} {ubuc} {buc}
642 \DeclareFontSeriesChangeRule {sbuc}{uc} {sbuc} {}
643 \DeclareFontSeriesChangeRule {sbuc}{ec} {sbec} {}
644 \DeclareFontSeriesChangeRule {sbuc}{c}  {sbc} {}
645 \DeclareFontSeriesChangeRule {sbuc}{sc} {sbsc} {}
646 \DeclareFontSeriesChangeRule {sbuc}{sx} {sbsx} {}
647 \DeclareFontSeriesChangeRule {sbuc}{x}  {sbx} {}
648 \DeclareFontSeriesChangeRule {sbuc}{ex} {sbex} {}
649 \DeclareFontSeriesChangeRule {sbuc}{ux} {sbux} {}

650 \DeclareFontSeriesChangeRule {sbec}{ul} {ulec} {}
651 \DeclareFontSeriesChangeRule {sbec}{el} {elec} {}
652 \DeclareFontSeriesChangeRule {sbec}{l}  {lec} {}
653 \DeclareFontSeriesChangeRule {sbec}{sl} {slec} {}
654 \DeclareFontSeriesChangeRule {sbec}{sb} {sbec} {}
655 \DeclareFontSeriesChangeRule {sbec}{b}  {bec} {}
656 \DeclareFontSeriesChangeRule {sbec}{eb} {ebec} {bec}
657 \DeclareFontSeriesChangeRule {sbec}{ub} {ubec} {bec}
658 \DeclareFontSeriesChangeRule {sbec}{uc} {sbuc} {}
659 \DeclareFontSeriesChangeRule {sbec}{ec} {sbec} {}
660 \DeclareFontSeriesChangeRule {sbec}{c}  {sbc} {}
661 \DeclareFontSeriesChangeRule {sbec}{sc} {sbsc} {}
662 \DeclareFontSeriesChangeRule {sbec}{sx} {sbsx} {}

```

```

663 \DeclareFontSeriesChangeRule {sbec}{x} {sbx} {}
664 \DeclareFontSeriesChangeRule {sbec}{ex} {sbex} {}
665 \DeclareFontSeriesChangeRule {sbec}{ux} {sbux} {}

666 \DeclareFontSeriesChangeRule {sbc}{ul} {ulc} {}
667 \DeclareFontSeriesChangeRule {sbc}{el} {elc} {}
668 \DeclareFontSeriesChangeRule {sbc}{l} {lc} {}
669 \DeclareFontSeriesChangeRule {sbc}{sl} {slc} {}
670 \DeclareFontSeriesChangeRule {sbc}{sb} {sbc} {}
671 \DeclareFontSeriesChangeRule {sbc}{b} {bc} {}
672 \DeclareFontSeriesChangeRule {sbc}{eb} {ebc} {bc}
673 \DeclareFontSeriesChangeRule {sbc}{ub} {ubc} {bc}
674 \DeclareFontSeriesChangeRule {sbc}{uc} {sbuc} {}
675 \DeclareFontSeriesChangeRule {sbc}{ec} {sbec} {}
676 \DeclareFontSeriesChangeRule {sbc}{c} {sbc} {}
677 \DeclareFontSeriesChangeRule {sbc}{sc} {sbsc} {}
678 \DeclareFontSeriesChangeRule {sbc}{sx} {sbsx} {}
679 \DeclareFontSeriesChangeRule {sbc}{x} {sbx} {}
680 \DeclareFontSeriesChangeRule {sbc}{ex} {sbex} {}
681 \DeclareFontSeriesChangeRule {sbc}{ux} {sbux} {}

682 \DeclareFontSeriesChangeRule {sbsc}{ul} {ulsc} {}
683 \DeclareFontSeriesChangeRule {sbsc}{el} {elsc} {}
684 \DeclareFontSeriesChangeRule {sbsc}{l} {lsc} {}
685 \DeclareFontSeriesChangeRule {sbsc}{sl} {slsc} {}
686 \DeclareFontSeriesChangeRule {sbsc}{sb} {sbsc} {}
687 \DeclareFontSeriesChangeRule {sbsc}{b} {bsc} {}
688 \DeclareFontSeriesChangeRule {sbsc}{eb} {ebsc} {bsc}
689 \DeclareFontSeriesChangeRule {sbsc}{ub} {ubsc} {bsc}
690 \DeclareFontSeriesChangeRule {sbsc}{uc} {sbuc} {}
691 \DeclareFontSeriesChangeRule {sbsc}{ec} {sbec} {}
692 \DeclareFontSeriesChangeRule {sbsc}{c} {sbc} {}
693 \DeclareFontSeriesChangeRule {sbsc}{sc} {sbsc} {}
694 \DeclareFontSeriesChangeRule {sbsc}{sx} {sbsx} {}
695 \DeclareFontSeriesChangeRule {sbsc}{x} {sbx} {}
696 \DeclareFontSeriesChangeRule {sbsc}{ex} {sbex} {}
697 \DeclareFontSeriesChangeRule {sbsc}{ux} {sbux} {}

698 \DeclareFontSeriesChangeRule {sb}{uc} {sbuc} {}
699 \DeclareFontSeriesChangeRule {sb}{ec} {sbec} {}
700 \DeclareFontSeriesChangeRule {sb}{c} {sbc} {}
701 \DeclareFontSeriesChangeRule {sb}{sc} {sbsc} {}
702 \DeclareFontSeriesChangeRule {sb}{sx} {sbsx} {}
703 \DeclareFontSeriesChangeRule {sb}{x} {sbx} {}
704 \DeclareFontSeriesChangeRule {sb}{ex} {sbex} {}
705 \DeclareFontSeriesChangeRule {sb}{ux} {sbux} {}
706 \DeclareFontSeriesChangeRule {sb}{eb} {eb} {b}
707 \DeclareFontSeriesChangeRule {sb}{ub} {ub} {b}

708 \DeclareFontSeriesChangeRule {sbsx}{ul} {ulsx} {}
709 \DeclareFontSeriesChangeRule {sbsx}{el} {elsx} {}
710 \DeclareFontSeriesChangeRule {sbsx}{l} {lsx} {}
711 \DeclareFontSeriesChangeRule {sbsx}{sl} {slsx} {}
712 \DeclareFontSeriesChangeRule {sbsx}{sb} {sbsx} {}
713 \DeclareFontSeriesChangeRule {sbsx}{b} {bsx} {}
714 \DeclareFontSeriesChangeRule {sbsx}{eb} {ebsx} {bsx}
715 \DeclareFontSeriesChangeRule {sbsx}{ub} {ubsx} {bsx}

```

```

716 \DeclareFontSeriesChangeRule {sbsx}{uc} {sbuc} {}
717 \DeclareFontSeriesChangeRule {sbsx}{ec} {sbec} {}
718 \DeclareFontSeriesChangeRule {sbsx}{c} {sbc} {}
719 \DeclareFontSeriesChangeRule {sbsx}{sc} {sbsc} {}
720 \DeclareFontSeriesChangeRule {sbsx}{sx} {sbsx} {}
721 \DeclareFontSeriesChangeRule {sbsx}{x} {sbx} {}
722 \DeclareFontSeriesChangeRule {sbsx}{ex} {sbex} {}
723 \DeclareFontSeriesChangeRule {sbsx}{ux} {sbux} {}

724 \DeclareFontSeriesChangeRule {sbx}{ul} {ulx} {}
725 \DeclareFontSeriesChangeRule {sbx}{el} {elx} {}
726 \DeclareFontSeriesChangeRule {sbx}{l} {lx} {}
727 \DeclareFontSeriesChangeRule {sbx}{sl} {slx} {}
728 \DeclareFontSeriesChangeRule {sbx}{sb} {sbx} {}
729 \DeclareFontSeriesChangeRule {sbx}{b} {bx} {}
730 \DeclareFontSeriesChangeRule {sbx}{eb} {ebx} {bx}
731 \DeclareFontSeriesChangeRule {sbx}{ub} {ubx} {bx}
732 \DeclareFontSeriesChangeRule {sbx}{uc} {sbuc} {}
733 \DeclareFontSeriesChangeRule {sbx}{ec} {sbec} {}
734 \DeclareFontSeriesChangeRule {sbx}{c} {sbc} {}
735 \DeclareFontSeriesChangeRule {sbx}{sc} {sbsc} {}
736 \DeclareFontSeriesChangeRule {sbx}{sx} {sbsx} {}
737 \DeclareFontSeriesChangeRule {sbx}{x} {sbx} {}
738 \DeclareFontSeriesChangeRule {sbx}{ex} {sbex} {}
739 \DeclareFontSeriesChangeRule {sbx}{ux} {sbux} {}

740 \DeclareFontSeriesChangeRule {sbex}{ul} {ulex} {}
741 \DeclareFontSeriesChangeRule {sbex}{el} {elex} {}
742 \DeclareFontSeriesChangeRule {sbex}{l} {lex} {}
743 \DeclareFontSeriesChangeRule {sbex}{sl} {slex} {}
744 \DeclareFontSeriesChangeRule {sbex}{sb} {sbex} {}
745 \DeclareFontSeriesChangeRule {sbex}{b} {bex} {}
746 \DeclareFontSeriesChangeRule {sbex}{eb} {ebex} {bex}
747 \DeclareFontSeriesChangeRule {sbex}{ub} {ubex} {bex}
748 \DeclareFontSeriesChangeRule {sbex}{uc} {sbuc} {}
749 \DeclareFontSeriesChangeRule {sbex}{ec} {sbec} {}
750 \DeclareFontSeriesChangeRule {sbex}{c} {sbc} {}
751 \DeclareFontSeriesChangeRule {sbex}{sc} {sbsc} {}
752 \DeclareFontSeriesChangeRule {sbex}{sx} {sbsx} {}
753 \DeclareFontSeriesChangeRule {sbex}{x} {sbx} {}
754 \DeclareFontSeriesChangeRule {sbex}{ex} {sbex} {}
755 \DeclareFontSeriesChangeRule {sbex}{ux} {sbux} {}

756 \DeclareFontSeriesChangeRule {sbux}{ul} {ulux} {}
757 \DeclareFontSeriesChangeRule {sbux}{el} {elux} {}
758 \DeclareFontSeriesChangeRule {sbux}{l} {lux} {}
759 \DeclareFontSeriesChangeRule {sbux}{sl} {slux} {}
760 \DeclareFontSeriesChangeRule {sbux}{sb} {sbux} {}
761 \DeclareFontSeriesChangeRule {sbux}{b} {bux} {}
762 \DeclareFontSeriesChangeRule {sbux}{eb} {ebux} {bux}
763 \DeclareFontSeriesChangeRule {sbux}{ub} {ubux} {bux}
764 \DeclareFontSeriesChangeRule {sbux}{uc} {sbuc} {}
765 \DeclareFontSeriesChangeRule {sbux}{ec} {sbec} {}
766 \DeclareFontSeriesChangeRule {sbux}{c} {sbc} {}
767 \DeclareFontSeriesChangeRule {sbux}{sc} {sbsc} {}
768 \DeclareFontSeriesChangeRule {sbux}{sx} {sbsx} {}

```

769 \DeclareFontSeriesChangeRule {sbux}{x} {sbx} {}
770 \DeclareFontSeriesChangeRule {sbux}{ex} {sbex} {}
771 \DeclareFontSeriesChangeRule {sbux}{ux} {sbux} {}
772 \DeclareFontSeriesChangeRule {buc}{ul} {uluc} {}
773 \DeclareFontSeriesChangeRule {buc}{el} {eluc} {}
774 \DeclareFontSeriesChangeRule {buc}{l} {luc} {}
775 \DeclareFontSeriesChangeRule {buc}{sl} {sluc} {}
776 \DeclareFontSeriesChangeRule {buc}{sb} {sbuc} {buc}
777 \DeclareFontSeriesChangeRule {buc}{b} {buc} {}
778 \DeclareFontSeriesChangeRule {buc}{eb} {ebuc} {buc}
779 \DeclareFontSeriesChangeRule {buc}{ub} {ubuc} {buc}
780 \DeclareFontSeriesChangeRule {buc}{uc} {buc} {}
781 \DeclareFontSeriesChangeRule {buc}{ec} {bec} {}
782 \DeclareFontSeriesChangeRule {buc}{c} {bc} {}
783 \DeclareFontSeriesChangeRule {buc}{sc} {bsc} {}
784 \DeclareFontSeriesChangeRule {buc}{sx} {bsx} {}
785 \DeclareFontSeriesChangeRule {buc}{x} {bx} {}
786 \DeclareFontSeriesChangeRule {buc}{ex} {bex} {}
787 \DeclareFontSeriesChangeRule {buc}{ux} {bux} {}
788 \DeclareFontSeriesChangeRule {bec}{ul} {ulec} {}
789 \DeclareFontSeriesChangeRule {bec}{el} {elec} {}
790 \DeclareFontSeriesChangeRule {bec}{l} {lec} {}
791 \DeclareFontSeriesChangeRule {bec}{sl} {slec} {}
792 \DeclareFontSeriesChangeRule {bec}{sb} {sbec} {bec}
793 \DeclareFontSeriesChangeRule {bec}{b} {bec} {}
794 \DeclareFontSeriesChangeRule {bec}{eb} {ebec} {bec}
795 \DeclareFontSeriesChangeRule {bec}{ub} {ubec} {bec}
796 \DeclareFontSeriesChangeRule {bec}{uc} {buc} {}
797 \DeclareFontSeriesChangeRule {bec}{ec} {bec} {}
798 \DeclareFontSeriesChangeRule {bec}{c} {bc} {}
799 \DeclareFontSeriesChangeRule {bec}{sc} {bsc} {}
800 \DeclareFontSeriesChangeRule {bec}{sx} {bsx} {}
801 \DeclareFontSeriesChangeRule {bec}{x} {bx} {}
802 \DeclareFontSeriesChangeRule {bec}{ex} {bex} {}
803 \DeclareFontSeriesChangeRule {bec}{ux} {bux} {}
804 \DeclareFontSeriesChangeRule {bc}{ul} {ulc} {}
805 \DeclareFontSeriesChangeRule {bc}{el} {elc} {}
806 \DeclareFontSeriesChangeRule {bc}{l} {lc} {}
807 \DeclareFontSeriesChangeRule {bc}{sl} {slc} {}
808 \DeclareFontSeriesChangeRule {bc}{sb} {sbc} {bc}
809 \DeclareFontSeriesChangeRule {bc}{b} {bc} {}
810 \DeclareFontSeriesChangeRule {bc}{eb} {ebc} {bc}
811 \DeclareFontSeriesChangeRule {bc}{ub} {ubc} {bc}
812 \DeclareFontSeriesChangeRule {bc}{uc} {buc} {}
813 \DeclareFontSeriesChangeRule {bc}{ec} {bec} {}
814 \DeclareFontSeriesChangeRule {bc}{c} {bc} {}
815 \DeclareFontSeriesChangeRule {bc}{sc} {bsc} {}
816 \DeclareFontSeriesChangeRule {bc}{sx} {bsx} {}
817 \DeclareFontSeriesChangeRule {bc}{x} {bx} {}
818 \DeclareFontSeriesChangeRule {bc}{ex} {bex} {}
819 \DeclareFontSeriesChangeRule {bc}{ux} {bux} {}
820 \DeclareFontSeriesChangeRule {bsc}{ul} {ulsc} {}
821 \DeclareFontSeriesChangeRule {bsc}{el} {elsc} {}

```

822 \DeclareFontSeriesChangeRule {bsc}{l} {lsc} {}
823 \DeclareFontSeriesChangeRule {bsc}{sl} {slsc} {}
824 \DeclareFontSeriesChangeRule {bsc}{sb} {sbsc} {bsc}
825 \DeclareFontSeriesChangeRule {bsc}{b} {bsc} {}
826 \DeclareFontSeriesChangeRule {bsc}{eb} {ebsc} {bsc}
827 \DeclareFontSeriesChangeRule {bsc}{ub} {ubsc} {bsc}
828 \DeclareFontSeriesChangeRule {bsc}{uc} {buc} {}
829 \DeclareFontSeriesChangeRule {bsc}{ec} {bec} {}
830 \DeclareFontSeriesChangeRule {bsc}{c} {bc} {}
831 \DeclareFontSeriesChangeRule {bsc}{sc} {bsc} {}
832 \DeclareFontSeriesChangeRule {bsc}{sx} {bsx} {}
833 \DeclareFontSeriesChangeRule {bsc}{x} {bx} {}
834 \DeclareFontSeriesChangeRule {bsc}{ex} {bex} {}
835 \DeclareFontSeriesChangeRule {bsc}{ux} {bux} {}

```

If we are in **b** and a width change is requested that leads to a missing font face we stay in **b** because then the font family probably doesn't implement width changes at all.

```

836 \DeclareFontSeriesChangeRule {b}{uc} {buc} {b}
837 \DeclareFontSeriesChangeRule {b}{ec} {bec} {b}
838 \DeclareFontSeriesChangeRule {b}{c} {bc} {b}
839 \DeclareFontSeriesChangeRule {b}{sc} {bsc} {b}
840 \DeclareFontSeriesChangeRule {b}{sx} {bsx} {b}
841 \DeclareFontSeriesChangeRule {b}{x} {bx} {b}
842 \DeclareFontSeriesChangeRule {b}{ex} {bex} {b}
843 \DeclareFontSeriesChangeRule {b}{ux} {bux} {b}
844 \DeclareFontSeriesChangeRule {b}{sb} {sb} {b}
845 \DeclareFontSeriesChangeRule {b}{eb} {eb} {b}
846 \DeclareFontSeriesChangeRule {b}{ub} {ub} {b}

847 \DeclareFontSeriesChangeRule {bsx}{ul} {ulsx} {}
848 \DeclareFontSeriesChangeRule {bsx}{el} {elsx} {}
849 \DeclareFontSeriesChangeRule {bsx}{l} {lsx} {}
850 \DeclareFontSeriesChangeRule {bsx}{sl} {slsx} {}
851 \DeclareFontSeriesChangeRule {bsx}{sb} {sbsx} {bsx}
852 \DeclareFontSeriesChangeRule {bsx}{b} {bsx} {}
853 \DeclareFontSeriesChangeRule {bsx}{eb} {ebsx} {bsx}
854 \DeclareFontSeriesChangeRule {bsx}{ub} {ubsx} {bsx}
855 \DeclareFontSeriesChangeRule {bsx}{uc} {buc} {}
856 \DeclareFontSeriesChangeRule {bsx}{ec} {bec} {}
857 \DeclareFontSeriesChangeRule {bsx}{c} {bc} {}
858 \DeclareFontSeriesChangeRule {bsx}{sc} {bsc} {}
859 \DeclareFontSeriesChangeRule {bsx}{sx} {bsx} {}
860 \DeclareFontSeriesChangeRule {bsx}{x} {bx} {}
861 \DeclareFontSeriesChangeRule {bsx}{ex} {bex} {}
862 \DeclareFontSeriesChangeRule {bsx}{ux} {bux} {}

863 \DeclareFontSeriesChangeRule {bx}{ul} {ulx} {}
864 \DeclareFontSeriesChangeRule {bx}{el} {elx} {}
865 \DeclareFontSeriesChangeRule {bx}{l} {lx} {}
866 \DeclareFontSeriesChangeRule {bx}{sl} {slx} {}
867 \DeclareFontSeriesChangeRule {bx}{sb} {sbx} {bx}
868 \DeclareFontSeriesChangeRule {bx}{b} {bx} {}
869 \DeclareFontSeriesChangeRule {bx}{eb} {ebx} {bx}
870 \DeclareFontSeriesChangeRule {bx}{ub} {ubx} {bx}
871 \DeclareFontSeriesChangeRule {bx}{uc} {buc} {}

```

```

872 \DeclareFontSeriesChangeRule {bx}{ec} {bec} {}
873 \DeclareFontSeriesChangeRule {bx}{c} {bc} {}
874 \DeclareFontSeriesChangeRule {bx}{sc} {bsc} {}
875 \DeclareFontSeriesChangeRule {bx}{sx} {bsx} {}
876 \DeclareFontSeriesChangeRule {bx}{x} {bx} {}
877 \DeclareFontSeriesChangeRule {bx}{ex} {bex} {}
878 \DeclareFontSeriesChangeRule {bx}{ux} {bux} {}

879 \DeclareFontSeriesChangeRule {bex}{ul} {ulex} {}
880 \DeclareFontSeriesChangeRule {bex}{el} {ellex} {}
881 \DeclareFontSeriesChangeRule {bex}{l} {llex} {}
882 \DeclareFontSeriesChangeRule {bex}{sl} {sllex} {}
883 \DeclareFontSeriesChangeRule {bex}{sb} {sbex} {bex}
884 \DeclareFontSeriesChangeRule {bex}{b} {bex} {}
885 \DeclareFontSeriesChangeRule {bex}{eb} {ebex} {bex}
886 \DeclareFontSeriesChangeRule {bex}{ub} {ubex} {bex}
887 \DeclareFontSeriesChangeRule {bex}{uc} {buc} {}
888 \DeclareFontSeriesChangeRule {bex}{ec} {bec} {}
889 \DeclareFontSeriesChangeRule {bex}{c} {bc} {}
890 \DeclareFontSeriesChangeRule {bex}{sc} {bsc} {}
891 \DeclareFontSeriesChangeRule {bex}{sx} {bsx} {}
892 \DeclareFontSeriesChangeRule {bex}{x} {bx} {}
893 \DeclareFontSeriesChangeRule {bex}{ex} {bex} {}
894 \DeclareFontSeriesChangeRule {bex}{ux} {bux} {}

895 \DeclareFontSeriesChangeRule {bux}{ul} {ulux} {}
896 \DeclareFontSeriesChangeRule {bux}{el} {elux} {}
897 \DeclareFontSeriesChangeRule {bux}{l} {lux} {}
898 \DeclareFontSeriesChangeRule {bux}{sl} {slux} {}
899 \DeclareFontSeriesChangeRule {bux}{sb} {sbux} {bux}
900 \DeclareFontSeriesChangeRule {bux}{b} {bux} {}
901 \DeclareFontSeriesChangeRule {bux}{eb} {ebux} {bux}
902 \DeclareFontSeriesChangeRule {bux}{ub} {ubux} {bux}
903 \DeclareFontSeriesChangeRule {bux}{uc} {buc} {}
904 \DeclareFontSeriesChangeRule {bux}{ec} {bec} {}
905 \DeclareFontSeriesChangeRule {bux}{c} {bc} {}
906 \DeclareFontSeriesChangeRule {bux}{sc} {bsc} {}
907 \DeclareFontSeriesChangeRule {bux}{sx} {bsx} {}
908 \DeclareFontSeriesChangeRule {bux}{x} {bx} {}
909 \DeclareFontSeriesChangeRule {bux}{ex} {bex} {}
910 \DeclareFontSeriesChangeRule {bux}{ux} {bux} {}

911 \DeclareFontSeriesChangeRule {ebuc}{ul} {uluc} {}
912 \DeclareFontSeriesChangeRule {ebuc}{el} {eluc} {}
913 \DeclareFontSeriesChangeRule {ebuc}{l} {luc} {}
914 \DeclareFontSeriesChangeRule {ebuc}{sl} {sluc} {}
915 \DeclareFontSeriesChangeRule {ebuc}{sb} {sbuc} {buc}
916 \DeclareFontSeriesChangeRule {ebuc}{b} {buc} {}
917 \DeclareFontSeriesChangeRule {ebuc}{eb} {ebuc} {}

```

In the following rule, we use `eb` instead of `b` in the fourth argument, since `eb` is a better approximation to `ub` than `b` and `ebuc` is already in the first argument and we can therefore assume that this font face actually exists. A similar consideration also applies to some other rules in the following.

```

918 \DeclareFontSeriesChangeRule {ebuc}{ub} {ubuc} {ebuc}
919 \DeclareFontSeriesChangeRule {ebuc}{uc} {ebuc} {}

```



```

920 \DeclareFontSeriesChangeRule {ebuc}{ec} {ebec} {}
921 \DeclareFontSeriesChangeRule {ebuc}{c} {ebc} {}
922 \DeclareFontSeriesChangeRule {ebuc}{sc} {ebsc} {}
923 \DeclareFontSeriesChangeRule {ebuc}{sx} {ebsx} {}
924 \DeclareFontSeriesChangeRule {ebuc}{x} {ebx} {}
925 \DeclareFontSeriesChangeRule {ebuc}{ex} {ebex} {}
926 \DeclareFontSeriesChangeRule {ebuc}{ux} {ebux} {}

927 \DeclareFontSeriesChangeRule {ebec}{ul} {ulec} {}
928 \DeclareFontSeriesChangeRule {ebec}{el} {elec} {}
929 \DeclareFontSeriesChangeRule {ebec}{l} {lec} {}
930 \DeclareFontSeriesChangeRule {ebec}{sl} {slec} {}
931 \DeclareFontSeriesChangeRule {ebec}{sb} {sbec} {bec}
932 \DeclareFontSeriesChangeRule {ebec}{b} {bec} {}
933 \DeclareFontSeriesChangeRule {ebec}{eb} {ebec} {}
934 \DeclareFontSeriesChangeRule {ebec}{ub} {ubec} {ebec}
935 \DeclareFontSeriesChangeRule {ebec}{uc} {ebuc} {}
936 \DeclareFontSeriesChangeRule {ebec}{ec} {ebec} {}
937 \DeclareFontSeriesChangeRule {ebec}{c} {ebc} {}
938 \DeclareFontSeriesChangeRule {ebec}{sc} {ebsc} {}
939 \DeclareFontSeriesChangeRule {ebec}{sx} {ebsx} {}
940 \DeclareFontSeriesChangeRule {ebec}{x} {ebx} {}
941 \DeclareFontSeriesChangeRule {ebec}{ex} {ebex} {}
942 \DeclareFontSeriesChangeRule {ebec}{ux} {ebux} {}

943 \DeclareFontSeriesChangeRule {ebc}{ul} {ulc} {}
944 \DeclareFontSeriesChangeRule {ebc}{el} {elc} {}
945 \DeclareFontSeriesChangeRule {ebc}{l} {lc} {}
946 \DeclareFontSeriesChangeRule {ebc}{sl} {slc} {}
947 \DeclareFontSeriesChangeRule {ebc}{sb} {sbc} {bc}
948 \DeclareFontSeriesChangeRule {ebc}{b} {bc} {}
949 \DeclareFontSeriesChangeRule {ebc}{eb} {ebc} {}
950 \DeclareFontSeriesChangeRule {ebc}{ub} {ubc} {ebc}
951 \DeclareFontSeriesChangeRule {ebc}{uc} {ebuc} {}
952 \DeclareFontSeriesChangeRule {ebc}{ec} {ebec} {}
953 \DeclareFontSeriesChangeRule {ebc}{c} {ebc} {}
954 \DeclareFontSeriesChangeRule {ebc}{sc} {ebsc} {}
955 \DeclareFontSeriesChangeRule {ebc}{sx} {ebsx} {}
956 \DeclareFontSeriesChangeRule {ebc}{x} {ebx} {}
957 \DeclareFontSeriesChangeRule {ebc}{ex} {ebex} {}
958 \DeclareFontSeriesChangeRule {ebc}{ux} {ebux} {}

959 \DeclareFontSeriesChangeRule {ebsc}{ul} {ulsc} {}
960 \DeclareFontSeriesChangeRule {ebsc}{el} {elsc} {}
961 \DeclareFontSeriesChangeRule {ebsc}{l} {lsc} {}
962 \DeclareFontSeriesChangeRule {ebsc}{sl} {slsc} {}
963 \DeclareFontSeriesChangeRule {ebsc}{sb} {sbsc} {bsc}
964 \DeclareFontSeriesChangeRule {ebsc}{b} {bsc} {}
965 \DeclareFontSeriesChangeRule {ebsc}{eb} {ebsc} {}
966 \DeclareFontSeriesChangeRule {ebsc}{ub} {ubsc} {ebsc}
967 \DeclareFontSeriesChangeRule {ebsc}{uc} {ebuc} {}
968 \DeclareFontSeriesChangeRule {ebsc}{ec} {ebec} {}
969 \DeclareFontSeriesChangeRule {ebsc}{c} {ebc} {}
970 \DeclareFontSeriesChangeRule {ebsc}{sc} {ebsc} {}
971 \DeclareFontSeriesChangeRule {ebsc}{sx} {ebsx} {}
972 \DeclareFontSeriesChangeRule {ebsc}{x} {ebx} {}

```

```

973 \DeclareFontSeriesChangeRule {ebsc}{ex} {ebex} {}
974 \DeclareFontSeriesChangeRule {ebsc}{ux} {ebux} {}

975 \DeclareFontSeriesChangeRule {eb}{uc} {ebuc} {}
976 \DeclareFontSeriesChangeRule {eb}{ec} {ebec} {}
977 \DeclareFontSeriesChangeRule {eb}{c} {ebc} {}
978 \DeclareFontSeriesChangeRule {eb}{sc} {ebsc} {}
979 \DeclareFontSeriesChangeRule {eb}{sx} {ebsx} {}
980 \DeclareFontSeriesChangeRule {eb}{x} {ebx} {}
981 \DeclareFontSeriesChangeRule {eb}{ex} {ebex} {}
982 \DeclareFontSeriesChangeRule {eb}{ux} {ebux} {}
983 \DeclareFontSeriesChangeRule {eb}{sb} {sb} {b}
984 \DeclareFontSeriesChangeRule {eb}{ub} {ub} {eb}

985 \DeclareFontSeriesChangeRule {ebsx}{ul} {ulsx} {}
986 \DeclareFontSeriesChangeRule {ebsx}{el} {elsx} {}
987 \DeclareFontSeriesChangeRule {ebsx}{l} {lsx} {}
988 \DeclareFontSeriesChangeRule {ebsx}{sl} {slsx} {}
989 \DeclareFontSeriesChangeRule {ebsx}{sb} {sbsx} {bsx}
990 \DeclareFontSeriesChangeRule {ebsx}{b} {bsx} {}
991 \DeclareFontSeriesChangeRule {ebsx}{eb} {ebsx} {}
992 \DeclareFontSeriesChangeRule {ebsx}{ub} {ubsx} {ebsx}
993 \DeclareFontSeriesChangeRule {ebsx}{uc} {ebuc} {}
994 \DeclareFontSeriesChangeRule {ebsx}{ec} {ebec} {}
995 \DeclareFontSeriesChangeRule {ebsx}{c} {ebc} {}
996 \DeclareFontSeriesChangeRule {ebsx}{sc} {ebsc} {}
997 \DeclareFontSeriesChangeRule {ebsx}{sx} {ebsx} {}
998 \DeclareFontSeriesChangeRule {ebsx}{x} {ebx} {}
999 \DeclareFontSeriesChangeRule {ebsx}{ex} {ebex} {}
1000 \DeclareFontSeriesChangeRule {ebsx}{ux} {ebux} {}

1001 \DeclareFontSeriesChangeRule {ebx}{ul} {ulx} {}
1002 \DeclareFontSeriesChangeRule {ebx}{el} {elx} {}
1003 \DeclareFontSeriesChangeRule {ebx}{l} {lx} {}
1004 \DeclareFontSeriesChangeRule {ebx}{sl} {slx} {}
1005 \DeclareFontSeriesChangeRule {ebx}{sb} {sbx} {bx}
1006 \DeclareFontSeriesChangeRule {ebx}{b} {bx} {}
1007 \DeclareFontSeriesChangeRule {ebx}{eb} {ebx} {}
1008 \DeclareFontSeriesChangeRule {ebx}{ub} {ubx} {ebx}
1009 \DeclareFontSeriesChangeRule {ebx}{uc} {ebuc} {}
1010 \DeclareFontSeriesChangeRule {ebx}{ec} {ebec} {}
1011 \DeclareFontSeriesChangeRule {ebx}{c} {ebc} {}
1012 \DeclareFontSeriesChangeRule {ebx}{sc} {ebsc} {}
1013 \DeclareFontSeriesChangeRule {ebx}{sx} {ebsx} {}
1014 \DeclareFontSeriesChangeRule {ebx}{x} {ebx} {}
1015 \DeclareFontSeriesChangeRule {ebx}{ex} {ebex} {}
1016 \DeclareFontSeriesChangeRule {ebx}{ux} {ebux} {}

1017 \DeclareFontSeriesChangeRule {ebex}{ul} {ulex} {}
1018 \DeclareFontSeriesChangeRule {ebex}{el} {elex} {}
1019 \DeclareFontSeriesChangeRule {ebex}{l} {lex} {}
1020 \DeclareFontSeriesChangeRule {ebex}{sl} {slex} {}
1021 \DeclareFontSeriesChangeRule {ebex}{sb} {sbox} {box}
1022 \DeclareFontSeriesChangeRule {ebex}{b} {box} {}
1023 \DeclareFontSeriesChangeRule {ebex}{eb} {ebex} {}
1024 \DeclareFontSeriesChangeRule {ebex}{ub} {ubex} {ebex}
1025 \DeclareFontSeriesChangeRule {ebex}{uc} {ebuc} {}

```

```

1026 \DeclareFontSeriesChangeRule {ebex}{ec} {ebec} {}
1027 \DeclareFontSeriesChangeRule {ebex}{c} {ebc} {}
1028 \DeclareFontSeriesChangeRule {ebex}{sc} {ebsc} {}
1029 \DeclareFontSeriesChangeRule {ebex}{sx} {ebsx} {}
1030 \DeclareFontSeriesChangeRule {ebex}{x} {ebx} {}
1031 \DeclareFontSeriesChangeRule {ebex}{ex} {ebex} {}
1032 \DeclareFontSeriesChangeRule {ebex}{ux} {ebux} {}

1033 \DeclareFontSeriesChangeRule {ebux}{ul} {ulux} {}
1034 \DeclareFontSeriesChangeRule {ebux}{el} {elux} {}
1035 \DeclareFontSeriesChangeRule {ebux}{l} {lux} {}
1036 \DeclareFontSeriesChangeRule {ebux}{sl} {slux} {}
1037 \DeclareFontSeriesChangeRule {ebux}{sb} {sbux} {bux}
1038 \DeclareFontSeriesChangeRule {ebux}{b} {bux} {}
1039 \DeclareFontSeriesChangeRule {ebux}{eb} {ebux} {}
1040 \DeclareFontSeriesChangeRule {ebux}{ub} {ubux} {ebux}
1041 \DeclareFontSeriesChangeRule {ebux}{uc} {ebuc} {}
1042 \DeclareFontSeriesChangeRule {ebux}{ec} {ebec} {}
1043 \DeclareFontSeriesChangeRule {ebux}{c} {ebc} {}
1044 \DeclareFontSeriesChangeRule {ebux}{sc} {ebsc} {}
1045 \DeclareFontSeriesChangeRule {ebux}{sx} {ebsx} {}
1046 \DeclareFontSeriesChangeRule {ebux}{x} {ebx} {}
1047 \DeclareFontSeriesChangeRule {ebux}{ex} {ebex} {}
1048 \DeclareFontSeriesChangeRule {ebux}{ux} {ebux} {}

1049 \DeclareFontSeriesChangeRule {ubuc}{ul} {uluc} {}
1050 \DeclareFontSeriesChangeRule {ubuc}{el} {eluc} {}
1051 \DeclareFontSeriesChangeRule {ubuc}{l} {luc} {}
1052 \DeclareFontSeriesChangeRule {ubuc}{sl} {sluc} {}
1053 \DeclareFontSeriesChangeRule {ubuc}{sb} {sbuc} {buc}
1054 \DeclareFontSeriesChangeRule {ubuc}{b} {buc} {}
1055 \DeclareFontSeriesChangeRule {ubuc}{eb} {ebuc} {ubuc}
1056 \DeclareFontSeriesChangeRule {ubuc}{ub} {ubuc} {}
1057 \DeclareFontSeriesChangeRule {ubuc}{uc} {ubuc} {}
1058 \DeclareFontSeriesChangeRule {ubuc}{ec} {ubec} {}
1059 \DeclareFontSeriesChangeRule {ubuc}{c} {ubc} {}
1060 \DeclareFontSeriesChangeRule {ubuc}{sc} {ubsc} {}
1061 \DeclareFontSeriesChangeRule {ubuc}{sx} {ubsx} {}
1062 \DeclareFontSeriesChangeRule {ubuc}{x} {ubx} {}
1063 \DeclareFontSeriesChangeRule {ubuc}{ex} {ubex} {}
1064 \DeclareFontSeriesChangeRule {ubuc}{ux} {ubux} {}

1065 \DeclareFontSeriesChangeRule {ubec}{ul} {ulec} {}
1066 \DeclareFontSeriesChangeRule {ubec}{el} {elec} {}
1067 \DeclareFontSeriesChangeRule {ubec}{l} {lec} {}
1068 \DeclareFontSeriesChangeRule {ubec}{sl} {slec} {}
1069 \DeclareFontSeriesChangeRule {ubec}{sb} {sbec} {bec}
1070 \DeclareFontSeriesChangeRule {ubec}{b} {bec} {}
1071 \DeclareFontSeriesChangeRule {ubec}{eb} {ebec} {ubec}
1072 \DeclareFontSeriesChangeRule {ubec}{ub} {ubec} {}
1073 \DeclareFontSeriesChangeRule {ubec}{uc} {ubuc} {}
1074 \DeclareFontSeriesChangeRule {ubec}{ec} {ubec} {}
1075 \DeclareFontSeriesChangeRule {ubec}{c} {ubc} {}
1076 \DeclareFontSeriesChangeRule {ubec}{sc} {ubsc} {}
1077 \DeclareFontSeriesChangeRule {ubec}{sx} {ubsx} {}
1078 \DeclareFontSeriesChangeRule {ubec}{x} {ubx} {}

```

```

1079 \DeclareFontSeriesChangeRule {ubec}{ex} {ubex} {}
1080 \DeclareFontSeriesChangeRule {ubec}{ux} {ubux} {}

1081 \DeclareFontSeriesChangeRule {ubc}{ul} {ulc} {}
1082 \DeclareFontSeriesChangeRule {ubc}{el} {elc} {}
1083 \DeclareFontSeriesChangeRule {ubc}{l} {lc} {}
1084 \DeclareFontSeriesChangeRule {ubc}{sl} {slc} {}
1085 \DeclareFontSeriesChangeRule {ubc}{sb} {sbc} {bc}
1086 \DeclareFontSeriesChangeRule {ubc}{b} {bc} {}
1087 \DeclareFontSeriesChangeRule {ubc}{eb} {ebc} {ubc}
1088 \DeclareFontSeriesChangeRule {ubc}{ub} {ubc} {}
1089 \DeclareFontSeriesChangeRule {ubc}{uc} {ubuc} {}
1090 \DeclareFontSeriesChangeRule {ubc}{ec} {ubec} {}
1091 \DeclareFontSeriesChangeRule {ubc}{c} {ubc} {}
1092 \DeclareFontSeriesChangeRule {ubc}{sc} {ubsc} {}
1093 \DeclareFontSeriesChangeRule {ubc}{sx} {ubsx} {}
1094 \DeclareFontSeriesChangeRule {ubc}{x} {ubx} {}
1095 \DeclareFontSeriesChangeRule {ubc}{ex} {ubex} {}
1096 \DeclareFontSeriesChangeRule {ubc}{ux} {ubux} {}

1097 \DeclareFontSeriesChangeRule {ubsc}{ul} {ulsc} {}
1098 \DeclareFontSeriesChangeRule {ubsc}{el} {elsc} {}
1099 \DeclareFontSeriesChangeRule {ubsc}{l} {lsc} {}
1100 \DeclareFontSeriesChangeRule {ubsc}{sl} {slsc} {}
1101 \DeclareFontSeriesChangeRule {ubsc}{sb} {sbsc} {bsc}
1102 \DeclareFontSeriesChangeRule {ubsc}{b} {bsc} {}
1103 \DeclareFontSeriesChangeRule {ubsc}{eb} {ebsc} {ubsc}
1104 \DeclareFontSeriesChangeRule {ubsc}{ub} {ubsc} {}
1105 \DeclareFontSeriesChangeRule {ubsc}{uc} {ubuc} {}
1106 \DeclareFontSeriesChangeRule {ubsc}{ec} {ubec} {}
1107 \DeclareFontSeriesChangeRule {ubsc}{c} {ubc} {}
1108 \DeclareFontSeriesChangeRule {ubsc}{sc} {ubsc} {}
1109 \DeclareFontSeriesChangeRule {ubsc}{sx} {ubsx} {}
1110 \DeclareFontSeriesChangeRule {ubsc}{x} {ubx} {}
1111 \DeclareFontSeriesChangeRule {ubsc}{ex} {ubex} {}
1112 \DeclareFontSeriesChangeRule {ubsc}{ux} {ubux} {}

1113 \DeclareFontSeriesChangeRule {ub}{uc} {ubuc} {}
1114 \DeclareFontSeriesChangeRule {ub}{ec} {ubec} {}
1115 \DeclareFontSeriesChangeRule {ub}{c} {ubc} {}
1116 \DeclareFontSeriesChangeRule {ub}{sc} {ubsc} {}
1117 \DeclareFontSeriesChangeRule {ub}{sx} {ubsx} {}
1118 \DeclareFontSeriesChangeRule {ub}{x} {ubx} {}
1119 \DeclareFontSeriesChangeRule {ub}{ex} {ubex} {}
1120 \DeclareFontSeriesChangeRule {ub}{ux} {ubux} {}
1121 \DeclareFontSeriesChangeRule {ub}{sb} {sb} {b}
1122 \DeclareFontSeriesChangeRule {ub}{eb} {eb} {ub}

1123 \DeclareFontSeriesChangeRule {ubsx}{ul} {ulsx} {}
1124 \DeclareFontSeriesChangeRule {ubsx}{el} {elsx} {}
1125 \DeclareFontSeriesChangeRule {ubsx}{l} {lsx} {}
1126 \DeclareFontSeriesChangeRule {ubsx}{sl} {slsx} {}
1127 \DeclareFontSeriesChangeRule {ubsx}{sb} {sbsx} {bsx}
1128 \DeclareFontSeriesChangeRule {ubsx}{b} {bsx} {}
1129 \DeclareFontSeriesChangeRule {ubsx}{eb} {ebsx} {ubsx}
1130 \DeclareFontSeriesChangeRule {ubsx}{ub} {ubsx} {}
1131 \DeclareFontSeriesChangeRule {ubsx}{uc} {ubuc} {}

```

1132 \DeclareFontSeriesChangeRule {ubsx}{ec} {ubec} {}
1133 \DeclareFontSeriesChangeRule {ubsx}{c} {ubc} {}
1134 \DeclareFontSeriesChangeRule {ubsx}{sc} {ubsc} {}
1135 \DeclareFontSeriesChangeRule {ubsx}{sx} {ubsx} {}
1136 \DeclareFontSeriesChangeRule {ubsx}{x} {ubx} {}
1137 \DeclareFontSeriesChangeRule {ubsx}{ex} {ubex} {}
1138 \DeclareFontSeriesChangeRule {ubsx}{ux} {ubux} {}

1139 \DeclareFontSeriesChangeRule {ubx}{ul} {ulx} {}
1140 \DeclareFontSeriesChangeRule {ubx}{el} {elx} {}
1141 \DeclareFontSeriesChangeRule {ubx}{l} {lx} {}
1142 \DeclareFontSeriesChangeRule {ubx}{sl} {slx} {}
1143 \DeclareFontSeriesChangeRule {ubx}{sb} {sbx} {bx}
1144 \DeclareFontSeriesChangeRule {ubx}{b} {bx} {}
1145 \DeclareFontSeriesChangeRule {ubx}{eb} {ebx} {ubx}
1146 \DeclareFontSeriesChangeRule {ubx}{ub} {ubx} {}
1147 \DeclareFontSeriesChangeRule {ubx}{uc} {ubuc} {}
1148 \DeclareFontSeriesChangeRule {ubx}{ec} {ubec} {}
1149 \DeclareFontSeriesChangeRule {ubx}{c} {ubc} {}
1150 \DeclareFontSeriesChangeRule {ubx}{sc} {ubsc} {}
1151 \DeclareFontSeriesChangeRule {ubx}{sx} {ubsx} {}
1152 \DeclareFontSeriesChangeRule {ubx}{x} {ubx} {}
1153 \DeclareFontSeriesChangeRule {ubx}{ex} {ubex} {}
1154 \DeclareFontSeriesChangeRule {ubx}{ux} {ubux} {}

1155 \DeclareFontSeriesChangeRule {ubex}{ul} {ulex} {}
1156 \DeclareFontSeriesChangeRule {ubex}{el} {elx} {}
1157 \DeclareFontSeriesChangeRule {ubex}{l} {lex} {}
1158 \DeclareFontSeriesChangeRule {ubex}{sl} {slex} {}
1159 \DeclareFontSeriesChangeRule {ubex}{sb} {sbex} {bex}
1160 \DeclareFontSeriesChangeRule {ubex}{b} {bex} {}
1161 \DeclareFontSeriesChangeRule {ubex}{eb} {ebx} {ubex}
1162 \DeclareFontSeriesChangeRule {ubex}{ub} {ubex} {}
1163 \DeclareFontSeriesChangeRule {ubex}{uc} {ubuc} {}
1164 \DeclareFontSeriesChangeRule {ubex}{ec} {ubec} {}
1165 \DeclareFontSeriesChangeRule {ubex}{c} {ubc} {}
1166 \DeclareFontSeriesChangeRule {ubex}{sc} {ubsc} {}
1167 \DeclareFontSeriesChangeRule {ubex}{sx} {ubsx} {}
1168 \DeclareFontSeriesChangeRule {ubex}{x} {ubx} {}
1169 \DeclareFontSeriesChangeRule {ubex}{ex} {ubex} {}
1170 \DeclareFontSeriesChangeRule {ubex}{ux} {ubux} {}

1171 \DeclareFontSeriesChangeRule {ubux}{ul} {ulux} {}
1172 \DeclareFontSeriesChangeRule {ubux}{el} {elux} {}
1173 \DeclareFontSeriesChangeRule {ubux}{l} {lux} {}
1174 \DeclareFontSeriesChangeRule {ubux}{sl} {slux} {}
1175 \DeclareFontSeriesChangeRule {ubux}{sb} {sbux} {bux}
1176 \DeclareFontSeriesChangeRule {ubux}{b} {bux} {}
1177 \DeclareFontSeriesChangeRule {ubux}{eb} {ebx} {ubux}
1178 \DeclareFontSeriesChangeRule {ubux}{ub} {ubux} {}
1179 \DeclareFontSeriesChangeRule {ubux}{uc} {ubuc} {}
1180 \DeclareFontSeriesChangeRule {ubux}{ec} {ubec} {}
1181 \DeclareFontSeriesChangeRule {ubux}{c} {ubc} {}
1182 \DeclareFontSeriesChangeRule {ubux}{sc} {ubsc} {}
1183 \DeclareFontSeriesChangeRule {ubux}{sx} {ubsx} {}
1184 \DeclareFontSeriesChangeRule {ubux}{x} {ubx} {}

```

1185 \DeclareFontSeriesChangeRule {ubux}{ex} {ubex} {}
1186 \DeclareFontSeriesChangeRule {ubux}{ux} {ubux} {}

```

Special rules for `lm` etc. aren't needed because if `lm` is requested, it will be used if there is no rule, and that is then reduced to `l` automatically. Same for `mc` and friends.

The following entries handle a request for `bx` and fall back to `b` if that can't be fulfilled.

```

1187 \DeclareFontSeriesChangeRule {uluc}{bx} {bx} {b}
1188 \DeclareFontSeriesChangeRule {ulec}{bx} {bx} {b}
1189 \DeclareFontSeriesChangeRule {ulc}{bx} {bx} {b}
1190 \DeclareFontSeriesChangeRule {ulsc}{bx} {bx} {b}
1191 \DeclareFontSeriesChangeRule {ul}{bx} {bx} {b}
1192 \DeclareFontSeriesChangeRule {ulsx}{bx} {bx} {b}
1193 \DeclareFontSeriesChangeRule {ulx}{bx} {bx} {b}
1194 \DeclareFontSeriesChangeRule {ulex}{bx} {bx} {b}
1195 \DeclareFontSeriesChangeRule {ulux}{bx} {bx} {b}

1196 \DeclareFontSeriesChangeRule {eluc}{bx} {bx} {b}
1197 \DeclareFontSeriesChangeRule {elec}{bx} {bx} {b}
1198 \DeclareFontSeriesChangeRule {elc}{bx} {bx} {b}
1199 \DeclareFontSeriesChangeRule {elsc}{bx} {bx} {b}
1200 \DeclareFontSeriesChangeRule {el}{bx} {bx} {b}
1201 \DeclareFontSeriesChangeRule {elsx}{bx} {bx} {b}
1202 \DeclareFontSeriesChangeRule {elx}{bx} {bx} {b}
1203 \DeclareFontSeriesChangeRule {ellex}{bx} {bx} {b}
1204 \DeclareFontSeriesChangeRule {elux}{bx} {bx} {b}

1205 \DeclareFontSeriesChangeRule {luc}{bx} {bx} {b}
1206 \DeclareFontSeriesChangeRule {lec}{bx} {bx} {b}
1207 \DeclareFontSeriesChangeRule {lc}{bx} {bx} {b}
1208 \DeclareFontSeriesChangeRule {lsc}{bx} {bx} {b}
1209 \DeclareFontSeriesChangeRule {l}{bx} {bx} {b}
1210 \DeclareFontSeriesChangeRule {lsx}{bx} {bx} {b}
1211 \DeclareFontSeriesChangeRule {lx}{bx} {bx} {b}
1212 \DeclareFontSeriesChangeRule {lex}{bx} {bx} {b}
1213 \DeclareFontSeriesChangeRule {lux}{bx} {bx} {b}

1214 \DeclareFontSeriesChangeRule {sluc}{bx} {bx} {b}
1215 \DeclareFontSeriesChangeRule {slec}{bx} {bx} {b}
1216 \DeclareFontSeriesChangeRule {slc}{bx} {bx} {b}
1217 \DeclareFontSeriesChangeRule {slsc}{bx} {bx} {b}
1218 \DeclareFontSeriesChangeRule {sl}{bx} {bx} {b}
1219 \DeclareFontSeriesChangeRule {slsx}{bx} {bx} {b}
1220 \DeclareFontSeriesChangeRule {slx}{bx} {bx} {b}
1221 \DeclareFontSeriesChangeRule {slex}{bx} {bx} {b}
1222 \DeclareFontSeriesChangeRule {slux}{bx} {bx} {b}

1223 \DeclareFontSeriesChangeRule {sbuc}{bx} {bx} {b}
1224 \DeclareFontSeriesChangeRule {sbec}{bx} {bx} {b}
1225 \DeclareFontSeriesChangeRule {sbc}{bx} {bx} {b}
1226 \DeclareFontSeriesChangeRule {sbsc}{bx} {bx} {b}
1227 \DeclareFontSeriesChangeRule {sb}{bx} {bx} {b}
1228 \DeclareFontSeriesChangeRule {sbsx}{bx} {bx} {b}
1229 \DeclareFontSeriesChangeRule {sbx}{bx} {bx} {b}
1230 \DeclareFontSeriesChangeRule {sbex}{bx} {bx} {b}
1231 \DeclareFontSeriesChangeRule {sbux}{bx} {bx} {b}

```

```

1232 \DeclareFontSeriesChangeRule {buc}{bx} {bx} {b}
1233 \DeclareFontSeriesChangeRule {bec}{bx} {bx} {b}
1234 \DeclareFontSeriesChangeRule {bc}{bx} {bx} {b}
1235 \DeclareFontSeriesChangeRule {bsc}{bx} {bx} {b}
1236 \DeclareFontSeriesChangeRule {b}{bx} {bx} {b}
1237 \DeclareFontSeriesChangeRule {bsx}{bx} {bx} {b}
1238 \DeclareFontSeriesChangeRule {bx}{bx} {bx} {b}
1239 \DeclareFontSeriesChangeRule {bex}{bx} {bx} {b}
1240 \DeclareFontSeriesChangeRule {bux}{bx} {bx} {b}

1241 \DeclareFontSeriesChangeRule {ebuc}{bx} {bx} {b}
1242 \DeclareFontSeriesChangeRule {ebec}{bx} {bx} {b}
1243 \DeclareFontSeriesChangeRule {ebc}{bx} {bx} {b}
1244 \DeclareFontSeriesChangeRule {ebsc}{bx} {bx} {b}
1245 \DeclareFontSeriesChangeRule {eb}{bx} {bx} {b}
1246 \DeclareFontSeriesChangeRule {ebsx}{bx} {bx} {b}
1247 \DeclareFontSeriesChangeRule {ebx}{bx} {bx} {b}
1248 \DeclareFontSeriesChangeRule {ebex}{bx} {bx} {b}
1249 \DeclareFontSeriesChangeRule {ebux}{bx} {bx} {b}

1250 \DeclareFontSeriesChangeRule {ubuc}{bx} {bx} {b}
1251 \DeclareFontSeriesChangeRule {ubec}{bx} {bx} {b}
1252 \DeclareFontSeriesChangeRule {ubc}{bx} {bx} {b}
1253 \DeclareFontSeriesChangeRule {ubsc}{bx} {bx} {b}
1254 \DeclareFontSeriesChangeRule {ub}{bx} {bx} {b}
1255 \DeclareFontSeriesChangeRule {ubsx}{bx} {bx} {b}
1256 \DeclareFontSeriesChangeRule {ubx}{bx} {bx} {b}
1257 \DeclareFontSeriesChangeRule {ubex}{bx} {bx} {b}
1258 \DeclareFontSeriesChangeRule {ubux}{bx} {bx} {b}

1259 \DeclareFontSeriesChangeRule {uc}{bx} {bx} {b}
1260 \DeclareFontSeriesChangeRule {ec}{bx} {bx} {b}
1261 \DeclareFontSeriesChangeRule {c}{bx} {bx} {b}
1262 \DeclareFontSeriesChangeRule {sc}{bx} {bx} {b}
1263 \DeclareFontSeriesChangeRule {m}{bx} {bx} {b}
1264 \DeclareFontSeriesChangeRule {sx}{bx} {bx} {b}
1265 \DeclareFontSeriesChangeRule {x}{bx} {bx} {b}
1266 \DeclareFontSeriesChangeRule {ex}{bx} {bx} {b}
1267 \DeclareFontSeriesChangeRule {ux}{bx} {bx} {b}

```

Here are the special rules for m?:

```

1268 \DeclareFontSeriesChangeRule {uluc}{m?} {uc} {}
1269 \DeclareFontSeriesChangeRule {ulec}{m?} {ec} {}
1270 \DeclareFontSeriesChangeRule {ulc}{m?} {c} {}
1271 \DeclareFontSeriesChangeRule {ulsc}{m?} {sc} {}
1272 \DeclareFontSeriesChangeRule {ul}{m?} {m} {}
1273 \DeclareFontSeriesChangeRule {ulsx}{m?} {sx} {}
1274 \DeclareFontSeriesChangeRule {ulx}{m?} {x} {}
1275 \DeclareFontSeriesChangeRule {ulex}{m?} {ex} {}
1276 \DeclareFontSeriesChangeRule {ulux}{m?} {ux} {}
1277 \DeclareFontSeriesChangeRule {eluc}{m?} {uc} {}
1278 \DeclareFontSeriesChangeRule {elec}{m?} {ec} {}
1279 \DeclareFontSeriesChangeRule {elc}{m?} {c} {}
1280 \DeclareFontSeriesChangeRule {elsc}{m?} {sc} {}
1281 \DeclareFontSeriesChangeRule {el}{m?} {m} {}
1282 \DeclareFontSeriesChangeRule {elsx}{m?} {sx} {}

```

1283 \DeclareFontSeriesChangeRule {elx}{m?} {x} {}
1284 \DeclareFontSeriesChangeRule {elx}{m?} {ex} {}
1285 \DeclareFontSeriesChangeRule {elux}{m?} {ux} {}
1286 \DeclareFontSeriesChangeRule {luc}{m?} {uc} {}
1287 \DeclareFontSeriesChangeRule {lec}{m?} {ec} {}
1288 \DeclareFontSeriesChangeRule {lc}{m?} {c} {}
1289 \DeclareFontSeriesChangeRule {lsc}{m?} {sc} {}
1290 \DeclareFontSeriesChangeRule {l}{m?} {m} {}
1291 \DeclareFontSeriesChangeRule {lsx}{m?} {sx} {}
1292 \DeclareFontSeriesChangeRule {lx}{m?} {x} {}
1293 \DeclareFontSeriesChangeRule {lex}{m?} {ex} {}
1294 \DeclareFontSeriesChangeRule {lux}{m?} {ux} {}
1295 \DeclareFontSeriesChangeRule {sluc}{m?} {uc} {}
1296 \DeclareFontSeriesChangeRule {slec}{m?} {ec} {}
1297 \DeclareFontSeriesChangeRule {slc}{m?} {c} {}
1298 \DeclareFontSeriesChangeRule {slsc}{m?} {sc} {}
1299 \DeclareFontSeriesChangeRule {sl}{m?} {m} {}
1300 \DeclareFontSeriesChangeRule {slsx}{m?} {sx} {}
1301 \DeclareFontSeriesChangeRule {slx}{m?} {x} {}
1302 \DeclareFontSeriesChangeRule {slex}{m?} {ex} {}
1303 \DeclareFontSeriesChangeRule {slux}{m?} {ux} {}
1304 \DeclareFontSeriesChangeRule {uc}{m?} {uc} {}
1305 \DeclareFontSeriesChangeRule {ec}{m?} {ec} {}
1306 \DeclareFontSeriesChangeRule {c}{m?} {c} {}
1307 \DeclareFontSeriesChangeRule {sc}{m?} {sc} {}
1308 \DeclareFontSeriesChangeRule {m}{m?} {m} {}
1309 \DeclareFontSeriesChangeRule {sx}{m?} {sx} {}
1310 \DeclareFontSeriesChangeRule {x}{m?} {x} {}
1311 \DeclareFontSeriesChangeRule {ex}{m?} {ex} {}
1312 \DeclareFontSeriesChangeRule {ux}{m?} {ux} {}
1313 \DeclareFontSeriesChangeRule {sbuc}{m?} {uc} {}
1314 \DeclareFontSeriesChangeRule {sbec}{m?} {ec} {}
1315 \DeclareFontSeriesChangeRule {sbc}{m?} {c} {}
1316 \DeclareFontSeriesChangeRule {sbsc}{m?} {sc} {}
1317 \DeclareFontSeriesChangeRule {sb}{m?} {m} {}
1318 \DeclareFontSeriesChangeRule {sbsx}{m?} {sx} {}
1319 \DeclareFontSeriesChangeRule {sbx}{m?} {x} {}
1320 \DeclareFontSeriesChangeRule {sbex}{m?} {ex} {}
1321 \DeclareFontSeriesChangeRule {sbux}{m?} {ux} {}
1322 \DeclareFontSeriesChangeRule {buc}{m?} {uc} {}
1323 \DeclareFontSeriesChangeRule {bec}{m?} {ec} {}
1324 \DeclareFontSeriesChangeRule {bc}{m?} {c} {}
1325 \DeclareFontSeriesChangeRule {bsc}{m?} {sc} {}
1326 \DeclareFontSeriesChangeRule {b}{m?} {m} {}
1327 \DeclareFontSeriesChangeRule {bsx}{m?} {sx} {}
1328 \DeclareFontSeriesChangeRule {bx}{m?} {x} {}
1329 \DeclareFontSeriesChangeRule {bex}{m?} {ex} {}
1330 \DeclareFontSeriesChangeRule {bux}{m?} {ux} {}
1331 \DeclareFontSeriesChangeRule {ebuc}{m?} {uc} {}
1332 \DeclareFontSeriesChangeRule {ebec}{m?} {ec} {}
1333 \DeclareFontSeriesChangeRule {ebc}{m?} {c} {}
1334 \DeclareFontSeriesChangeRule {ebsc}{m?} {sc} {}
1335 \DeclareFontSeriesChangeRule {eb}{m?} {m} {}
1336 \DeclareFontSeriesChangeRule {ebsx}{m?} {sx} {}


```

1337 \DeclareFontSeriesChangeRule {ebx}{m?} {x} {}
1338 \DeclareFontSeriesChangeRule {ebex}{m?} {ex} {}
1339 \DeclareFontSeriesChangeRule {ebux}{m?} {ux} {}
1340 \DeclareFontSeriesChangeRule {ubuc}{m?} {uc} {}
1341 \DeclareFontSeriesChangeRule {ubec}{m?} {ec} {}
1342 \DeclareFontSeriesChangeRule {ubc}{m?} {c} {}
1343 \DeclareFontSeriesChangeRule {ubsc}{m?} {sc} {}
1344 \DeclareFontSeriesChangeRule {ub}{m?} {m} {}
1345 \DeclareFontSeriesChangeRule {ubsx}{m?} {sx} {}
1346 \DeclareFontSeriesChangeRule {ubx}{m?} {x} {}
1347 \DeclareFontSeriesChangeRule {ubex}{m?} {ex} {}
1348 \DeclareFontSeriesChangeRule {ubux}{m?} {ux} {}

```

And there the special rules for ?m:

```

1349 \DeclareFontSeriesChangeRule {uluc}{?m} {ul} {}
1350 \DeclareFontSeriesChangeRule {ulec}{?m} {ul} {}
1351 \DeclareFontSeriesChangeRule {ulc}{?m} {ul} {}
1352 \DeclareFontSeriesChangeRule {ulsc}{?m} {ul} {}
1353 \DeclareFontSeriesChangeRule {ul}{?m} {ul} {}
1354 \DeclareFontSeriesChangeRule {ulsx}{?m} {ul} {}
1355 \DeclareFontSeriesChangeRule {ulx}{?m} {ul} {}
1356 \DeclareFontSeriesChangeRule {ulex}{?m} {ul} {}
1357 \DeclareFontSeriesChangeRule {ulux}{?m} {ul} {}
1358 \DeclareFontSeriesChangeRule {eluc}{?m} {el} {}
1359 \DeclareFontSeriesChangeRule {elec}{?m} {el} {}
1360 \DeclareFontSeriesChangeRule {elc}{?m} {el} {}
1361 \DeclareFontSeriesChangeRule {elsc}{?m} {el} {}
1362 \DeclareFontSeriesChangeRule {el}{?m} {el} {}
1363 \DeclareFontSeriesChangeRule {elsx}{?m} {el} {}
1364 \DeclareFontSeriesChangeRule {elx}{?m} {el} {}
1365 \DeclareFontSeriesChangeRule {elcx}{?m} {el} {}
1366 \DeclareFontSeriesChangeRule {elux}{?m} {el} {}
1367 \DeclareFontSeriesChangeRule {luc}{?m} {l} {}
1368 \DeclareFontSeriesChangeRule {lec}{?m} {l} {}
1369 \DeclareFontSeriesChangeRule {lc}{?m} {l} {}
1370 \DeclareFontSeriesChangeRule {lsc}{?m} {l} {}
1371 \DeclareFontSeriesChangeRule {l}{?m} {l} {}
1372 \DeclareFontSeriesChangeRule {lsx}{?m} {l} {}
1373 \DeclareFontSeriesChangeRule {lx}{?m} {l} {}
1374 \DeclareFontSeriesChangeRule {lex}{?m} {l} {}
1375 \DeclareFontSeriesChangeRule {lux}{?m} {l} {}
1376 \DeclareFontSeriesChangeRule {sluc}{?m} {sl} {}
1377 \DeclareFontSeriesChangeRule {slec}{?m} {sl} {}
1378 \DeclareFontSeriesChangeRule {slc}{?m} {sl} {}
1379 \DeclareFontSeriesChangeRule {slsc}{?m} {sl} {}
1380 \DeclareFontSeriesChangeRule {sl}{?m} {sl} {}
1381 \DeclareFontSeriesChangeRule {slsx}{?m} {sl} {}
1382 \DeclareFontSeriesChangeRule {slx}{?m} {sl} {}
1383 \DeclareFontSeriesChangeRule {slex}{?m} {sl} {}
1384 \DeclareFontSeriesChangeRule {slux}{?m} {sl} {}
1385 \DeclareFontSeriesChangeRule {uc}{?m} {m} {}
1386 \DeclareFontSeriesChangeRule {ec}{?m} {m} {}
1387 \DeclareFontSeriesChangeRule {c}{?m} {m} {}
1388 \DeclareFontSeriesChangeRule {sc}{?m} {m} {}
1389 \DeclareFontSeriesChangeRule {m}{?m} {m} {}

```

```

1390 \DeclareFontSeriesChangeRule {sx}{?m} {m} {}
1391 \DeclareFontSeriesChangeRule {x}{?m} {m} {}
1392 \DeclareFontSeriesChangeRule {ex}{?m} {m} {}
1393 \DeclareFontSeriesChangeRule {ux}{?m} {m} {}
1394 \DeclareFontSeriesChangeRule {sbuc}{?m} {sb} {}
1395 \DeclareFontSeriesChangeRule {sbec}{?m} {sb} {}
1396 \DeclareFontSeriesChangeRule {sbc}{?m} {sb} {}
1397 \DeclareFontSeriesChangeRule {sbsc}{?m} {sb} {}
1398 \DeclareFontSeriesChangeRule {sb}{?m} {sb} {}
1399 \DeclareFontSeriesChangeRule {sbsx}{?m} {sb} {}
1400 \DeclareFontSeriesChangeRule {sbx}{?m} {sb} {}
1401 \DeclareFontSeriesChangeRule {sbex}{?m} {sb} {}
1402 \DeclareFontSeriesChangeRule {sbux}{?m} {sb} {}
1403 \DeclareFontSeriesChangeRule {buc}{?m} {b} {}
1404 \DeclareFontSeriesChangeRule {bec}{?m} {b} {}
1405 \DeclareFontSeriesChangeRule {bc}{?m} {b} {}
1406 \DeclareFontSeriesChangeRule {bsc}{?m} {b} {}
1407 \DeclareFontSeriesChangeRule {b}{?m} {b} {}
1408 \DeclareFontSeriesChangeRule {bsx}{?m} {b} {}
1409 \DeclareFontSeriesChangeRule {bx}{?m} {b} {}
1410 \DeclareFontSeriesChangeRule {bex}{?m} {b} {}
1411 \DeclareFontSeriesChangeRule {bux}{?m} {b} {}
1412 \DeclareFontSeriesChangeRule {ebuc}{?m} {eb} {}
1413 \DeclareFontSeriesChangeRule {ebec}{?m} {eb} {}
1414 \DeclareFontSeriesChangeRule {ebc}{?m} {eb} {}
1415 \DeclareFontSeriesChangeRule {ebsc}{?m} {eb} {}
1416 \DeclareFontSeriesChangeRule {eb}{?m} {eb} {}
1417 \DeclareFontSeriesChangeRule {ebsx}{?m} {eb} {}
1418 \DeclareFontSeriesChangeRule {ebx}{?m} {eb} {}
1419 \DeclareFontSeriesChangeRule {ebex}{?m} {eb} {}
1420 \DeclareFontSeriesChangeRule {ebux}{?m} {eb} {}
1421 \DeclareFontSeriesChangeRule {ubuc}{?m} {ub} {}
1422 \DeclareFontSeriesChangeRule {ubec}{?m} {ub} {}
1423 \DeclareFontSeriesChangeRule {ubc}{?m} {ub} {}
1424 \DeclareFontSeriesChangeRule {ubsc}{?m} {ub} {}
1425 \DeclareFontSeriesChangeRule {ub}{?m} {ub} {}
1426 \DeclareFontSeriesChangeRule {ubsx}{?m} {ub} {}
1427 \DeclareFontSeriesChangeRule {ubx}{?m} {ub} {}
1428 \DeclareFontSeriesChangeRule {ubex}{?m} {ub} {}
1429 \DeclareFontSeriesChangeRule {ubux}{?m} {ub} {}

1430 \if2kernel | latexrelease)
1431 (latexrelease)\EndIncludeInRelease

Supporting rollback ...
1432 (latexrelease)\IncludeInRelease{2020/02/02}%
1433 (latexrelease) \DeclareFontSeriesChangeRule{Series change rules}%

The next definition is only needed if somebody rolls forward from a release older than
2020-02-02 but not to the latest version but one before 2025-06-01. Pretty unlikely, but
....
1434 (latexrelease)
1435 (latexrelease)\def\DeclareFontSeriesChangeRule#1#2#3#4{%
1436 (latexrelease) \@namedef{series@#1@#2}{{#3}{#4}}
1437 (latexrelease)

```

The huge set of declarations below are those from 2020-02-02 plus all from above that were newly added (but now with empty result and alternative result arguments). To compile this list I sorted both together and then dropped entries appearing twice. This is why the sorting is now different from the one above.

```

1438 <latexrelease>\DeclareFontSeriesChangeRule {bc}{bx} {} {}
1439 <latexrelease>\DeclareFontSeriesChangeRule {bc}{b}{bc}{}
1440 <latexrelease>\DeclareFontSeriesChangeRule {bc}{c}{bc}{}
1441 <latexrelease>\DeclareFontSeriesChangeRule {bc}{eb}{ebc}{}
1442 <latexrelease>\DeclareFontSeriesChangeRule {bc}{ec}{bec} {bc}
1443 <latexrelease>\DeclareFontSeriesChangeRule {bc}{el}{elc}{}
1444 <latexrelease>\DeclareFontSeriesChangeRule {bc}{ex} {} {}
1445 <latexrelease>\DeclareFontSeriesChangeRule {bc}{l}{lc}{}
1446 <latexrelease>\DeclareFontSeriesChangeRule {bc}{sb}{sbc}{}
1447 <latexrelease>\DeclareFontSeriesChangeRule {bc}{sc}{bsc} {bc}
1448 <latexrelease>\DeclareFontSeriesChangeRule {bc}{sl}{slc}{}
1449 <latexrelease>\DeclareFontSeriesChangeRule {bc}{sx} {} {}
1450 <latexrelease>\DeclareFontSeriesChangeRule {bc}{ub}{ubc}{}
1451 <latexrelease>\DeclareFontSeriesChangeRule {bc}{uc} {} {}
1452 <latexrelease>\DeclareFontSeriesChangeRule {bc}{ul}{ulc}{}
1453 <latexrelease>\DeclareFontSeriesChangeRule {bc}{ux} {} {}
1454 <latexrelease>\DeclareFontSeriesChangeRule {bc}{x}{bx}{}
1455 <latexrelease>\DeclareFontSeriesChangeRule {bec}{bx} {} {}
1456 <latexrelease>\DeclareFontSeriesChangeRule {bec}{b} {} {}
1457 <latexrelease>\DeclareFontSeriesChangeRule {bec}{c} {} {}
1458 <latexrelease>\DeclareFontSeriesChangeRule {bec}{eb} {} {}
1459 <latexrelease>\DeclareFontSeriesChangeRule {bec}{ec} {} {}
1460 <latexrelease>\DeclareFontSeriesChangeRule {bec}{el} {} {}
1461 <latexrelease>\DeclareFontSeriesChangeRule {bec}{ex} {} {}
1462 <latexrelease>\DeclareFontSeriesChangeRule {bec}{l} {} {}
1463 <latexrelease>\DeclareFontSeriesChangeRule {bec}{sb} {} {}
1464 <latexrelease>\DeclareFontSeriesChangeRule {bec}{sc} {} {}
1465 <latexrelease>\DeclareFontSeriesChangeRule {bec}{sl} {} {}
1466 <latexrelease>\DeclareFontSeriesChangeRule {bec}{sx} {} {}
1467 <latexrelease>\DeclareFontSeriesChangeRule {bec}{ub} {} {}
1468 <latexrelease>\DeclareFontSeriesChangeRule {bec}{uc} {} {}
1469 <latexrelease>\DeclareFontSeriesChangeRule {bec}{ul} {} {}
1470 <latexrelease>\DeclareFontSeriesChangeRule {bec}{ux} {} {}
1471 <latexrelease>\DeclareFontSeriesChangeRule {bec}{x} {} {}
1472 <latexrelease>\DeclareFontSeriesChangeRule {bex}{?m} {} {}
1473 <latexrelease>\DeclareFontSeriesChangeRule {bex}{bx} {} {}
1474 <latexrelease>\DeclareFontSeriesChangeRule {bex}{b} {} {}
1475 <latexrelease>\DeclareFontSeriesChangeRule {bex}{c} {} {}
1476 <latexrelease>\DeclareFontSeriesChangeRule {bex}{eb} {} {}
1477 <latexrelease>\DeclareFontSeriesChangeRule {bex}{ec} {} {}
1478 <latexrelease>\DeclareFontSeriesChangeRule {bex}{el} {} {}
1479 <latexrelease>\DeclareFontSeriesChangeRule {bex}{ex} {} {}
1480 <latexrelease>\DeclareFontSeriesChangeRule {bex}{l} {} {}
1481 <latexrelease>\DeclareFontSeriesChangeRule {bex}{m?} {} {}
1482 <latexrelease>\DeclareFontSeriesChangeRule {bex}{sb} {} {}
1483 <latexrelease>\DeclareFontSeriesChangeRule {bex}{sc} {} {}
1484 <latexrelease>\DeclareFontSeriesChangeRule {bex}{sl} {} {}
1485 <latexrelease>\DeclareFontSeriesChangeRule {bex}{sx} {} {}
1486 <latexrelease>\DeclareFontSeriesChangeRule {bex}{ub} {} {}
1487 <latexrelease>\DeclareFontSeriesChangeRule {bex}{uc} {} {}

```

```

1488 <latexrelease>\DeclareFontSeriesChangeRule {bex}{ul} {} {}
1489 <latexrelease>\DeclareFontSeriesChangeRule {bex}{ux} {} {}
1490 <latexrelease>\DeclareFontSeriesChangeRule {bex}{x} {} {}
1491 <latexrelease>\DeclareFontSeriesChangeRule {bsc}{bx} {} {}
1492 <latexrelease>\DeclareFontSeriesChangeRule {bsc}{b} {} {}
1493 <latexrelease>\DeclareFontSeriesChangeRule {bsc}{c} {} {}
1494 <latexrelease>\DeclareFontSeriesChangeRule {bsc}{eb} {} {}
1495 <latexrelease>\DeclareFontSeriesChangeRule {bsc}{ec} {} {}
1496 <latexrelease>\DeclareFontSeriesChangeRule {bsc}{el} {} {}
1497 <latexrelease>\DeclareFontSeriesChangeRule {bsc}{ex} {} {}
1498 <latexrelease>\DeclareFontSeriesChangeRule {bsc}{l} {} {}
1499 <latexrelease>\DeclareFontSeriesChangeRule {bsc}{sb} {} {}
1500 <latexrelease>\DeclareFontSeriesChangeRule {bsc}{sc} {} {}
1501 <latexrelease>\DeclareFontSeriesChangeRule {bsc}{sl} {} {}
1502 <latexrelease>\DeclareFontSeriesChangeRule {bsc}{sx} {} {}
1503 <latexrelease>\DeclareFontSeriesChangeRule {bsc}{ub} {} {}
1504 <latexrelease>\DeclareFontSeriesChangeRule {bsc}{uc} {} {}
1505 <latexrelease>\DeclareFontSeriesChangeRule {bsc}{ul} {} {}
1506 <latexrelease>\DeclareFontSeriesChangeRule {bsc}{ux} {} {}
1507 <latexrelease>\DeclareFontSeriesChangeRule {bsc}{x} {} {}
1508 <latexrelease>\DeclareFontSeriesChangeRule {bsx}{?m} {} {}
1509 <latexrelease>\DeclareFontSeriesChangeRule {bsx}{bx} {} {}
1510 <latexrelease>\DeclareFontSeriesChangeRule {bsx}{b} {} {}
1511 <latexrelease>\DeclareFontSeriesChangeRule {bsx}{c} {} {}
1512 <latexrelease>\DeclareFontSeriesChangeRule {bsx}{eb} {} {}
1513 <latexrelease>\DeclareFontSeriesChangeRule {bsx}{ec} {} {}
1514 <latexrelease>\DeclareFontSeriesChangeRule {bsx}{el} {} {}
1515 <latexrelease>\DeclareFontSeriesChangeRule {bsx}{ex} {} {}
1516 <latexrelease>\DeclareFontSeriesChangeRule {bsx}{l} {} {}
1517 <latexrelease>\DeclareFontSeriesChangeRule {bsx}{m?} {} {}
1518 <latexrelease>\DeclareFontSeriesChangeRule {bsx}{sb} {} {}
1519 <latexrelease>\DeclareFontSeriesChangeRule {bsx}{sc} {} {}
1520 <latexrelease>\DeclareFontSeriesChangeRule {bsx}{sl} {} {}
1521 <latexrelease>\DeclareFontSeriesChangeRule {bsx}{sx} {} {}
1522 <latexrelease>\DeclareFontSeriesChangeRule {bsx}{ub} {} {}
1523 <latexrelease>\DeclareFontSeriesChangeRule {bsx}{uc} {} {}
1524 <latexrelease>\DeclareFontSeriesChangeRule {bsx}{ul} {} {}
1525 <latexrelease>\DeclareFontSeriesChangeRule {bsx}{ux} {} {}
1526 <latexrelease>\DeclareFontSeriesChangeRule {bsx}{x} {} {}
1527 <latexrelease>\DeclareFontSeriesChangeRule {buc}{?m} {} {}
1528 <latexrelease>\DeclareFontSeriesChangeRule {buc}{bx} {} {}
1529 <latexrelease>\DeclareFontSeriesChangeRule {buc}{b} {} {}
1530 <latexrelease>\DeclareFontSeriesChangeRule {buc}{c} {} {}
1531 <latexrelease>\DeclareFontSeriesChangeRule {buc}{eb} {} {}
1532 <latexrelease>\DeclareFontSeriesChangeRule {buc}{ec} {} {}
1533 <latexrelease>\DeclareFontSeriesChangeRule {buc}{el} {} {}
1534 <latexrelease>\DeclareFontSeriesChangeRule {buc}{ex} {} {}
1535 <latexrelease>\DeclareFontSeriesChangeRule {buc}{l} {} {}
1536 <latexrelease>\DeclareFontSeriesChangeRule {buc}{m?} {} {}
1537 <latexrelease>\DeclareFontSeriesChangeRule {buc}{sb} {} {}
1538 <latexrelease>\DeclareFontSeriesChangeRule {buc}{sc} {} {}
1539 <latexrelease>\DeclareFontSeriesChangeRule {buc}{sl} {} {}
1540 <latexrelease>\DeclareFontSeriesChangeRule {buc}{sx} {} {}
1541 <latexrelease>\DeclareFontSeriesChangeRule {buc}{ub} {} {}

```

```

1542 <latexrelease>\DeclareFontSeriesChangeRule {buc}{uc} {} {}
1543 <latexrelease>\DeclareFontSeriesChangeRule {buc}{ul} {} {}
1544 <latexrelease>\DeclareFontSeriesChangeRule {buc}{ux} {} {}
1545 <latexrelease>\DeclareFontSeriesChangeRule {buc}{x} {} {}
1546 <latexrelease>\DeclareFontSeriesChangeRule {bux}{?m} {} {}
1547 <latexrelease>\DeclareFontSeriesChangeRule {bux}{bx} {} {}
1548 <latexrelease>\DeclareFontSeriesChangeRule {bux}{b} {} {}
1549 <latexrelease>\DeclareFontSeriesChangeRule {bux}{c} {} {}
1550 <latexrelease>\DeclareFontSeriesChangeRule {bux}{eb} {} {}
1551 <latexrelease>\DeclareFontSeriesChangeRule {bux}{ec} {} {}
1552 <latexrelease>\DeclareFontSeriesChangeRule {bux}{el} {} {}
1553 <latexrelease>\DeclareFontSeriesChangeRule {bux}{ex} {} {}
1554 <latexrelease>\DeclareFontSeriesChangeRule {bux}{l} {} {}
1555 <latexrelease>\DeclareFontSeriesChangeRule {bux}{m?} {} {}
1556 <latexrelease>\DeclareFontSeriesChangeRule {bux}{sb} {} {}
1557 <latexrelease>\DeclareFontSeriesChangeRule {bux}{sc} {} {}
1558 <latexrelease>\DeclareFontSeriesChangeRule {bux}{sl} {} {}
1559 <latexrelease>\DeclareFontSeriesChangeRule {bux}{sx} {} {}
1560 <latexrelease>\DeclareFontSeriesChangeRule {bux}{ub} {} {}
1561 <latexrelease>\DeclareFontSeriesChangeRule {bux}{uc} {} {}
1562 <latexrelease>\DeclareFontSeriesChangeRule {bux}{ul} {} {}
1563 <latexrelease>\DeclareFontSeriesChangeRule {bux}{ux} {} {}
1564 <latexrelease>\DeclareFontSeriesChangeRule {bux}{x} {} {}
1565 <latexrelease>\DeclareFontSeriesChangeRule {bx}{bx} {} {}
1566 <latexrelease>\DeclareFontSeriesChangeRule {bx}{b}{bx}{}
1567 <latexrelease>\DeclareFontSeriesChangeRule {bx}{c} {bc} {bx}
1568 <latexrelease>\DeclareFontSeriesChangeRule {bx}{eb}{ebx}{}
1569 <latexrelease>\DeclareFontSeriesChangeRule {bx}{ec} {bec} {bx}
1570 <latexrelease>\DeclareFontSeriesChangeRule {bx}{el}{elx}{}
1571 <latexrelease>\DeclareFontSeriesChangeRule {bx}{ex} {} {}
1572 <latexrelease>\DeclareFontSeriesChangeRule {bx}{l}{lx}{}
1573 <latexrelease>\DeclareFontSeriesChangeRule {bx}{sb} {sbx} {}
1574 <latexrelease>\DeclareFontSeriesChangeRule {bx}{sc} {bsc} {bx}
1575 <latexrelease>\DeclareFontSeriesChangeRule {bx}{sl}{slx} {}
1576 <latexrelease>\DeclareFontSeriesChangeRule {bx}{sx} {} {}
1577 <latexrelease>\DeclareFontSeriesChangeRule {bx}{ub}{ubx}{}
1578 <latexrelease>\DeclareFontSeriesChangeRule {bx}{uc} {} {}
1579 <latexrelease>\DeclareFontSeriesChangeRule {bx}{ul}{ulx}{}
1580 <latexrelease>\DeclareFontSeriesChangeRule {bx}{ux} {} {}
1581 <latexrelease>\DeclareFontSeriesChangeRule {bx}{x}{bx}{}
1582 <latexrelease>\DeclareFontSeriesChangeRule {b}{bx} {bx} {b}
1583 <latexrelease>\DeclareFontSeriesChangeRule {b}{c} {bc} {b}
1584 <latexrelease>\DeclareFontSeriesChangeRule {b}{eb} {} {}
1585 <latexrelease>\DeclareFontSeriesChangeRule {b}{ec} {bec} {b}
1586 <latexrelease>\DeclareFontSeriesChangeRule {b}{ex} {} {}
1587 <latexrelease>\DeclareFontSeriesChangeRule {b}{sb} {sb} {b}
1588 <latexrelease>\DeclareFontSeriesChangeRule {b}{sc} {bsc} {b}
1589 <latexrelease>\DeclareFontSeriesChangeRule {b}{sx} {} {}
1590 <latexrelease>\DeclareFontSeriesChangeRule {b}{ub} {} {}
1591 <latexrelease>\DeclareFontSeriesChangeRule {b}{uc} {} {}
1592 <latexrelease>\DeclareFontSeriesChangeRule {b}{ux} {} {}
1593 <latexrelease>\DeclareFontSeriesChangeRule {b}{x} {bx} {b}
1594 <latexrelease>\DeclareFontSeriesChangeRule {c}{bx} {bx} {b}
1595 <latexrelease>\DeclareFontSeriesChangeRule {c}{b}{bc}{}

```

```

1596 <latexrelease>\DeclareFontSeriesChangeRule {c}{eb}{ebc}{-}
1597 <latexrelease>\DeclareFontSeriesChangeRule {c}{ec} {} {}
1598 <latexrelease>\DeclareFontSeriesChangeRule {c}{el}{elc}{-}
1599 <latexrelease>\DeclareFontSeriesChangeRule {c}{l}{lc}{-}
1600 <latexrelease>\DeclareFontSeriesChangeRule {c}{sb}{sbc}{-}
1601 <latexrelease>\DeclareFontSeriesChangeRule {c}{sc} {} {}
1602 <latexrelease>\DeclareFontSeriesChangeRule {c}{sl}{slc}{-}
1603 <latexrelease>\DeclareFontSeriesChangeRule {c}{ub}{ubc}{-}
1604 <latexrelease>\DeclareFontSeriesChangeRule {c}{uc} {} {}
1605 <latexrelease>\DeclareFontSeriesChangeRule {c}{x}{x}{m}
1606 <latexrelease>\DeclareFontSeriesChangeRule {ebc}{bx} {} {}
1607 <latexrelease>\DeclareFontSeriesChangeRule {ebc}{b}{bc}{-}
1608 <latexrelease>\DeclareFontSeriesChangeRule {ebc}{c}{ebc}{-}
1609 <latexrelease>\DeclareFontSeriesChangeRule {ebc}{eb}{ebc}{-}
1610 <latexrelease>\DeclareFontSeriesChangeRule {ebc}{ec}{ebec}{ebc}
1611 <latexrelease>\DeclareFontSeriesChangeRule {ebc}{el}{elc}{-}
1612 <latexrelease>\DeclareFontSeriesChangeRule {ebc}{ex} {} {}
1613 <latexrelease>\DeclareFontSeriesChangeRule {ebc}{l}{lc}{-}
1614 <latexrelease>\DeclareFontSeriesChangeRule {ebc}{sb}{sbc}{-}
1615 <latexrelease>\DeclareFontSeriesChangeRule {ebc}{sc}{ebsc}{ebc}
1616 <latexrelease>\DeclareFontSeriesChangeRule {ebc}{sl}{slc}{-}
1617 <latexrelease>\DeclareFontSeriesChangeRule {ebc}{sx} {} {}
1618 <latexrelease>\DeclareFontSeriesChangeRule {ebc}{ub}{ubc}{-}
1619 <latexrelease>\DeclareFontSeriesChangeRule {ebc}{uc} {} {}
1620 <latexrelease>\DeclareFontSeriesChangeRule {ebc}{ul}{ulc}{-}
1621 <latexrelease>\DeclareFontSeriesChangeRule {ebc}{ux} {} {}
1622 <latexrelease>\DeclareFontSeriesChangeRule {ebc}{x}{ebx}{-}
1623 <latexrelease>\DeclareFontSeriesChangeRule {ebec}{bx} {} {}
1624 <latexrelease>\DeclareFontSeriesChangeRule {ebec}{b} {} {}
1625 <latexrelease>\DeclareFontSeriesChangeRule {ebec}{c} {} {}
1626 <latexrelease>\DeclareFontSeriesChangeRule {ebec}{eb} {} {}
1627 <latexrelease>\DeclareFontSeriesChangeRule {ebec}{ec} {} {}
1628 <latexrelease>\DeclareFontSeriesChangeRule {ebec}{el} {} {}
1629 <latexrelease>\DeclareFontSeriesChangeRule {ebec}{ex} {} {}
1630 <latexrelease>\DeclareFontSeriesChangeRule {ebec}{l} {} {}
1631 <latexrelease>\DeclareFontSeriesChangeRule {ebec}{sb} {} {}
1632 <latexrelease>\DeclareFontSeriesChangeRule {ebec}{sc} {} {}
1633 <latexrelease>\DeclareFontSeriesChangeRule {ebec}{sl} {} {}
1634 <latexrelease>\DeclareFontSeriesChangeRule {ebec}{sx} {} {}
1635 <latexrelease>\DeclareFontSeriesChangeRule {ebec}{ub} {} {}
1636 <latexrelease>\DeclareFontSeriesChangeRule {ebec}{uc} {} {}
1637 <latexrelease>\DeclareFontSeriesChangeRule {ebec}{ul} {} {}
1638 <latexrelease>\DeclareFontSeriesChangeRule {ebec}{ux} {} {}
1639 <latexrelease>\DeclareFontSeriesChangeRule {ebec}{x} {} {}
1640 <latexrelease>\DeclareFontSeriesChangeRule {ebex}{?m} {} {}
1641 <latexrelease>\DeclareFontSeriesChangeRule {ebex}{bx} {} {}
1642 <latexrelease>\DeclareFontSeriesChangeRule {ebex}{b} {} {}
1643 <latexrelease>\DeclareFontSeriesChangeRule {ebex}{c} {} {}
1644 <latexrelease>\DeclareFontSeriesChangeRule {ebex}{eb} {} {}
1645 <latexrelease>\DeclareFontSeriesChangeRule {ebex}{ec} {} {}
1646 <latexrelease>\DeclareFontSeriesChangeRule {ebex}{el} {} {}
1647 <latexrelease>\DeclareFontSeriesChangeRule {ebex}{ex} {} {}
1648 <latexrelease>\DeclareFontSeriesChangeRule {ebex}{l} {} {}
1649 <latexrelease>\DeclareFontSeriesChangeRule {ebex}{m?} {} {}

```

```

1650 <latexrelease>\DeclareFontSeriesChangeRule {ebex}{sb} {} {}
1651 <latexrelease>\DeclareFontSeriesChangeRule {ebex}{sc} {} {}
1652 <latexrelease>\DeclareFontSeriesChangeRule {ebex}{sl} {} {}
1653 <latexrelease>\DeclareFontSeriesChangeRule {ebex}{sx} {} {}
1654 <latexrelease>\DeclareFontSeriesChangeRule {ebex}{ub} {} {}
1655 <latexrelease>\DeclareFontSeriesChangeRule {ebex}{uc} {} {}
1656 <latexrelease>\DeclareFontSeriesChangeRule {ebex}{ul} {} {}
1657 <latexrelease>\DeclareFontSeriesChangeRule {ebex}{ux} {} {}
1658 <latexrelease>\DeclareFontSeriesChangeRule {ebex}{x} {} {}
1659 <latexrelease>\DeclareFontSeriesChangeRule {ebsc}{bx} {} {}
1660 <latexrelease>\DeclareFontSeriesChangeRule {ebsc}{b} {} {}
1661 <latexrelease>\DeclareFontSeriesChangeRule {ebsc}{c} {} {}
1662 <latexrelease>\DeclareFontSeriesChangeRule {ebsc}{eb} {} {}
1663 <latexrelease>\DeclareFontSeriesChangeRule {ebsc}{ec} {} {}
1664 <latexrelease>\DeclareFontSeriesChangeRule {ebsc}{el} {} {}
1665 <latexrelease>\DeclareFontSeriesChangeRule {ebsc}{ex} {} {}
1666 <latexrelease>\DeclareFontSeriesChangeRule {ebsc}{l} {} {}
1667 <latexrelease>\DeclareFontSeriesChangeRule {ebsc}{sb} {} {}
1668 <latexrelease>\DeclareFontSeriesChangeRule {ebsc}{sc} {} {}
1669 <latexrelease>\DeclareFontSeriesChangeRule {ebsc}{sl} {} {}
1670 <latexrelease>\DeclareFontSeriesChangeRule {ebsc}{sx} {} {}
1671 <latexrelease>\DeclareFontSeriesChangeRule {ebsc}{ub} {} {}
1672 <latexrelease>\DeclareFontSeriesChangeRule {ebsc}{uc} {} {}
1673 <latexrelease>\DeclareFontSeriesChangeRule {ebsc}{ul} {} {}
1674 <latexrelease>\DeclareFontSeriesChangeRule {ebsc}{ux} {} {}
1675 <latexrelease>\DeclareFontSeriesChangeRule {ebsc}{x} {} {}
1676 <latexrelease>\DeclareFontSeriesChangeRule {ebsx}{?m} {} {}
1677 <latexrelease>\DeclareFontSeriesChangeRule {ebsx}{bx} {} {}
1678 <latexrelease>\DeclareFontSeriesChangeRule {ebsx}{b} {} {}
1679 <latexrelease>\DeclareFontSeriesChangeRule {ebsx}{c} {} {}
1680 <latexrelease>\DeclareFontSeriesChangeRule {ebsx}{eb} {} {}
1681 <latexrelease>\DeclareFontSeriesChangeRule {ebsx}{ec} {} {}
1682 <latexrelease>\DeclareFontSeriesChangeRule {ebsx}{el} {} {}
1683 <latexrelease>\DeclareFontSeriesChangeRule {ebsx}{ex} {} {}
1684 <latexrelease>\DeclareFontSeriesChangeRule {ebsx}{l} {} {}
1685 <latexrelease>\DeclareFontSeriesChangeRule {ebsx}{m?} {} {}
1686 <latexrelease>\DeclareFontSeriesChangeRule {ebsx}{sb} {} {}
1687 <latexrelease>\DeclareFontSeriesChangeRule {ebsx}{sc} {} {}
1688 <latexrelease>\DeclareFontSeriesChangeRule {ebsx}{sl} {} {}
1689 <latexrelease>\DeclareFontSeriesChangeRule {ebsx}{sx} {} {}
1690 <latexrelease>\DeclareFontSeriesChangeRule {ebsx}{ub} {} {}
1691 <latexrelease>\DeclareFontSeriesChangeRule {ebsx}{uc} {} {}
1692 <latexrelease>\DeclareFontSeriesChangeRule {ebsx}{ul} {} {}
1693 <latexrelease>\DeclareFontSeriesChangeRule {ebsx}{ux} {} {}
1694 <latexrelease>\DeclareFontSeriesChangeRule {ebsx}{x} {} {}
1695 <latexrelease>\DeclareFontSeriesChangeRule {ebuc}{?m} {} {}
1696 <latexrelease>\DeclareFontSeriesChangeRule {ebuc}{bx} {} {}
1697 <latexrelease>\DeclareFontSeriesChangeRule {ebuc}{b} {} {}
1698 <latexrelease>\DeclareFontSeriesChangeRule {ebuc}{c} {} {}
1699 <latexrelease>\DeclareFontSeriesChangeRule {ebuc}{eb} {} {}
1700 <latexrelease>\DeclareFontSeriesChangeRule {ebuc}{ec} {} {}
1701 <latexrelease>\DeclareFontSeriesChangeRule {ebuc}{el} {} {}
1702 <latexrelease>\DeclareFontSeriesChangeRule {ebuc}{ex} {} {}
1703 <latexrelease>\DeclareFontSeriesChangeRule {ebuc}{l} {} {}

```

```

1704 <latexrelease>\DeclareFontSeriesChangeRule {ebuc}{m?} {} {}
1705 <latexrelease>\DeclareFontSeriesChangeRule {ebuc}{sb} {} {}
1706 <latexrelease>\DeclareFontSeriesChangeRule {ebuc}{sc} {} {}
1707 <latexrelease>\DeclareFontSeriesChangeRule {ebuc}{sl} {} {}
1708 <latexrelease>\DeclareFontSeriesChangeRule {ebuc}{sx} {} {}
1709 <latexrelease>\DeclareFontSeriesChangeRule {ebuc}{ub} {} {}
1710 <latexrelease>\DeclareFontSeriesChangeRule {ebuc}{uc} {} {}
1711 <latexrelease>\DeclareFontSeriesChangeRule {ebuc}{ul} {} {}
1712 <latexrelease>\DeclareFontSeriesChangeRule {ebuc}{ux} {} {}
1713 <latexrelease>\DeclareFontSeriesChangeRule {ebuc}{x} {} {}
1714 <latexrelease>\DeclareFontSeriesChangeRule {ebux}{?m} {} {}
1715 <latexrelease>\DeclareFontSeriesChangeRule {ebux}{bx} {} {}
1716 <latexrelease>\DeclareFontSeriesChangeRule {ebux}{b} {} {}
1717 <latexrelease>\DeclareFontSeriesChangeRule {ebux}{c} {} {}
1718 <latexrelease>\DeclareFontSeriesChangeRule {ebux}{eb} {} {}
1719 <latexrelease>\DeclareFontSeriesChangeRule {ebux}{ec} {} {}
1720 <latexrelease>\DeclareFontSeriesChangeRule {ebux}{el} {} {}
1721 <latexrelease>\DeclareFontSeriesChangeRule {ebux}{ex} {} {}
1722 <latexrelease>\DeclareFontSeriesChangeRule {ebux}{l} {} {}
1723 <latexrelease>\DeclareFontSeriesChangeRule {ebux}{m?} {} {}
1724 <latexrelease>\DeclareFontSeriesChangeRule {ebux}{sb} {} {}
1725 <latexrelease>\DeclareFontSeriesChangeRule {ebux}{sc} {} {}
1726 <latexrelease>\DeclareFontSeriesChangeRule {ebux}{sl} {} {}
1727 <latexrelease>\DeclareFontSeriesChangeRule {ebux}{sx} {} {}
1728 <latexrelease>\DeclareFontSeriesChangeRule {ebux}{ub} {} {}
1729 <latexrelease>\DeclareFontSeriesChangeRule {ebux}{uc} {} {}
1730 <latexrelease>\DeclareFontSeriesChangeRule {ebux}{ul} {} {}
1731 <latexrelease>\DeclareFontSeriesChangeRule {ebux}{ux} {} {}
1732 <latexrelease>\DeclareFontSeriesChangeRule {ebux}{x} {} {}
1733 <latexrelease>\DeclareFontSeriesChangeRule {ebx}{bx} {} {}
1734 <latexrelease>\DeclareFontSeriesChangeRule {ebx}{b}{bx}{}
1735 <latexrelease>\DeclareFontSeriesChangeRule {ebx}{c}{ebc}{}
1736 <latexrelease>\DeclareFontSeriesChangeRule {ebx}{eb}{ebx}{}
1737 <latexrelease>\DeclareFontSeriesChangeRule {ebx}{ec}{ebec}{}
1738 <latexrelease>\DeclareFontSeriesChangeRule {ebx}{el}{elx}{}
1739 <latexrelease>\DeclareFontSeriesChangeRule {ebx}{ex} {} {}
1740 <latexrelease>\DeclareFontSeriesChangeRule {ebx}{l}{lx}{}
1741 <latexrelease>\DeclareFontSeriesChangeRule {ebx}{sb}{sbx}{}
1742 <latexrelease>\DeclareFontSeriesChangeRule {ebx}{sc}{ebsc}{}
1743 <latexrelease>\DeclareFontSeriesChangeRule {ebx}{sl}{slx}{}
1744 <latexrelease>\DeclareFontSeriesChangeRule {ebx}{sx} {} {}
1745 <latexrelease>\DeclareFontSeriesChangeRule {ebx}{ub}{ubx}{}
1746 <latexrelease>\DeclareFontSeriesChangeRule {ebx}{uc} {} {}
1747 <latexrelease>\DeclareFontSeriesChangeRule {ebx}{ul}{ulx}{}
1748 <latexrelease>\DeclareFontSeriesChangeRule {ebx}{ux} {} {}
1749 <latexrelease>\DeclareFontSeriesChangeRule {ebx}{x}{ebx}{}
1750 <latexrelease>\DeclareFontSeriesChangeRule {eb}{bx} {} {}
1751 <latexrelease>\DeclareFontSeriesChangeRule {eb}{c}{ebc}{}
1752 <latexrelease>\DeclareFontSeriesChangeRule {eb}{ec}{ebec}{}
1753 <latexrelease>\DeclareFontSeriesChangeRule {eb}{ex} {} {}
1754 <latexrelease>\DeclareFontSeriesChangeRule {eb}{sb} {} {}
1755 <latexrelease>\DeclareFontSeriesChangeRule {eb}{sc}{ebsc}{}
1756 <latexrelease>\DeclareFontSeriesChangeRule {eb}{sx} {} {}
1757 <latexrelease>\DeclareFontSeriesChangeRule {eb}{ub} {} {}

```



```

1758 <latexrelease>\DeclareFontSeriesChangeRule {eb}{uc} {} {}
1759 <latexrelease>\DeclareFontSeriesChangeRule {eb}{ux} {} {}
1760 <latexrelease>\DeclareFontSeriesChangeRule {eb}{x}{ebx}{x}
1761 <latexrelease>\DeclareFontSeriesChangeRule {ec}{bx}{bx} {b}
1762 <latexrelease>\DeclareFontSeriesChangeRule {ec}{b}{bec}{x}
1763 <latexrelease>\DeclareFontSeriesChangeRule {ec}{eb}{ebec}{x}
1764 <latexrelease>\DeclareFontSeriesChangeRule {ec}{el}{elec}{x}
1765 <latexrelease>\DeclareFontSeriesChangeRule {ec}{l}{lec}{x}
1766 <latexrelease>\DeclareFontSeriesChangeRule {ec}{sb}{sbec}{x}
1767 <latexrelease>\DeclareFontSeriesChangeRule {ec}{sl}{slec}{x}
1768 <latexrelease>\DeclareFontSeriesChangeRule {ec}{ub}{ubec}{x}
1769 <latexrelease>\DeclareFontSeriesChangeRule {ec}{x}{x}{m}
1770 <latexrelease>\DeclareFontSeriesChangeRule {elc}{bx} {} {}
1771 <latexrelease>\DeclareFontSeriesChangeRule {elc}{b}{bc}{x}
1772 <latexrelease>\DeclareFontSeriesChangeRule {elc}{c}{elc}{x}
1773 <latexrelease>\DeclareFontSeriesChangeRule {elc}{eb}{ebc}{x}
1774 <latexrelease>\DeclareFontSeriesChangeRule {elc}{ec}{elec}{x}
1775 <latexrelease>\DeclareFontSeriesChangeRule {elc}{el}{elc}{x}
1776 <latexrelease>\DeclareFontSeriesChangeRule {elc}{ex} {} {}
1777 <latexrelease>\DeclareFontSeriesChangeRule {elc}{l}{lc}{x}
1778 <latexrelease>\DeclareFontSeriesChangeRule {elc}{sb}{sbc}{x}
1779 <latexrelease>\DeclareFontSeriesChangeRule {elc}{sc}{elsc}{x}
1780 <latexrelease>\DeclareFontSeriesChangeRule {elc}{sl}{slc}{x}
1781 <latexrelease>\DeclareFontSeriesChangeRule {elc}{sx} {} {}
1782 <latexrelease>\DeclareFontSeriesChangeRule {elc}{ub}{ubc}{x}
1783 <latexrelease>\DeclareFontSeriesChangeRule {elc}{uc} {} {}
1784 <latexrelease>\DeclareFontSeriesChangeRule {elc}{ul}{ulc}{x}
1785 <latexrelease>\DeclareFontSeriesChangeRule {elc}{ux} {} {}
1786 <latexrelease>\DeclareFontSeriesChangeRule {elc}{x}{elx}{x}
1787 <latexrelease>\DeclareFontSeriesChangeRule {elec}{bx} {} {}
1788 <latexrelease>\DeclareFontSeriesChangeRule {elec}{b} {} {}
1789 <latexrelease>\DeclareFontSeriesChangeRule {elec}{c} {} {}
1790 <latexrelease>\DeclareFontSeriesChangeRule {elec}{eb} {} {}
1791 <latexrelease>\DeclareFontSeriesChangeRule {elec}{ec} {} {}
1792 <latexrelease>\DeclareFontSeriesChangeRule {elec}{el} {} {}
1793 <latexrelease>\DeclareFontSeriesChangeRule {elec}{ex} {} {}
1794 <latexrelease>\DeclareFontSeriesChangeRule {elec}{l} {} {}
1795 <latexrelease>\DeclareFontSeriesChangeRule {elec}{sb} {} {}
1796 <latexrelease>\DeclareFontSeriesChangeRule {elec}{sc} {} {}
1797 <latexrelease>\DeclareFontSeriesChangeRule {elec}{sl} {} {}
1798 <latexrelease>\DeclareFontSeriesChangeRule {elec}{sx} {} {}
1799 <latexrelease>\DeclareFontSeriesChangeRule {elec}{ub} {} {}
1800 <latexrelease>\DeclareFontSeriesChangeRule {elec}{uc} {} {}
1801 <latexrelease>\DeclareFontSeriesChangeRule {elec}{ul} {} {}
1802 <latexrelease>\DeclareFontSeriesChangeRule {elec}{ux} {} {}
1803 <latexrelease>\DeclareFontSeriesChangeRule {elec}{x} {} {}
1804 <latexrelease>\DeclareFontSeriesChangeRule {elex}{?m} {} {}
1805 <latexrelease>\DeclareFontSeriesChangeRule {elex}{bx} {} {}
1806 <latexrelease>\DeclareFontSeriesChangeRule {elex}{b} {} {}
1807 <latexrelease>\DeclareFontSeriesChangeRule {elex}{c} {} {}
1808 <latexrelease>\DeclareFontSeriesChangeRule {elex}{eb} {} {}
1809 <latexrelease>\DeclareFontSeriesChangeRule {elex}{ec} {} {}
1810 <latexrelease>\DeclareFontSeriesChangeRule {elex}{el} {} {}
1811 <latexrelease>\DeclareFontSeriesChangeRule {elex}{ex} {} {}

```

```

1812 <latexrelease>\DeclareFontSeriesChangeRule {elx}{l} {} {}
1813 <latexrelease>\DeclareFontSeriesChangeRule {elx}{m?} {} {}
1814 <latexrelease>\DeclareFontSeriesChangeRule {elx}{sb} {} {}
1815 <latexrelease>\DeclareFontSeriesChangeRule {elx}{sc} {} {}
1816 <latexrelease>\DeclareFontSeriesChangeRule {elx}{sl} {} {}
1817 <latexrelease>\DeclareFontSeriesChangeRule {elx}{sx} {} {}
1818 <latexrelease>\DeclareFontSeriesChangeRule {elx}{ub} {} {}
1819 <latexrelease>\DeclareFontSeriesChangeRule {elx}{uc} {} {}
1820 <latexrelease>\DeclareFontSeriesChangeRule {elx}{ul} {} {}
1821 <latexrelease>\DeclareFontSeriesChangeRule {elx}{ux} {} {}
1822 <latexrelease>\DeclareFontSeriesChangeRule {elx}{x} {} {}
1823 <latexrelease>\DeclareFontSeriesChangeRule {elsc}{bx} {} {}
1824 <latexrelease>\DeclareFontSeriesChangeRule {elsc}{b} {} {}
1825 <latexrelease>\DeclareFontSeriesChangeRule {elsc}{c} {} {}
1826 <latexrelease>\DeclareFontSeriesChangeRule {elsc}{eb} {} {}
1827 <latexrelease>\DeclareFontSeriesChangeRule {elsc}{ec} {} {}
1828 <latexrelease>\DeclareFontSeriesChangeRule {elsc}{el} {} {}
1829 <latexrelease>\DeclareFontSeriesChangeRule {elsc}{ex} {} {}
1830 <latexrelease>\DeclareFontSeriesChangeRule {elsc}{l} {} {}
1831 <latexrelease>\DeclareFontSeriesChangeRule {elsc}{sb} {} {}
1832 <latexrelease>\DeclareFontSeriesChangeRule {elsc}{sc} {} {}
1833 <latexrelease>\DeclareFontSeriesChangeRule {elsc}{sl} {} {}
1834 <latexrelease>\DeclareFontSeriesChangeRule {elsc}{sx} {} {}
1835 <latexrelease>\DeclareFontSeriesChangeRule {elsc}{ub} {} {}
1836 <latexrelease>\DeclareFontSeriesChangeRule {elsc}{uc} {} {}
1837 <latexrelease>\DeclareFontSeriesChangeRule {elsc}{ul} {} {}
1838 <latexrelease>\DeclareFontSeriesChangeRule {elsc}{ux} {} {}
1839 <latexrelease>\DeclareFontSeriesChangeRule {elsc}{x} {} {}
1840 <latexrelease>\DeclareFontSeriesChangeRule {elsx}{?m} {} {}
1841 <latexrelease>\DeclareFontSeriesChangeRule {elsx}{bx} {} {}
1842 <latexrelease>\DeclareFontSeriesChangeRule {elsx}{b} {} {}
1843 <latexrelease>\DeclareFontSeriesChangeRule {elsx}{c} {} {}
1844 <latexrelease>\DeclareFontSeriesChangeRule {elsx}{eb} {} {}
1845 <latexrelease>\DeclareFontSeriesChangeRule {elsx}{ec} {} {}
1846 <latexrelease>\DeclareFontSeriesChangeRule {elsx}{el} {} {}
1847 <latexrelease>\DeclareFontSeriesChangeRule {elsx}{ex} {} {}
1848 <latexrelease>\DeclareFontSeriesChangeRule {elsx}{l} {} {}
1849 <latexrelease>\DeclareFontSeriesChangeRule {elsx}{m?} {} {}
1850 <latexrelease>\DeclareFontSeriesChangeRule {elsx}{sb} {} {}
1851 <latexrelease>\DeclareFontSeriesChangeRule {elsx}{sc} {} {}
1852 <latexrelease>\DeclareFontSeriesChangeRule {elsx}{sl} {} {}
1853 <latexrelease>\DeclareFontSeriesChangeRule {elsx}{sx} {} {}
1854 <latexrelease>\DeclareFontSeriesChangeRule {elsx}{ub} {} {}
1855 <latexrelease>\DeclareFontSeriesChangeRule {elsx}{uc} {} {}
1856 <latexrelease>\DeclareFontSeriesChangeRule {elsx}{ul} {} {}
1857 <latexrelease>\DeclareFontSeriesChangeRule {elsx}{ux} {} {}
1858 <latexrelease>\DeclareFontSeriesChangeRule {elsx}{x} {} {}
1859 <latexrelease>\DeclareFontSeriesChangeRule {eluc}{?m} {} {}
1860 <latexrelease>\DeclareFontSeriesChangeRule {eluc}{bx} {} {}
1861 <latexrelease>\DeclareFontSeriesChangeRule {eluc}{b} {} {}
1862 <latexrelease>\DeclareFontSeriesChangeRule {eluc}{c} {} {}
1863 <latexrelease>\DeclareFontSeriesChangeRule {eluc}{eb} {} {}
1864 <latexrelease>\DeclareFontSeriesChangeRule {eluc}{ec} {} {}
1865 <latexrelease>\DeclareFontSeriesChangeRule {eluc}{el} {} {}

```

```

1866 <latexrelease>\DeclareFontSeriesChangeRule {eluc}{ex} {} {}
1867 <latexrelease>\DeclareFontSeriesChangeRule {eluc}{l} {} {}
1868 <latexrelease>\DeclareFontSeriesChangeRule {eluc}{m?} {} {}
1869 <latexrelease>\DeclareFontSeriesChangeRule {eluc}{sb} {} {}
1870 <latexrelease>\DeclareFontSeriesChangeRule {eluc}{sc} {} {}
1871 <latexrelease>\DeclareFontSeriesChangeRule {eluc}{sl} {} {}
1872 <latexrelease>\DeclareFontSeriesChangeRule {eluc}{sx} {} {}
1873 <latexrelease>\DeclareFontSeriesChangeRule {eluc}{ub} {} {}
1874 <latexrelease>\DeclareFontSeriesChangeRule {eluc}{uc} {} {}
1875 <latexrelease>\DeclareFontSeriesChangeRule {eluc}{ul} {} {}
1876 <latexrelease>\DeclareFontSeriesChangeRule {eluc}{ux} {} {}
1877 <latexrelease>\DeclareFontSeriesChangeRule {eluc}{x} {} {}
1878 <latexrelease>\DeclareFontSeriesChangeRule {elux}{?m} {} {}
1879 <latexrelease>\DeclareFontSeriesChangeRule {elux}{bx} {} {}
1880 <latexrelease>\DeclareFontSeriesChangeRule {elux}{b} {} {}
1881 <latexrelease>\DeclareFontSeriesChangeRule {elux}{c} {} {}
1882 <latexrelease>\DeclareFontSeriesChangeRule {elux}{eb} {} {}
1883 <latexrelease>\DeclareFontSeriesChangeRule {elux}{ec} {} {}
1884 <latexrelease>\DeclareFontSeriesChangeRule {elux}{el} {} {}
1885 <latexrelease>\DeclareFontSeriesChangeRule {elux}{ex} {} {}
1886 <latexrelease>\DeclareFontSeriesChangeRule {elux}{l} {} {}
1887 <latexrelease>\DeclareFontSeriesChangeRule {elux}{m?} {} {}
1888 <latexrelease>\DeclareFontSeriesChangeRule {elux}{sb} {} {}
1889 <latexrelease>\DeclareFontSeriesChangeRule {elux}{sc} {} {}
1890 <latexrelease>\DeclareFontSeriesChangeRule {elux}{sl} {} {}
1891 <latexrelease>\DeclareFontSeriesChangeRule {elux}{sx} {} {}
1892 <latexrelease>\DeclareFontSeriesChangeRule {elux}{ub} {} {}
1893 <latexrelease>\DeclareFontSeriesChangeRule {elux}{uc} {} {}
1894 <latexrelease>\DeclareFontSeriesChangeRule {elux}{ul} {} {}
1895 <latexrelease>\DeclareFontSeriesChangeRule {elux}{ux} {} {}
1896 <latexrelease>\DeclareFontSeriesChangeRule {elux}{x} {} {}
1897 <latexrelease>\DeclareFontSeriesChangeRule {elx}{bx} {} {}
1898 <latexrelease>\DeclareFontSeriesChangeRule {elx}{b}{bx}{}
1899 <latexrelease>\DeclareFontSeriesChangeRule {elx}{c}{elc}{}
1900 <latexrelease>\DeclareFontSeriesChangeRule {elx}{eb}{ebx}{}
1901 <latexrelease>\DeclareFontSeriesChangeRule {elx}{ec}{elec}{}
1902 <latexrelease>\DeclareFontSeriesChangeRule {elx}{el}{elx}{}
1903 <latexrelease>\DeclareFontSeriesChangeRule {elx}{ex} {} {}
1904 <latexrelease>\DeclareFontSeriesChangeRule {elx}{l}{lx}{}
1905 <latexrelease>\DeclareFontSeriesChangeRule {elx}{sb}{sbx}{}
1906 <latexrelease>\DeclareFontSeriesChangeRule {elx}{sc}{elsc}{}
1907 <latexrelease>\DeclareFontSeriesChangeRule {elx}{sl}{slx}{}
1908 <latexrelease>\DeclareFontSeriesChangeRule {elx}{sx} {} {}
1909 <latexrelease>\DeclareFontSeriesChangeRule {elx}{ub}{ubx}{}
1910 <latexrelease>\DeclareFontSeriesChangeRule {elx}{uc} {} {}
1911 <latexrelease>\DeclareFontSeriesChangeRule {elx}{ul}{ulx}{}
1912 <latexrelease>\DeclareFontSeriesChangeRule {elx}{ux} {} {}
1913 <latexrelease>\DeclareFontSeriesChangeRule {elx}{x}{elx}{}
1914 <latexrelease>\DeclareFontSeriesChangeRule {el}{bx} {} {}
1915 <latexrelease>\DeclareFontSeriesChangeRule {el}{c}{elc}{}
1916 <latexrelease>\DeclareFontSeriesChangeRule {el}{eb} {} {}
1917 <latexrelease>\DeclareFontSeriesChangeRule {el}{ec}{elec}{}
1918 <latexrelease>\DeclareFontSeriesChangeRule {el}{ex} {} {}
1919 <latexrelease>\DeclareFontSeriesChangeRule {el}{sb} {} {}

```

```

1920 <latexrelease>\DeclareFontSeriesChangeRule {el}{sc}{elsc}{}
1921 <latexrelease>\DeclareFontSeriesChangeRule {el}{sx} {} {}
1922 <latexrelease>\DeclareFontSeriesChangeRule {el}{ub} {} {}
1923 <latexrelease>\DeclareFontSeriesChangeRule {el}{uc} {} {}
1924 <latexrelease>\DeclareFontSeriesChangeRule {el}{ux} {} {}
1925 <latexrelease>\DeclareFontSeriesChangeRule {el}{x}{elx}{}
1926 <latexrelease>\DeclareFontSeriesChangeRule {ex}{?m} {} {}
1927 <latexrelease>\DeclareFontSeriesChangeRule {ex}{bx} {} {}
1928 <latexrelease>\DeclareFontSeriesChangeRule {ex}{b} {} {}
1929 <latexrelease>\DeclareFontSeriesChangeRule {ex}{eb} {} {}
1930 <latexrelease>\DeclareFontSeriesChangeRule {ex}{el} {} {}
1931 <latexrelease>\DeclareFontSeriesChangeRule {ex}{l} {} {}
1932 <latexrelease>\DeclareFontSeriesChangeRule {ex}{m?} {} {}
1933 <latexrelease>\DeclareFontSeriesChangeRule {ex}{sb} {} {}
1934 <latexrelease>\DeclareFontSeriesChangeRule {ex}{sl} {} {}
1935 <latexrelease>\DeclareFontSeriesChangeRule {ex}{ub} {} {}
1936 <latexrelease>\DeclareFontSeriesChangeRule {ex}{ul} {} {}
1937 <latexrelease>\DeclareFontSeriesChangeRule {lc}{bx} {} {}
1938 <latexrelease>\DeclareFontSeriesChangeRule {lc}{b}{bc}{}
1939 <latexrelease>\DeclareFontSeriesChangeRule {lc}{c}{lc}{}
1940 <latexrelease>\DeclareFontSeriesChangeRule {lc}{eb}{ebc}{}
1941 <latexrelease>\DeclareFontSeriesChangeRule {lc}{ec}{lec}{}
1942 <latexrelease>\DeclareFontSeriesChangeRule {lc}{el}{elc}{}
1943 <latexrelease>\DeclareFontSeriesChangeRule {lc}{ex} {} {}
1944 <latexrelease>\DeclareFontSeriesChangeRule {lc}{l}{lc}{}
1945 <latexrelease>\DeclareFontSeriesChangeRule {lc}{sb}{sbc}{}
1946 <latexrelease>\DeclareFontSeriesChangeRule {lc}{sc}{lsc}{}
1947 <latexrelease>\DeclareFontSeriesChangeRule {lc}{sl}{slc}{}
1948 <latexrelease>\DeclareFontSeriesChangeRule {lc}{sx} {} {}
1949 <latexrelease>\DeclareFontSeriesChangeRule {lc}{ub}{ubc}{}
1950 <latexrelease>\DeclareFontSeriesChangeRule {lc}{uc} {} {}
1951 <latexrelease>\DeclareFontSeriesChangeRule {lc}{ul}{ulc}{}
1952 <latexrelease>\DeclareFontSeriesChangeRule {lc}{ux} {} {}
1953 <latexrelease>\DeclareFontSeriesChangeRule {lc}{x}{lx}{}
1954 <latexrelease>\DeclareFontSeriesChangeRule {lec}{bx} {} {}
1955 <latexrelease>\DeclareFontSeriesChangeRule {lec}{b} {} {}
1956 <latexrelease>\DeclareFontSeriesChangeRule {lec}{c} {} {}
1957 <latexrelease>\DeclareFontSeriesChangeRule {lec}{eb} {} {}
1958 <latexrelease>\DeclareFontSeriesChangeRule {lec}{ec} {} {}
1959 <latexrelease>\DeclareFontSeriesChangeRule {lec}{el} {} {}
1960 <latexrelease>\DeclareFontSeriesChangeRule {lec}{ex} {} {}
1961 <latexrelease>\DeclareFontSeriesChangeRule {lec}{l} {} {}
1962 <latexrelease>\DeclareFontSeriesChangeRule {lec}{sb} {} {}
1963 <latexrelease>\DeclareFontSeriesChangeRule {lec}{sc} {} {}
1964 <latexrelease>\DeclareFontSeriesChangeRule {lec}{sl} {} {}
1965 <latexrelease>\DeclareFontSeriesChangeRule {lec}{sx} {} {}
1966 <latexrelease>\DeclareFontSeriesChangeRule {lec}{ub} {} {}
1967 <latexrelease>\DeclareFontSeriesChangeRule {lec}{uc} {} {}
1968 <latexrelease>\DeclareFontSeriesChangeRule {lec}{ul} {} {}
1969 <latexrelease>\DeclareFontSeriesChangeRule {lec}{ux} {} {}
1970 <latexrelease>\DeclareFontSeriesChangeRule {lec}{x} {} {}
1971 <latexrelease>\DeclareFontSeriesChangeRule {lex}{?m} {} {}
1972 <latexrelease>\DeclareFontSeriesChangeRule {lex}{bx} {} {}
1973 <latexrelease>\DeclareFontSeriesChangeRule {lex}{b} {} {}

```

```

1974 <latexrelease>\DeclareFontSeriesChangeRule {lex}{c} {} {}
1975 <latexrelease>\DeclareFontSeriesChangeRule {lex}{eb} {} {}
1976 <latexrelease>\DeclareFontSeriesChangeRule {lex}{ec} {} {}
1977 <latexrelease>\DeclareFontSeriesChangeRule {lex}{el} {} {}
1978 <latexrelease>\DeclareFontSeriesChangeRule {lex}{ex} {} {}
1979 <latexrelease>\DeclareFontSeriesChangeRule {lex}{l} {} {}
1980 <latexrelease>\DeclareFontSeriesChangeRule {lex}{m?} {} {}
1981 <latexrelease>\DeclareFontSeriesChangeRule {lex}{sb} {} {}
1982 <latexrelease>\DeclareFontSeriesChangeRule {lex}{sc} {} {}
1983 <latexrelease>\DeclareFontSeriesChangeRule {lex}{sl} {} {}
1984 <latexrelease>\DeclareFontSeriesChangeRule {lex}{sx} {} {}
1985 <latexrelease>\DeclareFontSeriesChangeRule {lex}{ub} {} {}
1986 <latexrelease>\DeclareFontSeriesChangeRule {lex}{uc} {} {}
1987 <latexrelease>\DeclareFontSeriesChangeRule {lex}{ul} {} {}
1988 <latexrelease>\DeclareFontSeriesChangeRule {lex}{ux} {} {}
1989 <latexrelease>\DeclareFontSeriesChangeRule {lex}{x} {} {}
1990 <latexrelease>\DeclareFontSeriesChangeRule {lsc}{bx} {} {}
1991 <latexrelease>\DeclareFontSeriesChangeRule {lsc}{b} {} {}
1992 <latexrelease>\DeclareFontSeriesChangeRule {lsc}{c} {} {}
1993 <latexrelease>\DeclareFontSeriesChangeRule {lsc}{eb} {} {}
1994 <latexrelease>\DeclareFontSeriesChangeRule {lsc}{ec} {} {}
1995 <latexrelease>\DeclareFontSeriesChangeRule {lsc}{el} {} {}
1996 <latexrelease>\DeclareFontSeriesChangeRule {lsc}{ex} {} {}
1997 <latexrelease>\DeclareFontSeriesChangeRule {lsc}{l} {} {}
1998 <latexrelease>\DeclareFontSeriesChangeRule {lsc}{sb} {} {}
1999 <latexrelease>\DeclareFontSeriesChangeRule {lsc}{sc} {} {}
2000 <latexrelease>\DeclareFontSeriesChangeRule {lsc}{sl} {} {}
2001 <latexrelease>\DeclareFontSeriesChangeRule {lsc}{sx} {} {}
2002 <latexrelease>\DeclareFontSeriesChangeRule {lsc}{ub} {} {}
2003 <latexrelease>\DeclareFontSeriesChangeRule {lsc}{uc} {} {}
2004 <latexrelease>\DeclareFontSeriesChangeRule {lsc}{ul} {} {}
2005 <latexrelease>\DeclareFontSeriesChangeRule {lsc}{ux} {} {}
2006 <latexrelease>\DeclareFontSeriesChangeRule {lsc}{x} {} {}
2007 <latexrelease>\DeclareFontSeriesChangeRule {lsx}{?m} {} {}
2008 <latexrelease>\DeclareFontSeriesChangeRule {lsx}{bx} {} {}
2009 <latexrelease>\DeclareFontSeriesChangeRule {lsx}{b} {} {}
2010 <latexrelease>\DeclareFontSeriesChangeRule {lsx}{c} {} {}
2011 <latexrelease>\DeclareFontSeriesChangeRule {lsx}{eb} {} {}
2012 <latexrelease>\DeclareFontSeriesChangeRule {lsx}{ec} {} {}
2013 <latexrelease>\DeclareFontSeriesChangeRule {lsx}{el} {} {}
2014 <latexrelease>\DeclareFontSeriesChangeRule {lsx}{ex} {} {}
2015 <latexrelease>\DeclareFontSeriesChangeRule {lsx}{l} {} {}
2016 <latexrelease>\DeclareFontSeriesChangeRule {lsx}{m?} {} {}
2017 <latexrelease>\DeclareFontSeriesChangeRule {lsx}{sb} {} {}
2018 <latexrelease>\DeclareFontSeriesChangeRule {lsx}{sc} {} {}
2019 <latexrelease>\DeclareFontSeriesChangeRule {lsx}{sl} {} {}
2020 <latexrelease>\DeclareFontSeriesChangeRule {lsx}{sx} {} {}
2021 <latexrelease>\DeclareFontSeriesChangeRule {lsx}{ub} {} {}
2022 <latexrelease>\DeclareFontSeriesChangeRule {lsx}{uc} {} {}
2023 <latexrelease>\DeclareFontSeriesChangeRule {lsx}{ul} {} {}
2024 <latexrelease>\DeclareFontSeriesChangeRule {lsx}{ux} {} {}
2025 <latexrelease>\DeclareFontSeriesChangeRule {lsx}{x} {} {}
2026 <latexrelease>\DeclareFontSeriesChangeRule {luc}{?m} {} {}
2027 <latexrelease>\DeclareFontSeriesChangeRule {luc}{bx} {} {}

```

```

2028 <latexrelease>\DeclareFontSeriesChangeRule {luc}{b} {} {}
2029 <latexrelease>\DeclareFontSeriesChangeRule {luc}{c} {} {}
2030 <latexrelease>\DeclareFontSeriesChangeRule {luc}{eb} {} {}
2031 <latexrelease>\DeclareFontSeriesChangeRule {luc}{ec} {} {}
2032 <latexrelease>\DeclareFontSeriesChangeRule {luc}{el} {} {}
2033 <latexrelease>\DeclareFontSeriesChangeRule {luc}{ex} {} {}
2034 <latexrelease>\DeclareFontSeriesChangeRule {luc}{l} {} {}
2035 <latexrelease>\DeclareFontSeriesChangeRule {luc}{m?} {} {}
2036 <latexrelease>\DeclareFontSeriesChangeRule {luc}{sb} {} {}
2037 <latexrelease>\DeclareFontSeriesChangeRule {luc}{sc} {} {}
2038 <latexrelease>\DeclareFontSeriesChangeRule {luc}{sl} {} {}
2039 <latexrelease>\DeclareFontSeriesChangeRule {luc}{sx} {} {}
2040 <latexrelease>\DeclareFontSeriesChangeRule {luc}{ub} {} {}
2041 <latexrelease>\DeclareFontSeriesChangeRule {luc}{uc} {} {}
2042 <latexrelease>\DeclareFontSeriesChangeRule {luc}{ul} {} {}
2043 <latexrelease>\DeclareFontSeriesChangeRule {luc}{ux} {} {}
2044 <latexrelease>\DeclareFontSeriesChangeRule {luc}{x} {} {}
2045 <latexrelease>\DeclareFontSeriesChangeRule {lux}{?m} {} {}
2046 <latexrelease>\DeclareFontSeriesChangeRule {lux}{bx} {} {}
2047 <latexrelease>\DeclareFontSeriesChangeRule {lux}{b} {} {}
2048 <latexrelease>\DeclareFontSeriesChangeRule {lux}{c} {} {}
2049 <latexrelease>\DeclareFontSeriesChangeRule {lux}{eb} {} {}
2050 <latexrelease>\DeclareFontSeriesChangeRule {lux}{ec} {} {}
2051 <latexrelease>\DeclareFontSeriesChangeRule {lux}{el} {} {}
2052 <latexrelease>\DeclareFontSeriesChangeRule {lux}{ex} {} {}
2053 <latexrelease>\DeclareFontSeriesChangeRule {lux}{l} {} {}
2054 <latexrelease>\DeclareFontSeriesChangeRule {lux}{m?} {} {}
2055 <latexrelease>\DeclareFontSeriesChangeRule {lux}{sb} {} {}
2056 <latexrelease>\DeclareFontSeriesChangeRule {lux}{sc} {} {}
2057 <latexrelease>\DeclareFontSeriesChangeRule {lux}{sl} {} {}
2058 <latexrelease>\DeclareFontSeriesChangeRule {lux}{sx} {} {}
2059 <latexrelease>\DeclareFontSeriesChangeRule {lux}{ub} {} {}
2060 <latexrelease>\DeclareFontSeriesChangeRule {lux}{uc} {} {}
2061 <latexrelease>\DeclareFontSeriesChangeRule {lux}{ul} {} {}
2062 <latexrelease>\DeclareFontSeriesChangeRule {lux}{ux} {} {}
2063 <latexrelease>\DeclareFontSeriesChangeRule {lux}{x} {} {}
2064 <latexrelease>\DeclareFontSeriesChangeRule {lx}{bx} {} {}
2065 <latexrelease>\DeclareFontSeriesChangeRule {lx}{b}{bx}{}
2066 <latexrelease>\DeclareFontSeriesChangeRule {lx}{c}{lc}{}
2067 <latexrelease>\DeclareFontSeriesChangeRule {lx}{eb}{ebx}{}
2068 <latexrelease>\DeclareFontSeriesChangeRule {lx}{ec}{lec}{}
2069 <latexrelease>\DeclareFontSeriesChangeRule {lx}{el}{elx}{}
2070 <latexrelease>\DeclareFontSeriesChangeRule {lx}{ex} {} {}
2071 <latexrelease>\DeclareFontSeriesChangeRule {lx}{l}{lx}{}
2072 <latexrelease>\DeclareFontSeriesChangeRule {lx}{sb}{sbx}{}
2073 <latexrelease>\DeclareFontSeriesChangeRule {lx}{sc}{lsc}{}
2074 <latexrelease>\DeclareFontSeriesChangeRule {lx}{sl}{slx}{}
2075 <latexrelease>\DeclareFontSeriesChangeRule {lx}{sx} {} {}
2076 <latexrelease>\DeclareFontSeriesChangeRule {lx}{ub}{ubx}{}
2077 <latexrelease>\DeclareFontSeriesChangeRule {lx}{uc} {} {}
2078 <latexrelease>\DeclareFontSeriesChangeRule {lx}{ul}{ulx}{}
2079 <latexrelease>\DeclareFontSeriesChangeRule {lx}{ux} {} {}
2080 <latexrelease>\DeclareFontSeriesChangeRule {lx}{x}{lx}{}
2081 <latexrelease>\DeclareFontSeriesChangeRule {l}{b} {b} {bx}

```

```

2082 <latexrelease>\DeclareFontSeriesChangeRule {l}{c} {lc} {l}
2083 <latexrelease>\DeclareFontSeriesChangeRule {l}{eb} {} {}
2084 <latexrelease>\DeclareFontSeriesChangeRule {l}{ec} {lec} {l}
2085 <latexrelease>\DeclareFontSeriesChangeRule {l}{ex} {} {}
2086 <latexrelease>\DeclareFontSeriesChangeRule {l}{sb} {sb} {b}
2087 <latexrelease>\DeclareFontSeriesChangeRule {l}{sc} {lsc} {l}
2088 <latexrelease>\DeclareFontSeriesChangeRule {l}{sx} {} {}
2089 <latexrelease>\DeclareFontSeriesChangeRule {l}{ub} {} {}
2090 <latexrelease>\DeclareFontSeriesChangeRule {l}{uc} {} {}
2091 <latexrelease>\DeclareFontSeriesChangeRule {l}{ux} {} {}
2092 <latexrelease>\DeclareFontSeriesChangeRule {l}{x} {lx} {l}
2093 <latexrelease>\DeclareFontSeriesChangeRule {m}{bx} {bx} {b}
2094 <latexrelease>\DeclareFontSeriesChangeRule {m}{c} {c} {m}
2095 <latexrelease>\DeclareFontSeriesChangeRule {m}{l} {l} {m}
2096 <latexrelease>\DeclareFontSeriesChangeRule {m}{sc} {sc} {m}
2097 <latexrelease>\DeclareFontSeriesChangeRule {m}{x} {x} {m}
2098 <latexrelease>\DeclareFontSeriesChangeRule {sbc}{bx} {} {}
2099 <latexrelease>\DeclareFontSeriesChangeRule {sbc}{b}{bc}{}
2100 <latexrelease>\DeclareFontSeriesChangeRule {sbc}{c}{sbc}{}
2101 <latexrelease>\DeclareFontSeriesChangeRule {sbc}{eb}{ebc}{}
2102 <latexrelease>\DeclareFontSeriesChangeRule {sbc}{ec}{sbec}{sbc}
2103 <latexrelease>\DeclareFontSeriesChangeRule {sbc}{el}{elc}{}
2104 <latexrelease>\DeclareFontSeriesChangeRule {sbc}{ex} {} {}
2105 <latexrelease>\DeclareFontSeriesChangeRule {sbc}{l}{lc}{}
2106 <latexrelease>\DeclareFontSeriesChangeRule {sbc}{sb}{sbc}{}
2107 <latexrelease>\DeclareFontSeriesChangeRule {sbc}{sc}{sbsc}{sbc}
2108 <latexrelease>\DeclareFontSeriesChangeRule {sbc}{sl}{slc}{}
2109 <latexrelease>\DeclareFontSeriesChangeRule {sbc}{sx} {} {}
2110 <latexrelease>\DeclareFontSeriesChangeRule {sbc}{ub}{ubc}{}
2111 <latexrelease>\DeclareFontSeriesChangeRule {sbc}{uc} {} {}
2112 <latexrelease>\DeclareFontSeriesChangeRule {sbc}{ul}{ulc}{}
2113 <latexrelease>\DeclareFontSeriesChangeRule {sbc}{ux} {} {}
2114 <latexrelease>\DeclareFontSeriesChangeRule {sbc}{x}{sbx}{}
2115 <latexrelease>\DeclareFontSeriesChangeRule {sbec}{bx} {} {}
2116 <latexrelease>\DeclareFontSeriesChangeRule {sbec}{b} {} {}
2117 <latexrelease>\DeclareFontSeriesChangeRule {sbec}{c} {} {}
2118 <latexrelease>\DeclareFontSeriesChangeRule {sbec}{eb} {} {}
2119 <latexrelease>\DeclareFontSeriesChangeRule {sbec}{ec} {} {}
2120 <latexrelease>\DeclareFontSeriesChangeRule {sbec}{el} {} {}
2121 <latexrelease>\DeclareFontSeriesChangeRule {sbec}{ex} {} {}
2122 <latexrelease>\DeclareFontSeriesChangeRule {sbec}{l} {} {}
2123 <latexrelease>\DeclareFontSeriesChangeRule {sbec}{sb} {} {}
2124 <latexrelease>\DeclareFontSeriesChangeRule {sbec}{sc} {} {}
2125 <latexrelease>\DeclareFontSeriesChangeRule {sbec}{sl} {} {}
2126 <latexrelease>\DeclareFontSeriesChangeRule {sbec}{sx} {} {}
2127 <latexrelease>\DeclareFontSeriesChangeRule {sbec}{ub} {} {}
2128 <latexrelease>\DeclareFontSeriesChangeRule {sbec}{uc} {} {}
2129 <latexrelease>\DeclareFontSeriesChangeRule {sbec}{ul} {} {}
2130 <latexrelease>\DeclareFontSeriesChangeRule {sbec}{ux} {} {}
2131 <latexrelease>\DeclareFontSeriesChangeRule {sbec}{x} {} {}
2132 <latexrelease>\DeclareFontSeriesChangeRule {sbex}{?m} {} {}
2133 <latexrelease>\DeclareFontSeriesChangeRule {sbex}{bx} {} {}
2134 <latexrelease>\DeclareFontSeriesChangeRule {sbex}{b} {} {}
2135 <latexrelease>\DeclareFontSeriesChangeRule {sbex}{c} {} {}

```

```

2136 <latexrelease>\DeclareFontSeriesChangeRule {sbex}{eb} {} {}
2137 <latexrelease>\DeclareFontSeriesChangeRule {sbex}{ec} {} {}
2138 <latexrelease>\DeclareFontSeriesChangeRule {sbex}{el} {} {}
2139 <latexrelease>\DeclareFontSeriesChangeRule {sbex}{ex} {} {}
2140 <latexrelease>\DeclareFontSeriesChangeRule {sbex}{l} {} {}
2141 <latexrelease>\DeclareFontSeriesChangeRule {sbex}{m?} {} {}
2142 <latexrelease>\DeclareFontSeriesChangeRule {sbex}{sb} {} {}
2143 <latexrelease>\DeclareFontSeriesChangeRule {sbex}{sc} {} {}
2144 <latexrelease>\DeclareFontSeriesChangeRule {sbex}{sl} {} {}
2145 <latexrelease>\DeclareFontSeriesChangeRule {sbex}{sx} {} {}
2146 <latexrelease>\DeclareFontSeriesChangeRule {sbex}{ub} {} {}
2147 <latexrelease>\DeclareFontSeriesChangeRule {sbex}{uc} {} {}
2148 <latexrelease>\DeclareFontSeriesChangeRule {sbex}{ul} {} {}
2149 <latexrelease>\DeclareFontSeriesChangeRule {sbex}{ux} {} {}
2150 <latexrelease>\DeclareFontSeriesChangeRule {sbex}{x} {} {}
2151 <latexrelease>\DeclareFontSeriesChangeRule {sbsc}{bx} {} {}
2152 <latexrelease>\DeclareFontSeriesChangeRule {sbsc}{b} {} {}
2153 <latexrelease>\DeclareFontSeriesChangeRule {sbsc}{c} {} {}
2154 <latexrelease>\DeclareFontSeriesChangeRule {sbsc}{eb} {} {}
2155 <latexrelease>\DeclareFontSeriesChangeRule {sbsc}{ec} {} {}
2156 <latexrelease>\DeclareFontSeriesChangeRule {sbsc}{el} {} {}
2157 <latexrelease>\DeclareFontSeriesChangeRule {sbsc}{ex} {} {}
2158 <latexrelease>\DeclareFontSeriesChangeRule {sbsc}{l} {} {}
2159 <latexrelease>\DeclareFontSeriesChangeRule {sbsc}{sb} {} {}
2160 <latexrelease>\DeclareFontSeriesChangeRule {sbsc}{sc} {} {}
2161 <latexrelease>\DeclareFontSeriesChangeRule {sbsc}{sl} {} {}
2162 <latexrelease>\DeclareFontSeriesChangeRule {sbsc}{sx} {} {}
2163 <latexrelease>\DeclareFontSeriesChangeRule {sbsc}{ub} {} {}
2164 <latexrelease>\DeclareFontSeriesChangeRule {sbsc}{uc} {} {}
2165 <latexrelease>\DeclareFontSeriesChangeRule {sbsc}{ul} {} {}
2166 <latexrelease>\DeclareFontSeriesChangeRule {sbsc}{ux} {} {}
2167 <latexrelease>\DeclareFontSeriesChangeRule {sbsc}{x} {} {}
2168 <latexrelease>\DeclareFontSeriesChangeRule {sbsx}{?m} {} {}
2169 <latexrelease>\DeclareFontSeriesChangeRule {sbsx}{bx} {} {}
2170 <latexrelease>\DeclareFontSeriesChangeRule {sbsx}{b} {} {}
2171 <latexrelease>\DeclareFontSeriesChangeRule {sbsx}{c} {} {}
2172 <latexrelease>\DeclareFontSeriesChangeRule {sbsx}{eb} {} {}
2173 <latexrelease>\DeclareFontSeriesChangeRule {sbsx}{ec} {} {}
2174 <latexrelease>\DeclareFontSeriesChangeRule {sbsx}{el} {} {}
2175 <latexrelease>\DeclareFontSeriesChangeRule {sbsx}{ex} {} {}
2176 <latexrelease>\DeclareFontSeriesChangeRule {sbsx}{l} {} {}
2177 <latexrelease>\DeclareFontSeriesChangeRule {sbsx}{m?} {} {}
2178 <latexrelease>\DeclareFontSeriesChangeRule {sbsx}{sb} {} {}
2179 <latexrelease>\DeclareFontSeriesChangeRule {sbsx}{sc} {} {}
2180 <latexrelease>\DeclareFontSeriesChangeRule {sbsx}{sl} {} {}
2181 <latexrelease>\DeclareFontSeriesChangeRule {sbsx}{sx} {} {}
2182 <latexrelease>\DeclareFontSeriesChangeRule {sbsx}{ub} {} {}
2183 <latexrelease>\DeclareFontSeriesChangeRule {sbsx}{uc} {} {}
2184 <latexrelease>\DeclareFontSeriesChangeRule {sbsx}{ul} {} {}
2185 <latexrelease>\DeclareFontSeriesChangeRule {sbsx}{ux} {} {}
2186 <latexrelease>\DeclareFontSeriesChangeRule {sbsx}{x} {} {}
2187 <latexrelease>\DeclareFontSeriesChangeRule {sbuc}{?m} {} {}
2188 <latexrelease>\DeclareFontSeriesChangeRule {sbuc}{bx} {} {}
2189 <latexrelease>\DeclareFontSeriesChangeRule {sbuc}{b} {} {}

```



```

2190 <latexrelease>\DeclareFontSeriesChangeRule {sbuc}{c} {} {}
2191 <latexrelease>\DeclareFontSeriesChangeRule {sbuc}{eb} {} {}
2192 <latexrelease>\DeclareFontSeriesChangeRule {sbuc}{ec} {} {}
2193 <latexrelease>\DeclareFontSeriesChangeRule {sbuc}{el} {} {}
2194 <latexrelease>\DeclareFontSeriesChangeRule {sbuc}{ex} {} {}
2195 <latexrelease>\DeclareFontSeriesChangeRule {sbuc}{l} {} {}
2196 <latexrelease>\DeclareFontSeriesChangeRule {sbuc}{m?} {} {}
2197 <latexrelease>\DeclareFontSeriesChangeRule {sbuc}{sb} {} {}
2198 <latexrelease>\DeclareFontSeriesChangeRule {sbuc}{sc} {} {}
2199 <latexrelease>\DeclareFontSeriesChangeRule {sbuc}{sl} {} {}
2200 <latexrelease>\DeclareFontSeriesChangeRule {sbuc}{sx} {} {}
2201 <latexrelease>\DeclareFontSeriesChangeRule {sbuc}{ub} {} {}
2202 <latexrelease>\DeclareFontSeriesChangeRule {sbuc}{uc} {} {}
2203 <latexrelease>\DeclareFontSeriesChangeRule {sbuc}{ul} {} {}
2204 <latexrelease>\DeclareFontSeriesChangeRule {sbuc}{ux} {} {}
2205 <latexrelease>\DeclareFontSeriesChangeRule {sbuc}{x} {} {}
2206 <latexrelease>\DeclareFontSeriesChangeRule {sbux}{?m} {} {}
2207 <latexrelease>\DeclareFontSeriesChangeRule {sbux}{bx} {} {}
2208 <latexrelease>\DeclareFontSeriesChangeRule {sbux}{b} {} {}
2209 <latexrelease>\DeclareFontSeriesChangeRule {sbux}{c} {} {}
2210 <latexrelease>\DeclareFontSeriesChangeRule {sbux}{eb} {} {}
2211 <latexrelease>\DeclareFontSeriesChangeRule {sbux}{ec} {} {}
2212 <latexrelease>\DeclareFontSeriesChangeRule {sbux}{el} {} {}
2213 <latexrelease>\DeclareFontSeriesChangeRule {sbux}{ex} {} {}
2214 <latexrelease>\DeclareFontSeriesChangeRule {sbux}{l} {} {}
2215 <latexrelease>\DeclareFontSeriesChangeRule {sbux}{m?} {} {}
2216 <latexrelease>\DeclareFontSeriesChangeRule {sbux}{sb} {} {}
2217 <latexrelease>\DeclareFontSeriesChangeRule {sbux}{sc} {} {}
2218 <latexrelease>\DeclareFontSeriesChangeRule {sbux}{sl} {} {}
2219 <latexrelease>\DeclareFontSeriesChangeRule {sbux}{sx} {} {}
2220 <latexrelease>\DeclareFontSeriesChangeRule {sbux}{ub} {} {}
2221 <latexrelease>\DeclareFontSeriesChangeRule {sbux}{uc} {} {}
2222 <latexrelease>\DeclareFontSeriesChangeRule {sbux}{ul} {} {}
2223 <latexrelease>\DeclareFontSeriesChangeRule {sbux}{ux} {} {}
2224 <latexrelease>\DeclareFontSeriesChangeRule {sbux}{x} {} {}
2225 <latexrelease>\DeclareFontSeriesChangeRule {sbx}{bx} {} {}
2226 <latexrelease>\DeclareFontSeriesChangeRule {sbx}{b}{bx}{}
2227 <latexrelease>\DeclareFontSeriesChangeRule {sbx}{c}{sbc}{}
2228 <latexrelease>\DeclareFontSeriesChangeRule {sbx}{eb}{ebx}{}
2229 <latexrelease>\DeclareFontSeriesChangeRule {sbx}{ec}{sbec}{}
2230 <latexrelease>\DeclareFontSeriesChangeRule {sbx}{el}{elx}{}
2231 <latexrelease>\DeclareFontSeriesChangeRule {sbx}{ex} {} {}
2232 <latexrelease>\DeclareFontSeriesChangeRule {sbx}{l}{lx}{}
2233 <latexrelease>\DeclareFontSeriesChangeRule {sbx}{sb}{sbsx}{}
2234 <latexrelease>\DeclareFontSeriesChangeRule {sbx}{sc}{sbsc}{}
2235 <latexrelease>\DeclareFontSeriesChangeRule {sbx}{sl}{slx}{}
2236 <latexrelease>\DeclareFontSeriesChangeRule {sbx}{sx} {} {}
2237 <latexrelease>\DeclareFontSeriesChangeRule {sbx}{ub}{ubx}{}
2238 <latexrelease>\DeclareFontSeriesChangeRule {sbx}{uc} {} {}
2239 <latexrelease>\DeclareFontSeriesChangeRule {sbx}{ul}{ulx}{}
2240 <latexrelease>\DeclareFontSeriesChangeRule {sbx}{ux} {} {}
2241 <latexrelease>\DeclareFontSeriesChangeRule {sbx}{x}{sbx}{}
2242 <latexrelease>\DeclareFontSeriesChangeRule {sb}{bx} {} {}
2243 <latexrelease>\DeclareFontSeriesChangeRule {sb}{c} {sbc} {bc}

```

```

2244 <latexrelease>\DeclareFontSeriesChangeRule {sb}{eb} {} {}
2245 <latexrelease>\DeclareFontSeriesChangeRule {sb}{ec} {sbec} {sbc}
2246 <latexrelease>\DeclareFontSeriesChangeRule {sb}{ex} {} {}
2247 <latexrelease>\DeclareFontSeriesChangeRule {sb}{sc} {sbsc} {sbc}
2248 <latexrelease>\DeclareFontSeriesChangeRule {sb}{sx} {} {}
2249 <latexrelease>\DeclareFontSeriesChangeRule {sb}{ub} {} {}
2250 <latexrelease>\DeclareFontSeriesChangeRule {sb}{uc} {} {}
2251 <latexrelease>\DeclareFontSeriesChangeRule {sb}{ux} {} {}
2252 <latexrelease>\DeclareFontSeriesChangeRule {sb}{x} {sbx} {bx}
2253 <latexrelease>\DeclareFontSeriesChangeRule {sc}{bx} {bx} {b}
2254 <latexrelease>\DeclareFontSeriesChangeRule {sc}{b}{bsc}{c}
2255 <latexrelease>\DeclareFontSeriesChangeRule {sc}{eb}{ebsc}{c}
2256 <latexrelease>\DeclareFontSeriesChangeRule {sc}{el}{elsc}{c}
2257 <latexrelease>\DeclareFontSeriesChangeRule {sc}{l}{lsc}{c}
2258 <latexrelease>\DeclareFontSeriesChangeRule {sc}{sb}{sbsc}{c}
2259 <latexrelease>\DeclareFontSeriesChangeRule {sc}{sl}{slsc}{c}
2260 <latexrelease>\DeclareFontSeriesChangeRule {sc}{ub}{ubsc}{c}
2261 <latexrelease>\DeclareFontSeriesChangeRule {sc}{x}{x}{m}
2262 <latexrelease>\DeclareFontSeriesChangeRule {slc}{bx} {} {}
2263 <latexrelease>\DeclareFontSeriesChangeRule {slc}{b}{bc}{c}
2264 <latexrelease>\DeclareFontSeriesChangeRule {slc}{c}{slc}{c}
2265 <latexrelease>\DeclareFontSeriesChangeRule {slc}{eb}{ebc}{c}
2266 <latexrelease>\DeclareFontSeriesChangeRule {slc}{ec}{slec}{c}
2267 <latexrelease>\DeclareFontSeriesChangeRule {slc}{el}{elc}{c}
2268 <latexrelease>\DeclareFontSeriesChangeRule {slc}{ex} {} {}
2269 <latexrelease>\DeclareFontSeriesChangeRule {slc}{l}{lc}{c}
2270 <latexrelease>\DeclareFontSeriesChangeRule {slc}{sb}{sbc}{c}
2271 <latexrelease>\DeclareFontSeriesChangeRule {slc}{sc}{slsc}{c}
2272 <latexrelease>\DeclareFontSeriesChangeRule {slc}{sl}{slc}{c}
2273 <latexrelease>\DeclareFontSeriesChangeRule {slc}{sx} {} {}
2274 <latexrelease>\DeclareFontSeriesChangeRule {slc}{ub}{ubc}{c}
2275 <latexrelease>\DeclareFontSeriesChangeRule {slc}{uc} {} {}
2276 <latexrelease>\DeclareFontSeriesChangeRule {slc}{ul}{ulc}{c}
2277 <latexrelease>\DeclareFontSeriesChangeRule {slc}{ux} {} {}
2278 <latexrelease>\DeclareFontSeriesChangeRule {slc}{x}{slx}{c}
2279 <latexrelease>\DeclareFontSeriesChangeRule {slec}{bx} {} {}
2280 <latexrelease>\DeclareFontSeriesChangeRule {slec}{b} {} {}
2281 <latexrelease>\DeclareFontSeriesChangeRule {slec}{c} {} {}
2282 <latexrelease>\DeclareFontSeriesChangeRule {slec}{eb} {} {}
2283 <latexrelease>\DeclareFontSeriesChangeRule {slec}{ec} {} {}
2284 <latexrelease>\DeclareFontSeriesChangeRule {slec}{el} {} {}
2285 <latexrelease>\DeclareFontSeriesChangeRule {slec}{ex} {} {}
2286 <latexrelease>\DeclareFontSeriesChangeRule {slec}{l} {} {}
2287 <latexrelease>\DeclareFontSeriesChangeRule {slec}{sb} {} {}
2288 <latexrelease>\DeclareFontSeriesChangeRule {slec}{sc} {} {}
2289 <latexrelease>\DeclareFontSeriesChangeRule {slec}{sl} {} {}
2290 <latexrelease>\DeclareFontSeriesChangeRule {slec}{sx} {} {}
2291 <latexrelease>\DeclareFontSeriesChangeRule {slec}{ub} {} {}
2292 <latexrelease>\DeclareFontSeriesChangeRule {slec}{uc} {} {}
2293 <latexrelease>\DeclareFontSeriesChangeRule {slec}{ul} {} {}
2294 <latexrelease>\DeclareFontSeriesChangeRule {slec}{ux} {} {}
2295 <latexrelease>\DeclareFontSeriesChangeRule {slec}{x} {} {}
2296 <latexrelease>\DeclareFontSeriesChangeRule {slex}{?m} {} {}
2297 <latexrelease>\DeclareFontSeriesChangeRule {slex}{bx} {} {}

```

```

2298 <latexrelease>\DeclareFontSeriesChangeRule {slex}{b} {} {}
2299 <latexrelease>\DeclareFontSeriesChangeRule {slex}{c} {} {}
2300 <latexrelease>\DeclareFontSeriesChangeRule {slex}{eb} {} {}
2301 <latexrelease>\DeclareFontSeriesChangeRule {slex}{ec} {} {}
2302 <latexrelease>\DeclareFontSeriesChangeRule {slex}{el} {} {}
2303 <latexrelease>\DeclareFontSeriesChangeRule {slex}{ex} {} {}
2304 <latexrelease>\DeclareFontSeriesChangeRule {slex}{l} {} {}
2305 <latexrelease>\DeclareFontSeriesChangeRule {slex}{m?} {} {}
2306 <latexrelease>\DeclareFontSeriesChangeRule {slex}{sb} {} {}
2307 <latexrelease>\DeclareFontSeriesChangeRule {slex}{sc} {} {}
2308 <latexrelease>\DeclareFontSeriesChangeRule {slex}{sl} {} {}
2309 <latexrelease>\DeclareFontSeriesChangeRule {slex}{sx} {} {}
2310 <latexrelease>\DeclareFontSeriesChangeRule {slex}{ub} {} {}
2311 <latexrelease>\DeclareFontSeriesChangeRule {slex}{uc} {} {}
2312 <latexrelease>\DeclareFontSeriesChangeRule {slex}{ul} {} {}
2313 <latexrelease>\DeclareFontSeriesChangeRule {slex}{ux} {} {}
2314 <latexrelease>\DeclareFontSeriesChangeRule {slex}{x} {} {}
2315 <latexrelease>\DeclareFontSeriesChangeRule {slsc}{bx} {} {}
2316 <latexrelease>\DeclareFontSeriesChangeRule {slsc}{b} {} {}
2317 <latexrelease>\DeclareFontSeriesChangeRule {slsc}{c} {} {}
2318 <latexrelease>\DeclareFontSeriesChangeRule {slsc}{eb} {} {}
2319 <latexrelease>\DeclareFontSeriesChangeRule {slsc}{ec} {} {}
2320 <latexrelease>\DeclareFontSeriesChangeRule {slsc}{el} {} {}
2321 <latexrelease>\DeclareFontSeriesChangeRule {slsc}{ex} {} {}
2322 <latexrelease>\DeclareFontSeriesChangeRule {slsc}{l} {} {}
2323 <latexrelease>\DeclareFontSeriesChangeRule {slsc}{sb} {} {}
2324 <latexrelease>\DeclareFontSeriesChangeRule {slsc}{sc} {} {}
2325 <latexrelease>\DeclareFontSeriesChangeRule {slsc}{sl} {} {}
2326 <latexrelease>\DeclareFontSeriesChangeRule {slsc}{sx} {} {}
2327 <latexrelease>\DeclareFontSeriesChangeRule {slsc}{ub} {} {}
2328 <latexrelease>\DeclareFontSeriesChangeRule {slsc}{uc} {} {}
2329 <latexrelease>\DeclareFontSeriesChangeRule {slsc}{ul} {} {}
2330 <latexrelease>\DeclareFontSeriesChangeRule {slsc}{ux} {} {}
2331 <latexrelease>\DeclareFontSeriesChangeRule {slsc}{x} {} {}
2332 <latexrelease>\DeclareFontSeriesChangeRule {slsx}{?m} {} {}
2333 <latexrelease>\DeclareFontSeriesChangeRule {slsx}{bx} {} {}
2334 <latexrelease>\DeclareFontSeriesChangeRule {slsx}{b} {} {}
2335 <latexrelease>\DeclareFontSeriesChangeRule {slsx}{c} {} {}
2336 <latexrelease>\DeclareFontSeriesChangeRule {slsx}{eb} {} {}
2337 <latexrelease>\DeclareFontSeriesChangeRule {slsx}{ec} {} {}
2338 <latexrelease>\DeclareFontSeriesChangeRule {slsx}{el} {} {}
2339 <latexrelease>\DeclareFontSeriesChangeRule {slsx}{ex} {} {}
2340 <latexrelease>\DeclareFontSeriesChangeRule {slsx}{l} {} {}
2341 <latexrelease>\DeclareFontSeriesChangeRule {slsx}{m?} {} {}
2342 <latexrelease>\DeclareFontSeriesChangeRule {slsx}{sb} {} {}
2343 <latexrelease>\DeclareFontSeriesChangeRule {slsx}{sc} {} {}
2344 <latexrelease>\DeclareFontSeriesChangeRule {slsx}{sl} {} {}
2345 <latexrelease>\DeclareFontSeriesChangeRule {slsx}{sx} {} {}
2346 <latexrelease>\DeclareFontSeriesChangeRule {slsx}{ub} {} {}
2347 <latexrelease>\DeclareFontSeriesChangeRule {slsx}{uc} {} {}
2348 <latexrelease>\DeclareFontSeriesChangeRule {slsx}{ul} {} {}
2349 <latexrelease>\DeclareFontSeriesChangeRule {slsx}{ux} {} {}
2350 <latexrelease>\DeclareFontSeriesChangeRule {slsx}{x} {} {}
2351 <latexrelease>\DeclareFontSeriesChangeRule {sluc}{?m} {} {}

```

```

2352 <latexrelease>\DeclareFontSeriesChangeRule {sluc}{bx} {} {}
2353 <latexrelease>\DeclareFontSeriesChangeRule {sluc}{b} {} {}
2354 <latexrelease>\DeclareFontSeriesChangeRule {sluc}{c} {} {}
2355 <latexrelease>\DeclareFontSeriesChangeRule {sluc}{eb} {} {}
2356 <latexrelease>\DeclareFontSeriesChangeRule {sluc}{ec} {} {}
2357 <latexrelease>\DeclareFontSeriesChangeRule {sluc}{el} {} {}
2358 <latexrelease>\DeclareFontSeriesChangeRule {sluc}{ex} {} {}
2359 <latexrelease>\DeclareFontSeriesChangeRule {sluc}{l} {} {}
2360 <latexrelease>\DeclareFontSeriesChangeRule {sluc}{m?} {} {}
2361 <latexrelease>\DeclareFontSeriesChangeRule {sluc}{sb} {} {}
2362 <latexrelease>\DeclareFontSeriesChangeRule {sluc}{sc} {} {}
2363 <latexrelease>\DeclareFontSeriesChangeRule {sluc}{sl} {} {}
2364 <latexrelease>\DeclareFontSeriesChangeRule {sluc}{sx} {} {}
2365 <latexrelease>\DeclareFontSeriesChangeRule {sluc}{ub} {} {}
2366 <latexrelease>\DeclareFontSeriesChangeRule {sluc}{uc} {} {}
2367 <latexrelease>\DeclareFontSeriesChangeRule {sluc}{ul} {} {}
2368 <latexrelease>\DeclareFontSeriesChangeRule {sluc}{ux} {} {}
2369 <latexrelease>\DeclareFontSeriesChangeRule {sluc}{x} {} {}
2370 <latexrelease>\DeclareFontSeriesChangeRule {slux}{?m} {} {}
2371 <latexrelease>\DeclareFontSeriesChangeRule {slux}{bx} {} {}
2372 <latexrelease>\DeclareFontSeriesChangeRule {slux}{b} {} {}
2373 <latexrelease>\DeclareFontSeriesChangeRule {slux}{c} {} {}
2374 <latexrelease>\DeclareFontSeriesChangeRule {slux}{eb} {} {}
2375 <latexrelease>\DeclareFontSeriesChangeRule {slux}{ec} {} {}
2376 <latexrelease>\DeclareFontSeriesChangeRule {slux}{el} {} {}
2377 <latexrelease>\DeclareFontSeriesChangeRule {slux}{ex} {} {}
2378 <latexrelease>\DeclareFontSeriesChangeRule {slux}{l} {} {}
2379 <latexrelease>\DeclareFontSeriesChangeRule {slux}{m?} {} {}
2380 <latexrelease>\DeclareFontSeriesChangeRule {slux}{sb} {} {}
2381 <latexrelease>\DeclareFontSeriesChangeRule {slux}{sc} {} {}
2382 <latexrelease>\DeclareFontSeriesChangeRule {slux}{sl} {} {}
2383 <latexrelease>\DeclareFontSeriesChangeRule {slux}{sx} {} {}
2384 <latexrelease>\DeclareFontSeriesChangeRule {slux}{ub} {} {}
2385 <latexrelease>\DeclareFontSeriesChangeRule {slux}{uc} {} {}
2386 <latexrelease>\DeclareFontSeriesChangeRule {slux}{ul} {} {}
2387 <latexrelease>\DeclareFontSeriesChangeRule {slux}{ux} {} {}
2388 <latexrelease>\DeclareFontSeriesChangeRule {slux}{x} {} {}
2389 <latexrelease>\DeclareFontSeriesChangeRule {slx}{bx} {} {}
2390 <latexrelease>\DeclareFontSeriesChangeRule {slx}{b}{bx}{}
2391 <latexrelease>\DeclareFontSeriesChangeRule {slx}{c}{slc}{}
2392 <latexrelease>\DeclareFontSeriesChangeRule {slx}{eb}{ebx}{}
2393 <latexrelease>\DeclareFontSeriesChangeRule {slx}{ec}{slc}{}
2394 <latexrelease>\DeclareFontSeriesChangeRule {slx}{el}{elx}{}
2395 <latexrelease>\DeclareFontSeriesChangeRule {slx}{ex} {} {}
2396 <latexrelease>\DeclareFontSeriesChangeRule {slx}{l}{lx}{}
2397 <latexrelease>\DeclareFontSeriesChangeRule {slx}{sb}{sbx}{}
2398 <latexrelease>\DeclareFontSeriesChangeRule {slx}{sc}{slc}{}
2399 <latexrelease>\DeclareFontSeriesChangeRule {slx}{sl}{slx}{}
2400 <latexrelease>\DeclareFontSeriesChangeRule {slx}{sx} {} {}
2401 <latexrelease>\DeclareFontSeriesChangeRule {slx}{ub}{ubx}{}
2402 <latexrelease>\DeclareFontSeriesChangeRule {slx}{uc} {} {}
2403 <latexrelease>\DeclareFontSeriesChangeRule {slx}{ul}{ulx}{}
2404 <latexrelease>\DeclareFontSeriesChangeRule {slx}{ux} {} {}
2405 <latexrelease>\DeclareFontSeriesChangeRule {slx}{x}{slx}{}

```

```

2406 <latexrelease>\DeclareFontSeriesChangeRule {sl}{bx} {} {}
2407 <latexrelease>\DeclareFontSeriesChangeRule {sl}{c}{slc}{}
2408 <latexrelease>\DeclareFontSeriesChangeRule {sl}{eb} {} {}
2409 <latexrelease>\DeclareFontSeriesChangeRule {sl}{ec}{slec}{}
2410 <latexrelease>\DeclareFontSeriesChangeRule {sl}{ex} {} {}
2411 <latexrelease>\DeclareFontSeriesChangeRule {sl}{sb} {} {}
2412 <latexrelease>\DeclareFontSeriesChangeRule {sl}{sc}{slsc}{}
2413 <latexrelease>\DeclareFontSeriesChangeRule {sl}{sx} {} {}
2414 <latexrelease>\DeclareFontSeriesChangeRule {sl}{ub} {} {}
2415 <latexrelease>\DeclareFontSeriesChangeRule {sl}{uc} {} {}
2416 <latexrelease>\DeclareFontSeriesChangeRule {sl}{ux} {} {}
2417 <latexrelease>\DeclareFontSeriesChangeRule {sl}{x}{slx}{}
2418 <latexrelease>\DeclareFontSeriesChangeRule {sx}{?m} {} {}
2419 <latexrelease>\DeclareFontSeriesChangeRule {sx}{bx} {} {}
2420 <latexrelease>\DeclareFontSeriesChangeRule {sx}{b} {} {}
2421 <latexrelease>\DeclareFontSeriesChangeRule {sx}{eb} {} {}
2422 <latexrelease>\DeclareFontSeriesChangeRule {sx}{el} {} {}
2423 <latexrelease>\DeclareFontSeriesChangeRule {sx}{l} {} {}
2424 <latexrelease>\DeclareFontSeriesChangeRule {sx}{m?} {} {}
2425 <latexrelease>\DeclareFontSeriesChangeRule {sx}{sb} {} {}
2426 <latexrelease>\DeclareFontSeriesChangeRule {sx}{sl} {} {}
2427 <latexrelease>\DeclareFontSeriesChangeRule {sx}{ub} {} {}
2428 <latexrelease>\DeclareFontSeriesChangeRule {sx}{ul} {} {}
2429 <latexrelease>\DeclareFontSeriesChangeRule {ubc}{bx} {} {}
2430 <latexrelease>\DeclareFontSeriesChangeRule {ubc}{b}{bc}{}
2431 <latexrelease>\DeclareFontSeriesChangeRule {ubc}{c}{uc}{}
2432 <latexrelease>\DeclareFontSeriesChangeRule {ubc}{eb}{ebc}{}
2433 <latexrelease>\DeclareFontSeriesChangeRule {ubc}{ec}{ubec}{}
2434 <latexrelease>\DeclareFontSeriesChangeRule {ubc}{el}{elc}{}
2435 <latexrelease>\DeclareFontSeriesChangeRule {ubc}{ex} {} {}
2436 <latexrelease>\DeclareFontSeriesChangeRule {ubc}{l}{lc}{}
2437 <latexrelease>\DeclareFontSeriesChangeRule {ubc}{sb}{sbc}{}
2438 <latexrelease>\DeclareFontSeriesChangeRule {ubc}{sc}{ubsc}{}
2439 <latexrelease>\DeclareFontSeriesChangeRule {ubc}{sl}{slc}{}
2440 <latexrelease>\DeclareFontSeriesChangeRule {ubc}{sx} {} {}
2441 <latexrelease>\DeclareFontSeriesChangeRule {ubc}{ub}{uc}{}
2442 <latexrelease>\DeclareFontSeriesChangeRule {ubc}{uc} {} {}
2443 <latexrelease>\DeclareFontSeriesChangeRule {ubc}{ul}{ulc}{}
2444 <latexrelease>\DeclareFontSeriesChangeRule {ubc}{ux} {} {}
2445 <latexrelease>\DeclareFontSeriesChangeRule {ubc}{x}{ubx}{}
2446 <latexrelease>\DeclareFontSeriesChangeRule {ubec}{bx} {} {}
2447 <latexrelease>\DeclareFontSeriesChangeRule {ubec}{b} {} {}
2448 <latexrelease>\DeclareFontSeriesChangeRule {ubec}{c} {} {}
2449 <latexrelease>\DeclareFontSeriesChangeRule {ubec}{eb} {} {}
2450 <latexrelease>\DeclareFontSeriesChangeRule {ubec}{ec} {} {}
2451 <latexrelease>\DeclareFontSeriesChangeRule {ubec}{el} {} {}
2452 <latexrelease>\DeclareFontSeriesChangeRule {ubec}{ex} {} {}
2453 <latexrelease>\DeclareFontSeriesChangeRule {ubec}{l} {} {}
2454 <latexrelease>\DeclareFontSeriesChangeRule {ubec}{sb} {} {}
2455 <latexrelease>\DeclareFontSeriesChangeRule {ubec}{sc} {} {}
2456 <latexrelease>\DeclareFontSeriesChangeRule {ubec}{sl} {} {}
2457 <latexrelease>\DeclareFontSeriesChangeRule {ubec}{sx} {} {}
2458 <latexrelease>\DeclareFontSeriesChangeRule {ubec}{ub} {} {}
2459 <latexrelease>\DeclareFontSeriesChangeRule {ubec}{uc} {} {}

```

```

2460 <latexrelease>\DeclareFontSeriesChangeRule {ubec}{ul} {} {}
2461 <latexrelease>\DeclareFontSeriesChangeRule {ubec}{ux} {} {}
2462 <latexrelease>\DeclareFontSeriesChangeRule {ubec}{x} {} {}
2463 <latexrelease>\DeclareFontSeriesChangeRule {ubex}{?m} {} {}
2464 <latexrelease>\DeclareFontSeriesChangeRule {ubex}{bx} {} {}
2465 <latexrelease>\DeclareFontSeriesChangeRule {ubex}{b} {} {}
2466 <latexrelease>\DeclareFontSeriesChangeRule {ubex}{c} {} {}
2467 <latexrelease>\DeclareFontSeriesChangeRule {ubex}{eb} {} {}
2468 <latexrelease>\DeclareFontSeriesChangeRule {ubex}{ec} {} {}
2469 <latexrelease>\DeclareFontSeriesChangeRule {ubex}{el} {} {}
2470 <latexrelease>\DeclareFontSeriesChangeRule {ubex}{ex} {} {}
2471 <latexrelease>\DeclareFontSeriesChangeRule {ubex}{l} {} {}
2472 <latexrelease>\DeclareFontSeriesChangeRule {ubex}{m?} {} {}
2473 <latexrelease>\DeclareFontSeriesChangeRule {ubex}{sb} {} {}
2474 <latexrelease>\DeclareFontSeriesChangeRule {ubex}{sc} {} {}
2475 <latexrelease>\DeclareFontSeriesChangeRule {ubex}{sl} {} {}
2476 <latexrelease>\DeclareFontSeriesChangeRule {ubex}{sx} {} {}
2477 <latexrelease>\DeclareFontSeriesChangeRule {ubex}{ub} {} {}
2478 <latexrelease>\DeclareFontSeriesChangeRule {ubex}{uc} {} {}
2479 <latexrelease>\DeclareFontSeriesChangeRule {ubex}{ul} {} {}
2480 <latexrelease>\DeclareFontSeriesChangeRule {ubex}{ux} {} {}
2481 <latexrelease>\DeclareFontSeriesChangeRule {ubex}{x} {} {}
2482 <latexrelease>\DeclareFontSeriesChangeRule {ubsc}{bx} {} {}
2483 <latexrelease>\DeclareFontSeriesChangeRule {ubsc}{b} {} {}
2484 <latexrelease>\DeclareFontSeriesChangeRule {ubsc}{c} {} {}
2485 <latexrelease>\DeclareFontSeriesChangeRule {ubsc}{eb} {} {}
2486 <latexrelease>\DeclareFontSeriesChangeRule {ubsc}{ec} {} {}
2487 <latexrelease>\DeclareFontSeriesChangeRule {ubsc}{el} {} {}
2488 <latexrelease>\DeclareFontSeriesChangeRule {ubsc}{ex} {} {}
2489 <latexrelease>\DeclareFontSeriesChangeRule {ubsc}{l} {} {}
2490 <latexrelease>\DeclareFontSeriesChangeRule {ubsc}{sb} {} {}
2491 <latexrelease>\DeclareFontSeriesChangeRule {ubsc}{sc} {} {}
2492 <latexrelease>\DeclareFontSeriesChangeRule {ubsc}{sl} {} {}
2493 <latexrelease>\DeclareFontSeriesChangeRule {ubsc}{sx} {} {}
2494 <latexrelease>\DeclareFontSeriesChangeRule {ubsc}{ub} {} {}
2495 <latexrelease>\DeclareFontSeriesChangeRule {ubsc}{uc} {} {}
2496 <latexrelease>\DeclareFontSeriesChangeRule {ubsc}{ul} {} {}
2497 <latexrelease>\DeclareFontSeriesChangeRule {ubsc}{ux} {} {}
2498 <latexrelease>\DeclareFontSeriesChangeRule {ubsc}{x} {} {}
2499 <latexrelease>\DeclareFontSeriesChangeRule {ubsx}{?m} {} {}
2500 <latexrelease>\DeclareFontSeriesChangeRule {ubsx}{bx} {} {}
2501 <latexrelease>\DeclareFontSeriesChangeRule {ubsx}{b} {} {}
2502 <latexrelease>\DeclareFontSeriesChangeRule {ubsx}{c} {} {}
2503 <latexrelease>\DeclareFontSeriesChangeRule {ubsx}{eb} {} {}
2504 <latexrelease>\DeclareFontSeriesChangeRule {ubsx}{ec} {} {}
2505 <latexrelease>\DeclareFontSeriesChangeRule {ubsx}{el} {} {}
2506 <latexrelease>\DeclareFontSeriesChangeRule {ubsx}{ex} {} {}
2507 <latexrelease>\DeclareFontSeriesChangeRule {ubsx}{l} {} {}
2508 <latexrelease>\DeclareFontSeriesChangeRule {ubsx}{m?} {} {}
2509 <latexrelease>\DeclareFontSeriesChangeRule {ubsx}{sb} {} {}
2510 <latexrelease>\DeclareFontSeriesChangeRule {ubsx}{sc} {} {}
2511 <latexrelease>\DeclareFontSeriesChangeRule {ubsx}{sl} {} {}
2512 <latexrelease>\DeclareFontSeriesChangeRule {ubsx}{sx} {} {}
2513 <latexrelease>\DeclareFontSeriesChangeRule {ubsx}{ub} {} {}

```

```

2514 <latexrelease>\DeclareFontSeriesChangeRule {ubsx}{uc} {} {}
2515 <latexrelease>\DeclareFontSeriesChangeRule {ubsx}{ul} {} {}
2516 <latexrelease>\DeclareFontSeriesChangeRule {ubsx}{ux} {} {}
2517 <latexrelease>\DeclareFontSeriesChangeRule {ubsx}{x} {} {}
2518 <latexrelease>\DeclareFontSeriesChangeRule {ubuc}{?m} {} {}
2519 <latexrelease>\DeclareFontSeriesChangeRule {ubuc}{bx} {} {}
2520 <latexrelease>\DeclareFontSeriesChangeRule {ubuc}{b} {} {}
2521 <latexrelease>\DeclareFontSeriesChangeRule {ubuc}{c} {} {}
2522 <latexrelease>\DeclareFontSeriesChangeRule {ubuc}{eb} {} {}
2523 <latexrelease>\DeclareFontSeriesChangeRule {ubuc}{ec} {} {}
2524 <latexrelease>\DeclareFontSeriesChangeRule {ubuc}{el} {} {}
2525 <latexrelease>\DeclareFontSeriesChangeRule {ubuc}{ex} {} {}
2526 <latexrelease>\DeclareFontSeriesChangeRule {ubuc}{l} {} {}
2527 <latexrelease>\DeclareFontSeriesChangeRule {ubuc}{m?} {} {}
2528 <latexrelease>\DeclareFontSeriesChangeRule {ubuc}{sb} {} {}
2529 <latexrelease>\DeclareFontSeriesChangeRule {ubuc}{sc} {} {}
2530 <latexrelease>\DeclareFontSeriesChangeRule {ubuc}{sl} {} {}
2531 <latexrelease>\DeclareFontSeriesChangeRule {ubuc}{sx} {} {}
2532 <latexrelease>\DeclareFontSeriesChangeRule {ubuc}{ub} {} {}
2533 <latexrelease>\DeclareFontSeriesChangeRule {ubuc}{uc} {} {}
2534 <latexrelease>\DeclareFontSeriesChangeRule {ubuc}{ul} {} {}
2535 <latexrelease>\DeclareFontSeriesChangeRule {ubuc}{ux} {} {}
2536 <latexrelease>\DeclareFontSeriesChangeRule {ubuc}{x} {} {}
2537 <latexrelease>\DeclareFontSeriesChangeRule {ubux}{?m} {} {}
2538 <latexrelease>\DeclareFontSeriesChangeRule {ubux}{bx} {} {}
2539 <latexrelease>\DeclareFontSeriesChangeRule {ubux}{b} {} {}
2540 <latexrelease>\DeclareFontSeriesChangeRule {ubux}{c} {} {}
2541 <latexrelease>\DeclareFontSeriesChangeRule {ubux}{eb} {} {}
2542 <latexrelease>\DeclareFontSeriesChangeRule {ubux}{ec} {} {}
2543 <latexrelease>\DeclareFontSeriesChangeRule {ubux}{el} {} {}
2544 <latexrelease>\DeclareFontSeriesChangeRule {ubux}{ex} {} {}
2545 <latexrelease>\DeclareFontSeriesChangeRule {ubux}{l} {} {}
2546 <latexrelease>\DeclareFontSeriesChangeRule {ubux}{m?} {} {}
2547 <latexrelease>\DeclareFontSeriesChangeRule {ubux}{sb} {} {}
2548 <latexrelease>\DeclareFontSeriesChangeRule {ubux}{sc} {} {}
2549 <latexrelease>\DeclareFontSeriesChangeRule {ubux}{sl} {} {}
2550 <latexrelease>\DeclareFontSeriesChangeRule {ubux}{sx} {} {}
2551 <latexrelease>\DeclareFontSeriesChangeRule {ubux}{ub} {} {}
2552 <latexrelease>\DeclareFontSeriesChangeRule {ubux}{uc} {} {}
2553 <latexrelease>\DeclareFontSeriesChangeRule {ubux}{ul} {} {}
2554 <latexrelease>\DeclareFontSeriesChangeRule {ubux}{ux} {} {}
2555 <latexrelease>\DeclareFontSeriesChangeRule {ubux}{x} {} {}
2556 <latexrelease>\DeclareFontSeriesChangeRule {ubx}{bx} {} {}
2557 <latexrelease>\DeclareFontSeriesChangeRule {ubx}{b}{bx}{}
2558 <latexrelease>\DeclareFontSeriesChangeRule {ubx}{c}{ubc}{}
2559 <latexrelease>\DeclareFontSeriesChangeRule {ubx}{eb}{ebx}{}
2560 <latexrelease>\DeclareFontSeriesChangeRule {ubx}{ec}{ubec}{}
2561 <latexrelease>\DeclareFontSeriesChangeRule {ubx}{el}{elx}{}
2562 <latexrelease>\DeclareFontSeriesChangeRule {ubx}{ex} {} {}
2563 <latexrelease>\DeclareFontSeriesChangeRule {ubx}{l}{lx}{}
2564 <latexrelease>\DeclareFontSeriesChangeRule {ubx}{sb}{sbx}{}
2565 <latexrelease>\DeclareFontSeriesChangeRule {ubx}{sc}{ubsc}{}
2566 <latexrelease>\DeclareFontSeriesChangeRule {ubx}{sl}{slx}{}
2567 <latexrelease>\DeclareFontSeriesChangeRule {ubx}{sx} {} {}

```

```

2568 <latexrelease>\DeclareFontSeriesChangeRule {ubx}{ub}{ubx}{}
2569 <latexrelease>\DeclareFontSeriesChangeRule {ubx}{uc} {} {}
2570 <latexrelease>\DeclareFontSeriesChangeRule {ubx}{ul}{ulx}{}
2571 <latexrelease>\DeclareFontSeriesChangeRule {ubx}{ux} {} {}
2572 <latexrelease>\DeclareFontSeriesChangeRule {ubx}{x}{ubx}{}
2573 <latexrelease>\DeclareFontSeriesChangeRule {ub}{bx} {} {}
2574 <latexrelease>\DeclareFontSeriesChangeRule {ub}{c}{ubc}{}
2575 <latexrelease>\DeclareFontSeriesChangeRule {ub}{eb} {} {}
2576 <latexrelease>\DeclareFontSeriesChangeRule {ub}{ec}{ubec}{}
2577 <latexrelease>\DeclareFontSeriesChangeRule {ub}{ex} {} {}
2578 <latexrelease>\DeclareFontSeriesChangeRule {ub}{sb} {} {}
2579 <latexrelease>\DeclareFontSeriesChangeRule {ub}{sc}{ubsc}{}
2580 <latexrelease>\DeclareFontSeriesChangeRule {ub}{sx} {} {}
2581 <latexrelease>\DeclareFontSeriesChangeRule {ub}{uc} {} {}
2582 <latexrelease>\DeclareFontSeriesChangeRule {ub}{ux} {} {}
2583 <latexrelease>\DeclareFontSeriesChangeRule {ub}{x}{ubx}{}
2584 <latexrelease>\DeclareFontSeriesChangeRule {uc}{?m} {} {}
2585 <latexrelease>\DeclareFontSeriesChangeRule {uc}{bx} {} {}
2586 <latexrelease>\DeclareFontSeriesChangeRule {uc}{b} {} {}
2587 <latexrelease>\DeclareFontSeriesChangeRule {uc}{eb} {} {}
2588 <latexrelease>\DeclareFontSeriesChangeRule {uc}{el} {} {}
2589 <latexrelease>\DeclareFontSeriesChangeRule {uc}{l} {} {}
2590 <latexrelease>\DeclareFontSeriesChangeRule {uc}{m?} {} {}
2591 <latexrelease>\DeclareFontSeriesChangeRule {uc}{sb} {} {}
2592 <latexrelease>\DeclareFontSeriesChangeRule {uc}{sl} {} {}
2593 <latexrelease>\DeclareFontSeriesChangeRule {uc}{ub} {} {}
2594 <latexrelease>\DeclareFontSeriesChangeRule {uc}{ul} {} {}
2595 <latexrelease>\DeclareFontSeriesChangeRule {ulc}{bx} {} {}
2596 <latexrelease>\DeclareFontSeriesChangeRule {ulc}{b}{bc}{}
2597 <latexrelease>\DeclareFontSeriesChangeRule {ulc}{c}{ulc}{}
2598 <latexrelease>\DeclareFontSeriesChangeRule {ulc}{eb}{ebc}{}
2599 <latexrelease>\DeclareFontSeriesChangeRule {ulc}{ec}{ulec}{ulc}
2600 <latexrelease>\DeclareFontSeriesChangeRule {ulc}{el}{elc}{}
2601 <latexrelease>\DeclareFontSeriesChangeRule {ulc}{ex} {} {}
2602 <latexrelease>\DeclareFontSeriesChangeRule {ulc}{l}{lc}{}
2603 <latexrelease>\DeclareFontSeriesChangeRule {ulc}{sb}{sbc}{}
2604 <latexrelease>\DeclareFontSeriesChangeRule {ulc}{sc}{ulsc}{ulc}
2605 <latexrelease>\DeclareFontSeriesChangeRule {ulc}{sl}{slc}{}
2606 <latexrelease>\DeclareFontSeriesChangeRule {ulc}{sx} {} {}
2607 <latexrelease>\DeclareFontSeriesChangeRule {ulc}{ub}{ubc}{}
2608 <latexrelease>\DeclareFontSeriesChangeRule {ulc}{uc} {} {}
2609 <latexrelease>\DeclareFontSeriesChangeRule {ulc}{ul}{ulc}{}
2610 <latexrelease>\DeclareFontSeriesChangeRule {ulc}{ux} {} {}
2611 <latexrelease>\DeclareFontSeriesChangeRule {ulc}{x}{ulx}{}
2612 <latexrelease>\DeclareFontSeriesChangeRule {ulec}{bx} {} {}
2613 <latexrelease>\DeclareFontSeriesChangeRule {ulec}{b} {} {}
2614 <latexrelease>\DeclareFontSeriesChangeRule {ulec}{c} {} {}
2615 <latexrelease>\DeclareFontSeriesChangeRule {ulec}{eb} {} {}
2616 <latexrelease>\DeclareFontSeriesChangeRule {ulec}{ec} {} {}
2617 <latexrelease>\DeclareFontSeriesChangeRule {ulec}{el} {} {}
2618 <latexrelease>\DeclareFontSeriesChangeRule {ulec}{ex} {} {}
2619 <latexrelease>\DeclareFontSeriesChangeRule {ulec}{l} {} {}
2620 <latexrelease>\DeclareFontSeriesChangeRule {ulec}{sb} {} {}
2621 <latexrelease>\DeclareFontSeriesChangeRule {ulec}{sc} {} {}

```



```

2622 <latexrelease>\DeclareFontSeriesChangeRule {ulec}{sl} {} {}
2623 <latexrelease>\DeclareFontSeriesChangeRule {ulec}{sx} {} {}
2624 <latexrelease>\DeclareFontSeriesChangeRule {ulec}{ub} {} {}
2625 <latexrelease>\DeclareFontSeriesChangeRule {ulec}{uc} {} {}
2626 <latexrelease>\DeclareFontSeriesChangeRule {ulec}{ul} {} {}
2627 <latexrelease>\DeclareFontSeriesChangeRule {ulec}{ux} {} {}
2628 <latexrelease>\DeclareFontSeriesChangeRule {ulec}{x} {} {}
2629 <latexrelease>\DeclareFontSeriesChangeRule {ulex}{?m} {} {}
2630 <latexrelease>\DeclareFontSeriesChangeRule {ulex}{bx} {} {}
2631 <latexrelease>\DeclareFontSeriesChangeRule {ulex}{b} {} {}
2632 <latexrelease>\DeclareFontSeriesChangeRule {ulex}{c} {} {}
2633 <latexrelease>\DeclareFontSeriesChangeRule {ulex}{eb} {} {}
2634 <latexrelease>\DeclareFontSeriesChangeRule {ulex}{ec} {} {}
2635 <latexrelease>\DeclareFontSeriesChangeRule {ulex}{el} {} {}
2636 <latexrelease>\DeclareFontSeriesChangeRule {ulex}{ex} {} {}
2637 <latexrelease>\DeclareFontSeriesChangeRule {ulex}{l} {} {}
2638 <latexrelease>\DeclareFontSeriesChangeRule {ulex}{m?} {} {}
2639 <latexrelease>\DeclareFontSeriesChangeRule {ulex}{sb} {} {}
2640 <latexrelease>\DeclareFontSeriesChangeRule {ulex}{sc} {} {}
2641 <latexrelease>\DeclareFontSeriesChangeRule {ulex}{sl} {} {}
2642 <latexrelease>\DeclareFontSeriesChangeRule {ulex}{sx} {} {}
2643 <latexrelease>\DeclareFontSeriesChangeRule {ulex}{ub} {} {}
2644 <latexrelease>\DeclareFontSeriesChangeRule {ulex}{uc} {} {}
2645 <latexrelease>\DeclareFontSeriesChangeRule {ulex}{ul} {} {}
2646 <latexrelease>\DeclareFontSeriesChangeRule {ulex}{ux} {} {}
2647 <latexrelease>\DeclareFontSeriesChangeRule {ulex}{x} {} {}
2648 <latexrelease>\DeclareFontSeriesChangeRule {ulsc}{bx} {} {}
2649 <latexrelease>\DeclareFontSeriesChangeRule {ulsc}{b} {} {}
2650 <latexrelease>\DeclareFontSeriesChangeRule {ulsc}{c} {} {}
2651 <latexrelease>\DeclareFontSeriesChangeRule {ulsc}{eb} {} {}
2652 <latexrelease>\DeclareFontSeriesChangeRule {ulsc}{ec} {} {}
2653 <latexrelease>\DeclareFontSeriesChangeRule {ulsc}{el} {} {}
2654 <latexrelease>\DeclareFontSeriesChangeRule {ulsc}{ex} {} {}
2655 <latexrelease>\DeclareFontSeriesChangeRule {ulsc}{l} {} {}
2656 <latexrelease>\DeclareFontSeriesChangeRule {ulsc}{sb} {} {}
2657 <latexrelease>\DeclareFontSeriesChangeRule {ulsc}{sc} {} {}
2658 <latexrelease>\DeclareFontSeriesChangeRule {ulsc}{sl} {} {}
2659 <latexrelease>\DeclareFontSeriesChangeRule {ulsc}{sx} {} {}
2660 <latexrelease>\DeclareFontSeriesChangeRule {ulsc}{ub} {} {}
2661 <latexrelease>\DeclareFontSeriesChangeRule {ulsc}{uc} {} {}
2662 <latexrelease>\DeclareFontSeriesChangeRule {ulsc}{ul} {} {}
2663 <latexrelease>\DeclareFontSeriesChangeRule {ulsc}{ux} {} {}
2664 <latexrelease>\DeclareFontSeriesChangeRule {ulsc}{x} {} {}
2665 <latexrelease>\DeclareFontSeriesChangeRule {ulsx}{?m} {} {}
2666 <latexrelease>\DeclareFontSeriesChangeRule {ulsx}{bx} {} {}
2667 <latexrelease>\DeclareFontSeriesChangeRule {ulsx}{b} {} {}
2668 <latexrelease>\DeclareFontSeriesChangeRule {ulsx}{c} {} {}
2669 <latexrelease>\DeclareFontSeriesChangeRule {ulsx}{eb} {} {}
2670 <latexrelease>\DeclareFontSeriesChangeRule {ulsx}{ec} {} {}
2671 <latexrelease>\DeclareFontSeriesChangeRule {ulsx}{el} {} {}
2672 <latexrelease>\DeclareFontSeriesChangeRule {ulsx}{ex} {} {}
2673 <latexrelease>\DeclareFontSeriesChangeRule {ulsx}{l} {} {}
2674 <latexrelease>\DeclareFontSeriesChangeRule {ulsx}{m?} {} {}
2675 <latexrelease>\DeclareFontSeriesChangeRule {ulsx}{sb} {} {}

```

```

2676 <latexrelease>\DeclareFontSeriesChangeRule {ulsx}{sc} {} {}
2677 <latexrelease>\DeclareFontSeriesChangeRule {ulsx}{sl} {} {}
2678 <latexrelease>\DeclareFontSeriesChangeRule {ulsx}{sx} {} {}
2679 <latexrelease>\DeclareFontSeriesChangeRule {ulsx}{ub} {} {}
2680 <latexrelease>\DeclareFontSeriesChangeRule {ulsx}{uc} {} {}
2681 <latexrelease>\DeclareFontSeriesChangeRule {ulsx}{ul} {} {}
2682 <latexrelease>\DeclareFontSeriesChangeRule {ulsx}{ux} {} {}
2683 <latexrelease>\DeclareFontSeriesChangeRule {ulsx}{x} {} {}
2684 <latexrelease>\DeclareFontSeriesChangeRule {uluc}{?m} {} {}
2685 <latexrelease>\DeclareFontSeriesChangeRule {uluc}{bx} {} {}
2686 <latexrelease>\DeclareFontSeriesChangeRule {uluc}{b} {} {}
2687 <latexrelease>\DeclareFontSeriesChangeRule {uluc}{c} {} {}
2688 <latexrelease>\DeclareFontSeriesChangeRule {uluc}{eb} {} {}
2689 <latexrelease>\DeclareFontSeriesChangeRule {uluc}{ec} {} {}
2690 <latexrelease>\DeclareFontSeriesChangeRule {uluc}{el} {} {}
2691 <latexrelease>\DeclareFontSeriesChangeRule {uluc}{ex} {} {}
2692 <latexrelease>\DeclareFontSeriesChangeRule {uluc}{l} {} {}
2693 <latexrelease>\DeclareFontSeriesChangeRule {uluc}{m?} {} {}
2694 <latexrelease>\DeclareFontSeriesChangeRule {uluc}{sb} {} {}
2695 <latexrelease>\DeclareFontSeriesChangeRule {uluc}{sc} {} {}
2696 <latexrelease>\DeclareFontSeriesChangeRule {uluc}{sl} {} {}
2697 <latexrelease>\DeclareFontSeriesChangeRule {uluc}{sx} {} {}
2698 <latexrelease>\DeclareFontSeriesChangeRule {uluc}{ub} {} {}
2699 <latexrelease>\DeclareFontSeriesChangeRule {uluc}{uc} {} {}
2700 <latexrelease>\DeclareFontSeriesChangeRule {uluc}{ul} {} {}
2701 <latexrelease>\DeclareFontSeriesChangeRule {uluc}{ux} {} {}
2702 <latexrelease>\DeclareFontSeriesChangeRule {uluc}{x} {} {}
2703 <latexrelease>\DeclareFontSeriesChangeRule {ulux}{?m} {} {}
2704 <latexrelease>\DeclareFontSeriesChangeRule {ulux}{bx} {} {}
2705 <latexrelease>\DeclareFontSeriesChangeRule {ulux}{b} {} {}
2706 <latexrelease>\DeclareFontSeriesChangeRule {ulux}{c} {} {}
2707 <latexrelease>\DeclareFontSeriesChangeRule {ulux}{eb} {} {}
2708 <latexrelease>\DeclareFontSeriesChangeRule {ulux}{ec} {} {}
2709 <latexrelease>\DeclareFontSeriesChangeRule {ulux}{el} {} {}
2710 <latexrelease>\DeclareFontSeriesChangeRule {ulux}{ex} {} {}
2711 <latexrelease>\DeclareFontSeriesChangeRule {ulux}{l} {} {}
2712 <latexrelease>\DeclareFontSeriesChangeRule {ulux}{m?} {} {}
2713 <latexrelease>\DeclareFontSeriesChangeRule {ulux}{sb} {} {}
2714 <latexrelease>\DeclareFontSeriesChangeRule {ulux}{sc} {} {}
2715 <latexrelease>\DeclareFontSeriesChangeRule {ulux}{sl} {} {}
2716 <latexrelease>\DeclareFontSeriesChangeRule {ulux}{sx} {} {}
2717 <latexrelease>\DeclareFontSeriesChangeRule {ulux}{ub} {} {}
2718 <latexrelease>\DeclareFontSeriesChangeRule {ulux}{uc} {} {}
2719 <latexrelease>\DeclareFontSeriesChangeRule {ulux}{ul} {} {}
2720 <latexrelease>\DeclareFontSeriesChangeRule {ulux}{ux} {} {}
2721 <latexrelease>\DeclareFontSeriesChangeRule {ulux}{x} {} {}
2722 <latexrelease>\DeclareFontSeriesChangeRule {ulx}{bx} {} {}
2723 <latexrelease>\DeclareFontSeriesChangeRule {ulx}{b}{bx}{}
2724 <latexrelease>\DeclareFontSeriesChangeRule {ulx}{c}{ulc}{}
2725 <latexrelease>\DeclareFontSeriesChangeRule {ulx}{eb}{ebx}{}
2726 <latexrelease>\DeclareFontSeriesChangeRule {ulx}{ec}{ulec}{}
2727 <latexrelease>\DeclareFontSeriesChangeRule {ulx}{el}{elx}{}
2728 <latexrelease>\DeclareFontSeriesChangeRule {ulx}{ex} {} {}
2729 <latexrelease>\DeclareFontSeriesChangeRule {ulx}{l}{lx}{}

```

```

2730 <latexrelease>\DeclareFontSeriesChangeRule {ulx}{sb}{sbx}{}
2731 <latexrelease>\DeclareFontSeriesChangeRule {ulx}{sc}{ulsc}{}
2732 <latexrelease>\DeclareFontSeriesChangeRule {ulx}{sl}{slx}{}
2733 <latexrelease>\DeclareFontSeriesChangeRule {ulx}{sx} {} {}
2734 <latexrelease>\DeclareFontSeriesChangeRule {ulx}{ub}{ubx}{}
2735 <latexrelease>\DeclareFontSeriesChangeRule {ulx}{uc} {} {}
2736 <latexrelease>\DeclareFontSeriesChangeRule {ulx}{ul}{ulx}{}
2737 <latexrelease>\DeclareFontSeriesChangeRule {ulx}{ux} {} {}
2738 <latexrelease>\DeclareFontSeriesChangeRule {ulx}{x}{ulx}{}
2739 <latexrelease>\DeclareFontSeriesChangeRule {ul}{bx} {} {}
2740 <latexrelease>\DeclareFontSeriesChangeRule {ul}{c}{ulc}{}
2741 <latexrelease>\DeclareFontSeriesChangeRule {ul}{eb} {} {}
2742 <latexrelease>\DeclareFontSeriesChangeRule {ul}{ec}{ulec}{}
2743 <latexrelease>\DeclareFontSeriesChangeRule {ul}{ex} {} {}
2744 <latexrelease>\DeclareFontSeriesChangeRule {ul}{sb} {} {}
2745 <latexrelease>\DeclareFontSeriesChangeRule {ul}{sc}{ulsc}{}
2746 <latexrelease>\DeclareFontSeriesChangeRule {ul}{sx} {} {}
2747 <latexrelease>\DeclareFontSeriesChangeRule {ul}{ub} {} {}
2748 <latexrelease>\DeclareFontSeriesChangeRule {ul}{uc} {} {}
2749 <latexrelease>\DeclareFontSeriesChangeRule {ul}{ux} {} {}
2750 <latexrelease>\DeclareFontSeriesChangeRule {ul}{x}{ulx}{}
2751 <latexrelease>\DeclareFontSeriesChangeRule {ux}{?m} {} {}
2752 <latexrelease>\DeclareFontSeriesChangeRule {ux}{bx} {} {}
2753 <latexrelease>\DeclareFontSeriesChangeRule {ux}{b} {} {}
2754 <latexrelease>\DeclareFontSeriesChangeRule {ux}{eb} {} {}
2755 <latexrelease>\DeclareFontSeriesChangeRule {ux}{el} {} {}
2756 <latexrelease>\DeclareFontSeriesChangeRule {ux}{l} {} {}
2757 <latexrelease>\DeclareFontSeriesChangeRule {ux}{m?} {} {}
2758 <latexrelease>\DeclareFontSeriesChangeRule {ux}{sb} {} {}
2759 <latexrelease>\DeclareFontSeriesChangeRule {ux}{sl} {} {}
2760 <latexrelease>\DeclareFontSeriesChangeRule {ux}{ub} {} {}
2761 <latexrelease>\DeclareFontSeriesChangeRule {ux}{ul} {} {}
2762 <latexrelease>\DeclareFontSeriesChangeRule {x}{bx} {} {}
2763 <latexrelease>\DeclareFontSeriesChangeRule {x}{c}{c}{}
2764 <latexrelease>\DeclareFontSeriesChangeRule {x}{ec}{ec}{}
2765 <latexrelease>\DeclareFontSeriesChangeRule {x}{el}{elx}{}
2766 <latexrelease>\DeclareFontSeriesChangeRule {x}{l}{lx}{}
2767 <latexrelease>\DeclareFontSeriesChangeRule {x}{sc}{sc}{}
2768 <latexrelease>\DeclareFontSeriesChangeRule {x}{sl}{slx}{}
2769 <latexrelease>\DeclareFontSeriesChangeRule {x}{ub}{ubx}{}
2770 <latexrelease>\DeclareFontSeriesChangeRule {x}{ul}{ulx}{}
2771 <latexrelease>
2772 <latexrelease>\EndIncludeInRelease

2773 <latexrelease>\IncludeInRelease{0000/00/00}%
2774 <latexrelease> {\DeclareFontSeriesChangeRule}{Series change rules}%
2775 <latexrelease>
2776 <latexrelease>\let\DeclareFontSeriesChangeRule\@undefined
2777 <latexrelease>
2778 <latexrelease>\EndIncludeInRelease

```

1.3 Changing to a new series

```

2779 <*2ekernel | latexrelease>
2780 <latexrelease>\IncludeInRelease{2021/06/01}%

```

2781 <latexrelease> {\fontseries}{delay fontseries update}%

\fontseries The `\fontseries` command takes one argument which is the requested new font series. In the original implementation it simply saved the expanded value in `\f@series`. Now we do a bit more processing and look up the final value in the font series data base. This is done by `\merge@font@series`. But the lookup should be done within the target family and call to `\fontseries` might be followed by a `\fontfamily` call. So we delay the processing to `\selectfont` and only record the necessary action in `\delayed@f@adjustment`.

```
2782 \DeclareRobustCommand\fontseries[1]{\@forced@seriesfalse
2783   \expandafter\def\expandafter\delayed@f@adjustment\expandafter
2784     {\delayed@f@adjustment\delayed@merge@font@series{#1}}}
```

(End of definition for `\fontseries`.)

\delayed@f@adjustment The macro holding the delayed action(s) for use in `\selectfont`.

```
2785 \let\delayed@f@adjustment\@empty
```

(End of definition for `\delayed@f@adjustment`.)

\fontseriesforce To change unconditionally to a new series you can use `\fontseriesforce`. Of course, if the series doesn't exist for the current family substitution still happens, but there is not dependency on the current series.

```
2786 \DeclareRobustCommand\fontseriesforce[1]{\@forced@seriestrue
2787   \expandafter\def\expandafter\delayed@f@adjustment\expandafter
2788     {\delayed@f@adjustment\edef\f@series{#1}}}
```

(End of definition for `\fontseriesforce`.)

\if@forced@series If the series gets forced we need to know that fact later on.

```
2789 \newif\if@forced@series
```

(End of definition for `\if@forced@series`.)

```
2790 </2ekernel | latexrelease>
```

```
2791 <latexrelease>\EndIncludeInRelease
```

```
2792 <latexrelease>\IncludeInRelease{2020/02/02}%
```

```
2793 <latexrelease>                   {\fontseries}{delay fontseries update}%
```

```
2794 <latexrelease>
```

```
2795 <latexrelease>\DeclareRobustCommand\fontseries[1]{\@forced@seriesfalse
```

```
2796 <latexrelease>                                           \merge@font@series{#1}}
```

```
2797 <latexrelease>\DeclareRobustCommand\fontseriesforce[1]{\@forced@seriestrue
```

```
2798 <latexrelease>                                           \edef\f@series{#1}}
```

```
2799 <latexrelease>\let\delayed@f@adjustment\@undefined
```

```
2800 <latexrelease>
```

For a roll forward we may have to define `\if@forced@series` but this needs doing in a way that \TeX doesn't see it when skipping over conditionals.

```
2801 <latexrelease>\expandafter\newif\csname if@forced@series\endcsname
```

```
2802 <latexrelease>
```

```
2803 <latexrelease>\EndIncludeInRelease
```

```

2804 <latexrelease>\IncludeInRelease{0000/00/00}%
2805 <latexrelease>          {\fontseries}{delay fontseries update}%
2806 <latexrelease>
2807 <latexrelease>\DeclareRobustCommand\fontseries[1]{\edef\f@series{#1}}
2808 <latexrelease>\let\fontseriesforce\@undefined
2809 <latexrelease>
2810 <latexrelease>\EndIncludeInRelease
2811 <*2ekernel | latexrelease>
2812 <latexrelease>\IncludeInRelease{2020/02/02}%
2813 <latexrelease>  {\merge@font@series}{Merge series values}%

```

`\merge@font@series` We look up the data base value by expanding the right command twice. If no such value exist then the result will be `\relax` otherwise it will be the two brace groups: the desired result and the alternate result. The first case means that the third argument to `\merge@font@series` will be empty.

```

2814 \def\merge@font@series#1{%
2815   \expandafter\expandafter\expandafter
2816   \merge@font@series@
2817   \csname series@\f@series @#1\endcsname
2818   {#1}%
2819   \@nil
2820 }

```

(End of definition for \merge@font@series.)

`\merge@font@series@` This now defines the new `\f@series`:

```

2821 \def\merge@font@series@#1#2#3\@nil{%

```

If the third argument is empty there is no database entry for the combination and the second argument holds the new series so we return that.

Originally the test was simply `\ifx!#3!` but that actually dies if `#3` starts with a conditional and in the definition of `\AmSfont` that is actually the case.

```

2822 %\ifcat\expandafter X\detokenize{#1}X%
2823 \def\reserved@a{#3}%
2824 \ifx\reserved@a\@empty
2825   \set@target@series{#2}%
2826 \else

```

Otherwise we check if the desired result for the series (`#1`) exists for the font family and the current shape. All this happens inside `\selectfont` which has already taken care to load the `.fd`, file if necessary.

```

2827 \edef\reserved@a{\f@encoding /\f@family /#1/\f@shape}%
2828 \ifcsname \reserved@a \endcsname

```

If the desired result is available then we use that. However, we do need some post-processing because we need to drop surplus `ms` due to the way naming convention was designed in the '90s (sigh).

```

2829   \set@target@series{#1}%

```

If not, then we try the alternate result (`#2`).

```

2830 \else
2831   \ifcsname \f@encoding /\f@family /#2/\f@shape \endcsname

```

If the alternate result exist we use that and also issue a warning (or rather a log entry) that we didn't managed to change to the desired font.

```
2832         \set@target@series{#2}%
2833         \@font@shape@subst@warning
```

If that doesn't exist either, then we use the requested series unmodified (again with a warning).

```
2834         \else
2835         \set@target@series{#3}%
2836         \@font@shape@subst@warning
2837     \fi
2838 \fi
2839 \fi
2840 }
```

It is possible that the previous font and the new one are actually identical (and the font was not found because it still needs loading) in which case a warning would look rather odd. So we make a quick check for that (which is the reason why we defined `\@reserveda` above instead of doing inline testing inside `\ifcstype`).

```
2841 \def\@font@shape@subst@warning{%
2842     \edef\reserved@b{\curr@fontshape}%
2843     \ifx\reserved@a\reserved@b \else
2844         \@font@warning{Font shape '\reserved@a' undefined\MessageBreak
2845             using '\reserved@b' instead}%
2846     \fi
2847 }
```

(End of definition for `\merge@font@series@`.)

```
\merge@font@series@without@substitution
\merge@font@series@without@substitution@
\delayed@merge@font@series
```

`\merge@font@series@without@substitution` works like `\merge@font@series`, i.e., it looks up the combination in the rule base and if there exists an entry it uses it and if not it uses the new series value. However, it doesn't check if there is actually a font face with the new series value as `\merge@font@series` does. This simplified command is used in `\selectfont` at a point where other font attributes are not yet updated so that checking the font face might result incorrect in substitutions.

```
2848 \def\merge@font@series@without@substitution#1{%
2849     \expandafter\expandafter\expandafter
2850     \merge@font@series@without@substitution@
2851     \cstype series@\f@series @#1\endcstype
2852     {#1}%
2853     \@nil
2854 }
2855 \def\merge@font@series@without@substitution@#1#2#3\@nil{%
2856     \def\reserved@a{#3}%
2857     \ifx\reserved@a@empty
2858         \set@target@series{#2}%
2859     \else
2860         \set@target@series{#1}%
2861     \fi
2862 }
```

(End of definition for `\merge@font@series@without@substitution`, `\merge@font@series@without@substitution@`, and `\delayed@merge@font@series`.)

`\delayed@merge@font@series` When we delay the merge action in `\fontseries` we first attempt to use merging without substitution. If that results in a non-existing font face the merge is redone in `\selectfont` using a version with substitution. See `\selectfont` for details.

```
2863 \let\delayed@merge@font@series\merge@font@series@without@substitution
```

(End of definition for \delayed@merge@font@series.)

`\maybe@load@fontshape` A small helper that we use a couple of times: try loading a fontshape (in a group because `\try@load@fontshape` normalizes catcodes and we also want to change `\typeout` so that it doesn't report missing .fd files on the terminal).

```
2864 \def\maybe@load@fontshape{%
2865   \begingroup
2866   \let \typeout \@font@info
2867   \try@load@fontshape
2868   \endgroup}
```

(End of definition for \maybe@load@fontshape.)

`\set@target@series` Finally the code for normalizing the `\f@series` value.

The combined series value determined by the mapping may still contain an `m` that we have to remove (as the .fd files use `c` not `mc` to denote a medium weight condensed series, etc.). We do this in all branches above because a user might have written

```
\DeclareFontSeriesChangeRule {m}{sc}{msc}{mc}
```

instead of using `sc` and `c` as needed in the .fd file.

```
2869 \def\set@target@series#1{%
```

We need to `\edef` the argument first in case it starts with a conditional. Then we check (and perhaps drop) an “m” from the value and assign the result to `\f@series`.

```
2870   \edef\f@series{#1}%
2871   \series@maybe@drop@one@m\f@series\f@series
2872 }
```

(End of definition for \set@target@series.)

`\series@maybe@drop@one@m` If the series value is in NFSS notation then it should not contain any “m” unless it is just an “m” by its own. So we need to drop surplus “m”s. But we better don't do this for full names, such as “semibold” as used by `autoinst`, for example. So we test against the possible explicit values that should drop an “m”. After that we assign the result to `#2` for further use.

```
2873 \def\series@maybe@drop@one@m#1{%
2874   \expandafter\series@maybe@drop@one@m@x\expandafter{#1}}
```

```
2875
```

```
2876 \def\series@maybe@drop@one@m@x#1#2{%
```

The code below is an inline version of the `\in@` macro without the group, so that it works in `\accent`.

```
2877   \def\in@@##1,#1,{}%
2878   \series@check@toks\expandafter{\in@@
2879     ,ulm,elm,lm,slm,mm,sbm,bm,ebm,ubm,muc,mec,mc,msc,msx,mx,mex,mux,{}},#1,}%
2880   \edef\in@{\the\series@check@toks}%
2881   \ifx\in@@\@empty
```

The default definition for `\bfdefault` etc is actually `b\empty` so that we can detect if the user has changed the default. However that means a) the above test will definitely fail (maybe something to change) and b) we better use `\edef` on the next line to get rid of it as otherwise the test against `#2` (e.g., `\bfdef@ult`) will fail in other places.

```
2882   \edef#2{#1}%
2883   \else
2884   \edef#2{\expandafter\series@drop@one@m #1\series@drop@one@m}%
2885   \fi
2886 }
```

As a precaution we use a private toks register not `\toks@` as that is no longer hidden inside the group.

```
2887 \newtoks\series@check@toks
```

(End of definition for \series@maybe@drop@one@m.)

`\series@drop@one@m` Drop up to two ms but keep one if that makes the series value empty. Actually, with the current implementation we know that there is at least one in the series value itself and we added one after it, so all we have to do is now returning `#1#2` and dropping the rest.

```
2888 \def\series@drop@one@m#1#2#3\series@drop@one@m{%
2889 % \ifx\relax#1#2\relax m\else#1#2\fi
2890 #1#2%
2891 }
```

(End of definition for \series@drop@one@m.)

Supporting rollback ...

```
2892 </2ekernel | latexrelease>
2893 <latexrelease>\EndIncludeInRelease
2894 <latexrelease>\IncludeInRelease{0000/00/00}%
2895 <latexrelease> {\merge@font@series}{Merge series values}%
2896 <latexrelease>
2897 <latexrelease>\let\merge@font@series\@undefined
2898 <latexrelease>\let\merge@font@series@\@undefined
2899 <latexrelease>\let\@font@shape@subst@warning\@undefined
2900 <latexrelease>\let\merge@font@series@without@substitution\@undefined
2901 <latexrelease>\let\merge@font@series@without@substitution@\@undefined
2902 <latexrelease>\let\delayed@merge@font@series\@undefined
2903 <latexrelease>\let\maybe@load@fontshape\@undefined
2904 <latexrelease>\let\set@target@series\@undefined
2905 <latexrelease>\let\series@maybe@drop@one@m\@undefined
2906 <latexrelease>\let\series@drop@one@m\@undefined
2907 <latexrelease>
2908 <latexrelease>\EndIncludeInRelease
```

2 Changing the shape

Shapes are also split in two axes (though it could be more if that is desirable), essentially building in an “sc” axis.

```
2909 <*2ekernel | latexrelease>
2910 <latexrelease>\IncludeInRelease{2020/02/02}%
2911 <latexrelease> {\ulcshape}{Font shape change rules}%
```


`\DeclareFontShapeChangeRule` The database for shapes is done in exactly the same way, only that it is much smaller and we usually have no alternative shape (or rather it is empty thus not used).

```
2912 \def\DeclareFontShapeChangeRule #1#2#3#4{%
2913   \@namedef{shape@#1@#2}{#3}{#4}}
```

(End of definition for \DeclareFontShapeChangeRule.)

There is kind of the same problem with returning back from `sc` to normal. It sort of needs its own letter. In `fontspec` this was solved by the first time `\upshape` changes `it` or `sl` back (so only `sc` remains) and second time it changes then `sc` back to normal. Maybe that's not a bad way to handle it, but decided for a slightly different approach: `n` always returns to "normal", ie resets everything and `up` changes italic or slanted to upright and `ulc` undoes small caps.

So we now offer `\normalshape` (using `\shapedefault`) which is normally the same as calling both `\ulcshape` and `\upshape`, only more efficient.

`\ulcshape` To request going back to upper/lowercase we need a new command. It uses `ulc` as shape name but this shape is virtual, i.e., it doesn't exist as a real shape, it is only used as part of the database table entries and thus only appears in the second argument there (but not in the first).

```
2914 \DeclareRobustCommand\ulcshape
2915   {\not@math@alphabet\ulcshape\relax
2916   \fontshape\ulcdefault\selectfont}
2917 \let\ulcdefault\@undefined % for rollback
2918 \newcommand\ulcdefault{ulc}
```

(End of definition for \ulcshape, \textulc, and \ulcdefault.)

`\swshape` New command to select a swash shape. The standard rules put this in the same category as italics or slanted, i.e., if you ask for it then italics are undone. One could provide more complicated rules so that `it + sw` becomes `swit` but given that there are only very few fonts that have swash letters that level of flexibility (these days) would be just resulting in a lot of combinations that do not exist.

```
2919 \DeclareRobustCommand\swshape
2920   {\not@math@alphabet\swshape\relax
2921   \fontshape\swdefault\selectfont}
2922 \let\swdefault\@undefined % for rollback
2923 \newcommand\swdefault{sw}
```

(End of definition for \swshape, \textsw, and \swdefault.)

`\sscshape` New commands to select spaced small capitals. There isn't a single free font that supports it. However, some commercial ones do, so we offer it. It is also possible to produce spaced small capitals from normal small capitals in OTF fonts using `otftotfm` with calls such as

```
otftotfm -e TEXMF/fonts/enc/dvips/base/texnansx.enc \
SourceSerifPro-Regular.otf -fkern -fliga --feature=smcp \
--letterspacing=80 SourceSerifPro-Regular-ssc-LY1
```

Michael Ummels kindly prepared the necessary rules for `ssc` ages ago, but until recently they managed to hide deep down in my inbox. I finally got around to integrating them (with a few changes) and I also took the opportunity to rationalize (a bit) the rules for the only other uncommon shape, `sw`.

```

2924 \DeclareRobustCommand\sscshape
2925     {\not@math@alphabet\sscshape\relax
2926     \fontshape\sscdefault\selectfont}
2927 \let\sscdefault\@undefined      % for rollback
2928 \newcommand\sscdefault{ssc}

(End of definition for \sscshape, \textssc, and \sscdefault.)
    Supporting rollback ...

2929 </2ekernel | latexrelease>
2930 <latexrelease>\EndIncludeInRelease
2931 <latexrelease>\IncludeInRelease{0000/00/00}%
2932 <latexrelease>    {\ulcshape}{Font shape change rules}%
2933 <latexrelease>
2934 <latexrelease>\let\DeclareFontShapeChangeRule\@undefined
2935 <latexrelease>\let\ulcshape\@undefined
2936 <latexrelease>\let\ulcdefault\@undefined
2937 <latexrelease>\let\swshape\@undefined
2938 <latexrelease>\let\swdefault\@undefined
2939 <latexrelease>\let\sscshape\@undefined
2940 <latexrelease>\let\sscdefault\@undefined
2941 <latexrelease>
2942 <latexrelease>\EndIncludeInRelease
2943 <*2ekernel>

```

2.1 Mapping rules for shape combinations

Many of the entries are commented out as we will get that result without any entry.

```

2944 </2ekernel>
2945 <*2ekernel | latexrelease>
2946 <latexrelease>\IncludeInRelease{2025/06/01}%
2947 <latexrelease>    {\DeclareFontShapeChangeRule}{Rules for ssc and sw}%
2948 %
2949 % \DeclareFontShapeChangeRule {n}{n}    {n}    {}
2950 \DeclareFontShapeChangeRule {n}{it}    {it}    {sl}
2951 \DeclareFontShapeChangeRule {n}{sl}    {sl}    {it}

```

If `sw` is requested but not available (not many font families offer it) we try to fallback to `it` instead of `sw`. That isn't always perfect, because some swash shapes are actually upright, but it is only a fallback and most of the time it would be better than `n`.

```

2952 \DeclareFontShapeChangeRule {n}{sw}    {sw}    {it}
2953 % \DeclareFontShapeChangeRule {n}{sc}    {sc}    {}
2954 \DeclareFontShapeChangeRule {n}{ulc}    {n}    {}
2955 \DeclareFontShapeChangeRule {n}{up}    {n}    {}

```

For the `ssc` shape we make the following general assumptions: if `ssc(X)` exists then `sc(X)` and `X` also exist in the font. If the `ssc(X)` shape doesn't exist but the user had asked for `ssc` we try to replace it by `sc` because we then assume that the current font family simply doesn't have any `ssc` shapes. However, if we are already in some `ssc` shape and a shape change is requested we know that at least some `ssc` shapes exist for the current font family, so rather than falling back to some `sc` shape we try to stay within `ssc` shapes in a fallback situation.

```

2956 \DeclareFontShapeChangeRule {n}{ssc}    {ssc}    {sc}

```

```

2957 % \DeclareFontShapeChangeRule {it}{n} {n} {}
2958 % \DeclareFontShapeChangeRule {it}{it} {it} {}
2959 \DeclareFontShapeChangeRule {it}{sl} {sl} {it}
2960 \DeclareFontShapeChangeRule {it}{sw} {sw} {it}

```

If neither `scit` nor `scsl` exist then `sc` will be used as a fallback albeit with a log entry, so except for the latter there will be no change for CM or Latin Modern fonts.

```

2961 \DeclareFontShapeChangeRule {it}{sc} {scit} {scsl}
2962 \DeclareFontShapeChangeRule {it}{ulc} {it} {}
2963 \DeclareFontShapeChangeRule {it}{up} {n} {}
2964 \DeclareFontShapeChangeRule {it}{ssc} {sscit} {scit}

2965 % \DeclareFontShapeChangeRule {sl}{n} {n} {}
2966 \DeclareFontShapeChangeRule {sl}{it} {it} {sl}
2967 % \DeclareFontShapeChangeRule {sl}{sl} {sl} {}
2968 \DeclareFontShapeChangeRule {sl}{sw} {sw} {it}
2969 \DeclareFontShapeChangeRule {sl}{sc} {scsl} {scit}
2970 \DeclareFontShapeChangeRule {sl}{ulc} {sl} {}
2971 \DeclareFontShapeChangeRule {sl}{up} {n} {}
2972 \DeclareFontShapeChangeRule {sl}{ssc} {sscsl} {scsl}

2973 % \DeclareFontShapeChangeRule {sc}{n} {n} {}
2974 \DeclareFontShapeChangeRule {sc}{it} {scit} {scsl}
2975 \DeclareFontShapeChangeRule {sc}{sl} {scsl} {scit}
2976 \DeclareFontShapeChangeRule {sc}{sw} {scsw} {scit}
2977 % \DeclareFontShapeChangeRule {sc}{sc} {sc} {}
2978 \DeclareFontShapeChangeRule {sc}{ulc} {n} {}

```

The next rule might be a bit surprising, and rightly so. It would be more correct if `sc` were not affected by `up`, so that it remains `sc` as shown in the commented out rule. However, for nearly three decades commands such as `\upshape` or `\textup` changed small caps back to the “normal” shape. So for backward compatibility we keep that behavior.

As a result you are currently typesetting in `scit` or `scsl` using `\upshape` twice will return you to the normal shape too, the first will change to `sc` and the second (because of the rule below) change that to `n`. This is the way `fontspec` implemented its version on this interface, so this rule means we are also compatible with the way `fontspec` behaved. Still it remains an oddity which I would rather liked to have avoided.

```

2979 % \DeclareFontShapeChangeRule {sc}{up} {sc} {}
2980 \DeclareFontShapeChangeRule {sc}{up} {n} {}
2981 % \DeclareFontShapeChangeRule {sc}{ssc} {ssc} {}

2982 % \DeclareFontShapeChangeRule {scit}{n} {n} {}
2983 \DeclareFontShapeChangeRule {scit}{it} {scit} {}
2984 \DeclareFontShapeChangeRule {scit}{sl} {scsl} {scit}
2985 \DeclareFontShapeChangeRule {scit}{sw} {scsw} {scit}
2986 \DeclareFontShapeChangeRule {scit}{sc} {scit} {}
2987 \DeclareFontShapeChangeRule {scit}{ulc} {it} {}

```

The next rule assumes that if `scit` exists then `it` exists as well. If not, the mechanism will save `ulc` in `\f@series`, which most certainly doesn’t exist. So when a font is later selected that would result in a substitution (so no harm done really). Alternatively, we could in this case use `n` as alternative, which may be a bit faster, but such a setup would be so weird in the first place that this isn’t worth the effort.

```

2988 \DeclareFontShapeChangeRule {scit}{up} {sc} {}
2989 \DeclareFontShapeChangeRule {scit}{ssc} {sscit} {scit}

```

```

2990 % \DeclareFontShapeChangeRule {scsl}{n} {n} {}
2991 \DeclareFontShapeChangeRule {scsl}{it} {scit} {scsl}
2992 \DeclareFontShapeChangeRule {scsl}{sl} {scsl} {}
2993 \DeclareFontShapeChangeRule {scsl}{sw} {scsw} {scsl}
2994 \DeclareFontShapeChangeRule {scsl}{sc} {scsl} {}
2995 \DeclareFontShapeChangeRule {scsl}{ulc} {sl} {}
2996 \DeclareFontShapeChangeRule {scsl}{up} {sc} {}
2997 \DeclareFontShapeChangeRule {scsl}{ssc} {sscsl} {scsl}

2998 % \DeclareFontShapeChangeRule {scsw}{n} {n} {}
2999 \DeclareFontShapeChangeRule {scsw}{it} {scit} {scsw}
3000 \DeclareFontShapeChangeRule {scsw}{sl} {scsl} {}
3001 \DeclareFontShapeChangeRule {scsw}{sw} {scsw} {}
3002 \DeclareFontShapeChangeRule {scsw}{sc} {scsw} {}
3003 \DeclareFontShapeChangeRule {scsw}{ulc} {sw} {}
3004 \DeclareFontShapeChangeRule {scsw}{up} {sc} {}
3005 \DeclareFontShapeChangeRule {scsw}{ssc} {sscsw} {scsw}

3006 % \DeclareFontShapeChangeRule {sw}{n} {n} {}
3007 % \DeclareFontShapeChangeRule {sw}{it} {it} {}
3008 % \DeclareFontShapeChangeRule {sw}{sl} {sl} {}
3009 % \DeclareFontShapeChangeRule {sw}{sw} {sw} {}
3010 \DeclareFontShapeChangeRule {sw}{sc} {scsw} {scit}
3011 \DeclareFontShapeChangeRule {sw}{ulc} {sw} {}
3012 \DeclareFontShapeChangeRule {sw}{up} {n} {}
3013 \DeclareFontShapeChangeRule {sw}{ssc} {sscsw} {scsw}

3014 % \DeclareFontShapeChangeRule {ssc}{n} {n} {}
3015 \DeclareFontShapeChangeRule {ssc}{it} {sscit} {sscsl}
3016 \DeclareFontShapeChangeRule {ssc}{sl} {sscsl} {sscit}
3017 \DeclareFontShapeChangeRule {ssc}{sw} {sscsw} {sscit}
3018 \DeclareFontShapeChangeRule {ssc}{sc} {sc} {}
3019 % \DeclareFontShapeChangeRule {ssc}{ssc} {ssc} {}
3020 \DeclareFontShapeChangeRule {ssc}{ulc} {n} {}

```

We implement the same logic as for `sc`, see above. The `ssc` shape doesn't have to care about 30 years of history, but it would be surprising if `\sscshape\upshape` did not work like `\scshape\upshape`.

```

3021 % \DeclareFontShapeChangeRule {ssc}{up} {ssc} {}
3022 \DeclareFontShapeChangeRule {ssc}{up} {n} {}

3023 % \DeclareFontShapeChangeRule {sscit}{n} {n} {}
3024 \DeclareFontShapeChangeRule {sscit}{it} {sscit} {}
3025 \DeclareFontShapeChangeRule {sscit}{sl} {sscsl} {sscit}
3026 \DeclareFontShapeChangeRule {sscit}{sw} {sscsw} {sscit}
3027 \DeclareFontShapeChangeRule {sscit}{ssc} {sscit} {}
3028 \DeclareFontShapeChangeRule {sscit}{sc} {scit} {}
3029 \DeclareFontShapeChangeRule {sscit}{ulc} {it} {}
3030 \DeclareFontShapeChangeRule {sscit}{up} {ssc} {}

3031 % \DeclareFontShapeChangeRule {sscsl}{n} {n} {}
3032 \DeclareFontShapeChangeRule {sscsl}{it} {sscit} {sscsl}
3033 \DeclareFontShapeChangeRule {sscsl}{sl} {sscsl} {}
3034 \DeclareFontShapeChangeRule {sscsl}{sw} {sscsw} {sscit}
3035 \DeclareFontShapeChangeRule {sscsl}{sc} {scsl} {}
3036 \DeclareFontShapeChangeRule {sscsl}{ulc} {sl} {}
3037 \DeclareFontShapeChangeRule {sscsl}{up} {ssc} {}

```

```

3038 % \DeclareFontShapeChangeRule {sscsw}{n} {n} {}
3039 \DeclareFontShapeChangeRule {sscsw}{it} {sscit} {sscs1}
3040 \DeclareFontShapeChangeRule {sscsw}{sl} {sscs1} {sscit}
3041 \DeclareFontShapeChangeRule {sscsw}{sw} {sscsw} {}
3042 \DeclareFontShapeChangeRule {sscsw}{ssc} {sscsw} {}
3043 \DeclareFontShapeChangeRule {sscsw}{sc} {scsw} {scit}
3044 \DeclareFontShapeChangeRule {sscsw}{ulc} {sw} {it}
3045 \DeclareFontShapeChangeRule {sscsw}{up} {ssc} {}
3046  $\langle$ latexrelease $\rangle$  /2ekernel | latexrelease)
3047  $\langle$ latexrelease $\rangle$  \EndIncludeInRelease

3048  $\langle$ latexrelease $\rangle$  \IncludeInRelease{2020/02/02}%
3049  $\langle$ latexrelease $\rangle$  {\DeclareFontShapeChangeRule}{Rules for ssc and sw}%
3050  $\langle$ latexrelease $\rangle$ 
3051  $\langle$ latexrelease $\rangle$  \DeclareFontShapeChangeRule {n}{it} {it} {sl}
3052  $\langle$ latexrelease $\rangle$  \DeclareFontShapeChangeRule {n}{sl} {sl} {it}
3053  $\langle$ latexrelease $\rangle$  \DeclareFontShapeChangeRule {n}{ulc} {n} {}
3054  $\langle$ latexrelease $\rangle$  \DeclareFontShapeChangeRule {n}{up} {n} {}
3055  $\langle$ latexrelease $\rangle$  \DeclareFontShapeChangeRule {it}{sl} {sl} {it}
3056  $\langle$ latexrelease $\rangle$  \DeclareFontShapeChangeRule {it}{sc} {scit} {scsl}
3057  $\langle$ latexrelease $\rangle$  \DeclareFontShapeChangeRule {it}{ulc} {it} {}
3058  $\langle$ latexrelease $\rangle$  \DeclareFontShapeChangeRule {it}{up} {n} {}
3059  $\langle$ latexrelease $\rangle$  \DeclareFontShapeChangeRule {sl}{it} {it} {sl}
3060  $\langle$ latexrelease $\rangle$  \DeclareFontShapeChangeRule {sl}{sc} {scsl} {scit}
3061  $\langle$ latexrelease $\rangle$  \DeclareFontShapeChangeRule {sl}{ulc} {sl} {}
3062  $\langle$ latexrelease $\rangle$  \DeclareFontShapeChangeRule {sl}{up} {n} {}
3063  $\langle$ latexrelease $\rangle$  \DeclareFontShapeChangeRule {sc}{it} {scit} {scsl}
3064  $\langle$ latexrelease $\rangle$  \DeclareFontShapeChangeRule {sc}{sl} {scsl} {scit}
3065  $\langle$ latexrelease $\rangle$  \DeclareFontShapeChangeRule {sc}{sw} {scsw} {sw}
3066  $\langle$ latexrelease $\rangle$  \DeclareFontShapeChangeRule {sc}{ulc} {n} {}
3067  $\langle$ latexrelease $\rangle$  \DeclareFontShapeChangeRule {sc}{up} {n} {}
3068  $\langle$ latexrelease $\rangle$  \DeclareFontShapeChangeRule {scit}{it} {scit} {}
3069  $\langle$ latexrelease $\rangle$  \DeclareFontShapeChangeRule {scit}{sl} {scsl} {scit}
3070  $\langle$ latexrelease $\rangle$  \DeclareFontShapeChangeRule {scit}{sw} {scsw} {sc} % or scit?
3071  $\langle$ latexrelease $\rangle$  \DeclareFontShapeChangeRule {scit}{sc} {scit} {}
3072  $\langle$ latexrelease $\rangle$  \DeclareFontShapeChangeRule {scit}{ulc} {it} {}
3073  $\langle$ latexrelease $\rangle$  \DeclareFontShapeChangeRule {scit}{up} {sc} {}
3074  $\langle$ latexrelease $\rangle$  \DeclareFontShapeChangeRule {scsl}{it} {scit} {scsl}
3075  $\langle$ latexrelease $\rangle$  \DeclareFontShapeChangeRule {scsl}{sl} {scsl} {}
3076  $\langle$ latexrelease $\rangle$  \DeclareFontShapeChangeRule {scsl}{sw} {scsw} {sc} % or scsl?
3077  $\langle$ latexrelease $\rangle$  \DeclareFontShapeChangeRule {scsl}{sc} {scsl} {}
3078  $\langle$ latexrelease $\rangle$  \DeclareFontShapeChangeRule {scsl}{ulc} {sl} {}
3079  $\langle$ latexrelease $\rangle$  \DeclareFontShapeChangeRule {scsl}{up} {sc} {}
3080  $\langle$ latexrelease $\rangle$  \DeclareFontShapeChangeRule {scsw}{it} {scit} {scsw}
3081  $\langle$ latexrelease $\rangle$  \DeclareFontShapeChangeRule {scsw}{sl} {scsl} {}
3082  $\langle$ latexrelease $\rangle$  \DeclareFontShapeChangeRule {scsw}{sw} {scsw} {}
3083  $\langle$ latexrelease $\rangle$  \DeclareFontShapeChangeRule {scsw}{sc} {scsw} {}
3084  $\langle$ latexrelease $\rangle$  \DeclareFontShapeChangeRule {scsw}{ulc} {sw} {}
3085  $\langle$ latexrelease $\rangle$  \DeclareFontShapeChangeRule {scsw}{up} {sc} {}
3086  $\langle$ latexrelease $\rangle$  \DeclareFontShapeChangeRule {sw}{sc} {scsw} {}
3087  $\langle$ latexrelease $\rangle$  \DeclareFontShapeChangeRule {sw}{ulc} {sw} {}
3088  $\langle$ latexrelease $\rangle$  \DeclareFontShapeChangeRule {sw}{up} {n} {}
3089  $\langle$ latexrelease $\rangle$ 

```

In 2022-02-02 the ssc shape had no rules, so that any use of it would have resulted in

just switching to that shape or switching away from it; so we mimic that by providing the corresponding trivial rules to overrule those from above when we roll back.

```

3090 <latexrelease>\DeclareFontShapeChangeRule {n}{ssc} {ssc} {}
3091 <latexrelease>\DeclareFontShapeChangeRule {it}{ssc} {ssc} {}
3092 <latexrelease>\DeclareFontShapeChangeRule {sl}{ssc} {ssc} {}
3093 <latexrelease>\DeclareFontShapeChangeRule {sw}{ssc} {ssc} {}
3094 <latexrelease>
3095 <latexrelease>\DeclareFontShapeChangeRule {scit}{ssc}{ssc} {}
3096 <latexrelease>\DeclareFontShapeChangeRule {scsl}{ssc}{ssc} {}
3097 <latexrelease>\DeclareFontShapeChangeRule {scsw}{ssc}{ssc} {}
3098 <latexrelease>
3099 <latexrelease>\DeclareFontShapeChangeRule {ssc}{it} {it} {}
3100 <latexrelease>\DeclareFontShapeChangeRule {ssc}{sl} {sl} {}
3101 <latexrelease>\DeclareFontShapeChangeRule {ssc}{sw} {sw} {}
3102 <latexrelease>\DeclareFontShapeChangeRule {ssc}{sc} {sc} {}
3103 <latexrelease>\DeclareFontShapeChangeRule {ssc}{ulc} {n} {}
3104 <latexrelease>\DeclareFontShapeChangeRule {ssc}{up} {n} {}
3105 <latexrelease>
3106 <latexrelease>\DeclareFontShapeChangeRule {sscit}{it} {it} {}
3107 <latexrelease>\DeclareFontShapeChangeRule {sscit}{sl} {sl} {}
3108 <latexrelease>\DeclareFontShapeChangeRule {sscit}{sw} {sw} {}
3109 <latexrelease>\DeclareFontShapeChangeRule {sscit}{ssc} {ssc} {}
3110 <latexrelease>\DeclareFontShapeChangeRule {sscit}{sc} {sc} {}
3111 <latexrelease>\DeclareFontShapeChangeRule {sscit}{ulc} {n} {}
3112 <latexrelease>\DeclareFontShapeChangeRule {sscit}{up} {ssc} {}
3113 <latexrelease>
3114 <latexrelease>\DeclareFontShapeChangeRule {sscsl}{it} {it} {}
3115 <latexrelease>\DeclareFontShapeChangeRule {sscsl}{sl} {sl} {}
3116 <latexrelease>\DeclareFontShapeChangeRule {sscsl}{sw} {sw} {}
3117 <latexrelease>\DeclareFontShapeChangeRule {sscsl}{sc} {sc} {}
3118 <latexrelease>\DeclareFontShapeChangeRule {sscsl}{ulc} {n} {}
3119 <latexrelease>\DeclareFontShapeChangeRule {sscsl}{up} {ssc} {}
3120 <latexrelease>
3121 <latexrelease>\DeclareFontShapeChangeRule {sscsw}{it} {it} {}
3122 <latexrelease>\DeclareFontShapeChangeRule {sscsw}{sl} {sl} {}
3123 <latexrelease>\DeclareFontShapeChangeRule {sscsw}{sw} {sw} {}
3124 <latexrelease>\DeclareFontShapeChangeRule {sscsw}{ssc} {ssc} {}
3125 <latexrelease>\DeclareFontShapeChangeRule {sscsw}{sc} {sc} {}
3126 <latexrelease>\DeclareFontShapeChangeRule {sscsw}{ulc} {n} {}
3127 <latexrelease>\DeclareFontShapeChangeRule {sscsw}{up} {ssc} {}
3128 <latexrelease>\EndIncludeInRelease

```

Before 2020-02-02 we don't really care about any existing shape change rules since the mechanism isn't used.

```

3129 <latexrelease>\IncludeInRelease{0000/00/00}%
3130 <latexrelease>    {\DeclareFontShapeChangeRule}{Rules for ssc and sw}%
3131 <latexrelease>\EndIncludeInRelease

```

2.2 Changing to a new shape

```

3132 <*2ekernel | latexrelease>
3133 <latexrelease>\IncludeInRelease{2021/06/01}%
3134 <latexrelease>    {\fontshape}{Font shape change}%

```

`\fontshape` Again the `\fontshape` now has to do a lookup to get to its new value in `\f@shape`. The method is exactly the same as in `\fontseries`.

```
3135 \DeclareRobustCommand\fontshape[1]
3136     {\expandafter\def\expandafter\delayed@f@adjustment\expandafter
3137         {\delayed@f@adjustment\delayed@merge@font@shape{#1}}}
```

(End of definition for \fontshape.)

`\fontshapeforce` The unconditional version:

```
3138 \DeclareRobustCommand\fontshapeforce[1]
3139     {\expandafter\def\expandafter\delayed@f@adjustment\expandafter
3140         {\delayed@f@adjustment\edef\f@shape{#1}}}
```

(End of definition for \fontshapeforce.)

Supporting rollback ...

```
3141 </2ekernel | latexrelease>
3142 <latexrelease>\EndIncludeInRelease
3143 <latexrelease>\IncludeInRelease{2020/02/02}%
3144 <latexrelease>    {\fontshape}{Font shape change}%
3145 <latexrelease>
3146 <latexrelease>\DeclareRobustCommand\fontshape[1]{\merge@font@shape{#1}}
3147 <latexrelease>\DeclareRobustCommand\fontshapeforce[1]{\edef\f@shape{#1}}
3148 <latexrelease>
3149 <latexrelease>\EndIncludeInRelease
3150 <latexrelease>\IncludeInRelease{0000/00/00}%
3151 <latexrelease>    {\fontshape}{Font shape change}%
3152 <latexrelease>
3153 <latexrelease>\DeclareRobustCommand\fontshape [1]{\edef\f@shape{#1}}
3154 <latexrelease>\let\fontshapeforce@undefined
3155 <latexrelease>
3156 <latexrelease>\EndIncludeInRelease
3157 <*2ekernel | latexrelease>
3158 <latexrelease>\IncludeInRelease{2020/02/02}%
3159 <latexrelease>    {\merge@font@shape}{Font shape change rules}%
```

`\merge@font@shape` Look up the database entry (if existing) and act accordingly.

```
3160 \def\merge@font@shape#1{%
3161     \expandafter\expandafter\expandafter
3162     \merge@font@shape@
3163     \csname shape@\f@shape @#1\endcsname
3164     {#1}%
3165     \@nil
3166 }
```

(End of definition for \merge@font@shape.)

`\merge@font@shape@` Same game now, except that we look at shapes not series values and we can set the shape without the complication of dropping “m”s from the name as we had to for the series.

```
3167 \def\merge@font@shape@#1#2#3\@nil{%
3168     \def\reserved@a{#3}%
3169     \ifx\reserved@a@empty
3170         \edef\f@shape{#2}%
3171     \else
```

`\reserved@a` is used in `\@font@shape@subst@warning` so we have to define it in addition to do the `\ifcsname` test

```

3172 \edef\reserved@a{\f@encoding /\f@family /\f@series/#1}%
3173 \ifcsname \reserved@a\endcsname
3174 \edef\f@shape{#1}%
3175 \else
3176 \ifcsname \f@encoding /\f@family /\f@series/#2\endcsname
3177 \edef\f@shape{#2}%
3178 \@font@shape@subst@warning
3179 \else
3180 \edef\f@shape{#3}%
3181 \@font@shape@subst@warning
3182 \fi
3183 \fi
3184 \fi
3185 }

```

(End of definition for `\merge@font@shape@`.)

See definition of `\selectfont` for how these macros are used.

`\merge@font@shape@without@substitution`
`\merge@font@shape@without@substitution@`
`\delayed@merge@font@shape`

```

3186 \def\merge@font@shape@without@substitution#1{%
3187 \expandafter\expandafter\expandafter
3188 \merge@font@shape@without@substitution@
3189 \csname shape@\f@shape @#1\endcsname
3190 {#1}%
3191 \@nil
3192 }
3193 \def\merge@font@shape@without@substitution@#1#2#3\@nil{%
3194 \def\reserved@a{#3}%
3195 \ifx\reserved@a@empty
3196 \edef\f@shape{#2}%
3197 \else
3198 \edef\f@shape{#1}%
3199 \fi
3200 }
3201 \let\delayed@merge@font@shape\merge@font@shape@without@substitution

```

*(End of definition for `\merge@font@shape@without@substitution`,
`\merge@font@shape@without@substitution@`, and `\delayed@merge@font@shape`.)*

`\normalshape` `\normalshape` resets both sub-axes if the default rules are used.

```

3202 \protected\def\normalshape
3203 {\not@math@alphabet\normalshape\relax
3204 \fontshape\shapedefault\selectfont}%

```

(End of definition for `\normalshape`.)

3 Make sure we win . . .

This code implements one aspect of what the package `fontaxes` provide. So its redefinitions for the various shape commands, such as `\itshape` should no longer happen. We therefore force the standard definitions at `\AtBeginDocument` (later when this is defined). Once `fontaxes` is no longer doing such redefinitions that could be taken out again.

We use a separate macro so that we can easily disable this (in case of rollback).

`\reinstall@nfss@defs` I use `\protected` here not `\DeclareRobustCommand` to avoid extra status lines.

```
3205 \def\reinstall@nfss@defs{%
3206   \protected\def\upshape
3207     {\not@math@alphabet\upshape\relax
3208     \fontshape\updefault\selectfont}%
3209   \protected\def\slshape
3210     {\not@math@alphabet\slshape\relax
3211     \fontshape\sldefault\selectfont}%
3212   \protected\def\scshape
3213     {\not@math@alphabet\scshape\relax
3214     \fontshape\scdefault\selectfont}%
3215   \protected\def\itshape
3216     {\not@math@alphabet\itshape\mathit
3217     \fontshape\itdefault\selectfont}%
3218   \protected\def\ulcshape
3219     {\not@math@alphabet\ulcshape\relax
3220     \fontshape\ulc\selectfont}%
3221   \protected\def\swshape
3222     {\not@math@alphabet\swshape\relax
3223     \fontshape\swdefault\selectfont}%
3224   \protected\def\sscshape
3225     {\not@math@alphabet\sscshape\relax
3226     \fontshape\sscdefault\selectfont}%
3227 }
```

(End of definition for `\reinstall@nfss@defs`.)

Supporting rollback ...

```
3228 </2ekernel | latexrelease>
3229 <latexrelease>\EndIncludeInRelease
3230 <latexrelease>\IncludeInRelease{0000/00/00}%
3231 <latexrelease>  {\merge@font@shape}{Font shape change rules}%
3232 <latexrelease>
3233 <latexrelease>\DeclareRobustCommand\fontshape [1]{\edef\f@shape{#1}}
3234 <latexrelease>\let\fontshapeforce\@undefined
3235 <latexrelease>
3236 <latexrelease>\let\merge@font@shape\@undefined
3237 <latexrelease>\let\merge@font@shape@\@undefined
3238 <latexrelease>
3239 <latexrelease>\let\merge@font@shape@without@substitution\@undefined
3240 <latexrelease>\let\merge@font@shape@without@substitution@\@undefined
3241 <latexrelease>\let\delayed@merge@font@shape@\@undefined
3242 <latexrelease>
3243 <latexrelease>\let\normalshape\@undefined
3244 <latexrelease>
```

This is always called in `\document` so don't make it undefined.

```
3245 <latexrelease>
3246 <latexrelease>\let\reinstall@nfss@defs\relax
3247 <latexrelease>\EndIncludeInRelease
```

This initializes the 2020/02/02 extensions to NFSS after any changes in the preamble.

```
3248 <*2ekernel | latexrelease>
3249 <latexrelease>\IncludeInRelease{2020/10/01}%
```

```

3250 <latexrelease>                {\reinstall@nfss@defs}{NFSS series init}%
3251 \g@addto@macro\@kernel@after@begindocument@before
3252                {\reinstall@nfss@defs\init@series@setup}
3253 </2ekernel | latexrelease>
3254 <latexrelease>\EndIncludeInRelease
        The initialization was introduced in 2020/02/02 but
3255 <latexrelease>\IncludeInRelease{2020/02/02}%
3256 <latexrelease>                {\reinstall@nfss@defs}{NFSS series init}%
3257 <latexrelease>\AtBeginDocument{\reinstall@nfss@defs\init@series@setup}
3258 <latexrelease>\EndIncludeInRelease
3259 <latexrelease>\IncludeInRelease{0000/00/00}%
3260 <latexrelease>                {\reinstall@nfss@defs}{NFSS series init}%
3261 <latexrelease>\EndIncludeInRelease
3262 <*2ekernel>
3263 </2ekernel>

```

File 26

lfsstrc.dtx

1 Introduction

This package contains the code for tracing font loading and font changes. It basically overlays some of the low-level functions of NFSS with additional code used for tracing.

The package accepts the following options:

errorshow Write all information about font changes etc. only to the transcript file unless an error happens. This means that information about font substitution will not be shown on the terminal.

warningshow Show all NFSS warnings on the terminal. This setting corresponds to the default behaviour of NFSS if the `tracefnt` package is *not* loaded!

infoshow Show all NFSS warning and all NFSS info messages (that are normally only written to the transcript file) also on the terminal. This is the default if the `tracefnt` package is loaded.

debugshow In addition to `infoshow` show also changing of math fonts as far as possible (this option can produce a large amount of output).

loading Show the name of external fonts when they are loaded. This option shows only “newly” loaded fonts not those already preloaded in the format or the class file before the `tracefnt` package became active.

pausing Turn all font warnings into errors so that L^AT_EX will stop.

2 A driver for this document

The next bit of code contains the documentation driver file for T_EX, i.e., the file that will produce the documentation you are currently reading. It will be extracted from this file by the DOCSTRIP program.

When this file is processed directly by L^AT_EX this will produce the documentation as well.

```
1 <*driver>
2 \documentclass{ltxdoc}
3
4
5 %\OnlyDescription % comment out for implementation details
6
7 \begin{document}
8   \DocInput{lfsstrc.dtx}
9 \end{document}
10 </driver>
```

3 The Implementation

Warning: Read the macro documentation with a grain of salt. It is still basically the documentation from the first NFSS release and therefore in some cases probably not completely accurate.

If we are making a package file it is a good idea to test whether we are running under 2e. This code is actually placed at the very beginning of this file for easier maintenance, thus commented out here.

```
11 <*package>
12 %\NeedsTeXFormat{LaTeX2e}
13 %\ProvidesPackage{tracefnt}[??/??/?? v?.??]
14 %
15 </package>
```

The `debug` module makes use of commands contained in a special package file named `trace.sty`.³⁴

```
16 <+debug> \input trace.sty
```

4 Handling Options

`\tracingfonts` Here is the definition of the integer register for the font trace. As a default in a package file we use 1 to give error messages if fonts are substituted. If this code is used for debugging or tracing reasons in the format file (i.e. in `fam.dtx`) we use 0 as the default. But if no font trace is used we build a definition that will produce a warning message.

```
17 <2kernel>
18 \message{NFSS tracing,}
19 \def\tracingfonts{%
20   \@font@warning{Command \noexpand\tracingfonts
21     not provided.\MessageBreak
22     Use the ‘tracefnt’ package.\MessageBreak Command found:}%
23   \count@}
24 </2kernel>
```

The `\count@` in the line above will remove the number after `\tracingfonts`. Note that this definition will be overwritten by the next line if one of these modules are included.

```
25 <*package, trace, debug>
26 \newcount\tracingfonts
27 \tracingfonts=0
28 </package, trace, debug>
```

(End of definition for \tracingfonts.)

The option `errorshow` turns off all warnings so that only real errors are shown. `warningshow` corresponds to the NFSS default (when `tracefnt` is not loaded). `infoshow` is the default for this package here; and `debugshow`, `loading`, and `pausing` extend the amount of information even further.

```
29 <*package>
30 \DeclareOption{errorshow}{%
31   \def\@font@info#1{%
32     \GenericInfo{(Font)\@spaces\@spaces\@spaces\space\space}}%
```

³⁴This package is not in distribution at the moment (and probably doesn't any longer work). Think of this part of the code as being historical artifacts.

```

33             {LaTeX Font Info: \space\space\space#1}}%
34 \def\@font@warning#1{%
35     \GenericInfo{(Font)\@spaces\@spaces\@spaces\space\space}%
36         {LaTeX Font Warning: #1}}%
37 }
38 \DeclareOption{warningshow}{%
39 \def\@font@info#1{%
40     \GenericInfo{(Font)\@spaces\@spaces\@spaces\space\space}%
41         {LaTeX Font Info: \space\space\space#1}}%
42 \def\@font@warning#1{%
43     \GenericWarning{(Font)\@spaces\@spaces\@spaces\space\space}%
44         {LaTeX Font Warning: #1}}%
45 }
46 \DeclareOption{infoshow}{%
47 \def\@font@info#1{%
48     \GenericWarning{(Font)\@spaces\@spaces\@spaces\space\space}%
49         {LaTeX Font Info: \space\space\space#1}}%
50 \def\@font@warning#1{%
51     \GenericWarning{(Font)\@spaces\@spaces\@spaces\space\space}%
52         {LaTeX Font Warning: #1}}%
53 }
54 \DeclareOption{loading}{%
55     \tracingfonts\tw@
56 }
57 \DeclareOption{debugshow}{%
58     \ExecuteOptions{infoshow}%
59     \tracingfonts\thr@@
60 }
61 \DeclareOption{pausing}{%
62 \def\@font@warning#1{%
63     \GenericError
64         {(Font)\@spaces\@spaces\@spaces\space\space}%
65         {LaTeX Font Warning: #1}%
66         {See the LaTeX Companion for details.}%
67         {I'll stop for every LaTeX Font Warning because
68         you requested\MessageBreak the 'pausing' option
69         to the tracefnt package.}}%
70 }

```

We make `infoshow` the default, which in turn defines `\font@warning` and `\font@info`.

```

71 \ExecuteOptions{infoshow}
72 \ProcessOptions
73 \</package>

```

We also need a default definition inside the kernel:

```

74 \<2kernel>
75 \def\@font@info#1{%
76     \GenericInfo{(Font)\@spaces\@spaces\@spaces\space\space}%
77         {LaTeX Font Info: \space\space\space#1}}%
78 \def\@font@warning#1{%
79     \GenericWarning{(Font)\@spaces\@spaces\@spaces\space\space}%
80         {LaTeX Font Warning: #1}}%
81 \</2kernel>

```

5 Macros common to `fam.tex` and `tracefmt.sty`

In the first versions of `tracefmt.dtx` some macros of `fam.dtx`³⁵ were redefined to include the extra tracing information. Now these macros are all defined in this file (i.e. removed from `fam.dtx`) and different production versions can be obtained simply by specifying a different set of modules to include when generating `lftss.dtx`.

5.1 General font loading

`\extract@font` This macro organizes the font loading. It first calls `\get@external@font` which will return in `\external@font` the name of the external font file (the `.tfm`) as it was determined by the NFSS tables.

```
82 <*2kernel | package>
83 \def\extract@font{%
84   \get@external@font
```

Then the external font is loaded and assigned to the font identifier stored inside `\font@name` (for this reason we need `\expandafter`).

```
85   \global\expandafter\font\font@name\external@font\relax
```

When tracing we typeout the internal and external font name.

```
86 <*trace>
87   \ifnum \tracingfonts >\@ne
88   \@font@info{External font '\external@font'
89     loaded as\MessageBreak \font@name}\fi
90 </trace>
```

Finally we call the corresponding “loading action” macros to finish things. First the font is locally selected to allow the use of `\font` inside the loading action macros.

```
91   \font@name \relax
```

The next two lines execute the “loading actions” for the family and then for the individual font shape.

```
92   \csname \f@encoding+\f@family\endcsname
93   \csname\curr@fontshape\endcsname
94   \relax
95   }
96 </2kernel | package>
```

The `\relax` at the end needs to be explained. This is inserted to prevent \TeX from scanning too far when it is executing the replacement text of the loading code macros.

(End of definition for `\extract@font`.)

`\get@external@font` This function tries to find an external font name. It will place the name into the macro `\external@font`. If no font is found it will return the one that was defined via `\DeclareErrorFont`.

```
97 <*2kernel>
98 \def\get@external@font{%
```

We don't know the external font name at the beginning.

```
99   \let\external@font\@empty
100   \edef\font@info{\expandafter\expandafter\expandafter\string
101     \csname \curr@fontshape \endcsname}%
102   \try@size@range
```

³⁵This file is currently not distributed in documented form. Its code is part of `lftss.dtx`.

If this failed, we'll try to substitute another size of the same font. This is done by the `\try@size@substitution` macro. It “knows about” `\do@extract@font`, `\font@name`, `\f@size`, and so on.

```

103 \ifx\external@font\@empty
104 \try@size@substitution
105 \ifx\external@font\@empty
106 \latex@error{Font \expandafter \string\font@name\space
107 not found}\@eha
108 \error@fontshape
109 \get@external@font
110 \fi\fi
111 }
112 </2ekernel>

```

(End of definition for `\get@external@font`.)

```

113 <*2ekernel | latexrelease | package>
114 <latexrelease> \IncludeInRelease{2021/06/01}%
115 <latexrelease> \selectfont}{Add hook to \selectfont}%

```

`\selectfont` The macro `\selectfont` is called whenever a font change must take place.

```

116 \DeclareRobustCommand\selectfont
117 {%

```

When `debug` is specified we actually want something like ‘`undebug`’. The font selection is now stable so that using `\tracingall` on some other macros will show us a lot of unwanted information about font loading. Therefore we disable tracing during font loading as long as `\tracingfonts` is less than 4.

```

118 <+debug> \pushtracing
119 <+debug> \ifnum\tracingfonts<4 \tracingoff
120 <+debug> \else \tracingon\p@selectfont \fi

```

If `\baselinestretch` was redefined by the user it will not longer match its internal counterpart `\f@linespread`. If so we call `\set@fontsize` to prepare `\size@update`.

```

121 \ifx\f@linespread\baselinestretch \else
122 \set@fontsize\baselinestretch\f@size\f@baselineskip \fi

```

The series and shape updates are only prepared by `\fontseries` and `\fontshape` but not executed until after we are ready to change the font face. This way they happen after a possibly new family is set which is important because they look at the available font faces in that family and alter the selection based on availability. Several calls to `\fontseries` or `\fontshape` are delayed in the order in which they appear, so that by switching them one can work around missing intermediate font faces and avoid substitutions.

We first attempt to do the merge without any substitution. As we might end up with a non-existing font face we may have to restart and therefore save the current values of `\f@series` and `\f@shape` before the merge.

But first we make a quick test to see if there are any delayed actions, because if not it is pointless to make all the assignments and try loading a missing fontshape.

```

123 \ifx\delayed@f@adjustment\@empty
124 \else
125 \let\f@shape@savd\f@shape
126 \let\f@series@savd\f@series

```

Then we run the delayed adjustments (which use the `\..@without@substitution` commands):

```
127     \delayed@f@adjustment
```

We then check if the resulting combination is valid but for this we have to make sure that the appropriate `.fd` is loaded if that hasn't happened so far.

```
128     \maybe@load@fontshape
129     \ifcsname \f@encoding/\f@family/\f@series/\f@shape \endcsname
```

If this macro is defined then we are good and no further action is necessary.

Otherwise the combination is not valid, so we redo the merge but this time with substitutions.

```
130     \else
131     \let\f@shape\f@shape@saved
132     \let\f@series\f@series@saved
133     \let\delayed@merge@font@shape\merge@font@shape
134     \let\delayed@merge@font@series\merge@font@series
135     \delayed@f@adjustment
136     \let\delayed@merge@font@shape\merge@font@shape@without@substitution
137     \let\delayed@merge@font@series\merge@font@series@without@substitution
138     \fi
```

Now the series and shape values are updated and we clear `\delayed@f@adjustment`. This is important because on the next execution of `\selectfont` we should not mistakenly redo the delayed actions if there wasn't any series or shape change.

```
139     \let\delayed@f@adjustment\@empty
140     \fi
```

If the series was forced we should now cancel that in case the next series change is done with some low-level setting to `\f@series`.

```
141     \@forced@seriesfalse
```

Then we generate the internal name of the font by concatenating *family*, *series*, *shape*, and current *size*, with slashes as delimiters between them. This is much more readable than standard L^AT_EX's `\twfbbf`, etc. We define `\font@name` globally, as always. The reason for this is explained later on.

```
142     \xdef\font@name{%
143     \csname\curr@fontshape/\f@size\endcsname}%
```

We call the macro `\pickup@font` which will load the font if necessary.

```
144     \pickup@font
```

Then we select the font.

```
145     \font@name
```

After switching fonts we run a hook, so that packages can make last minute alterations based on the new font (originally provided in `everyysel` but using a different interface).

```
146     \UseHook{selectfont}%
```

Finally we call `\size@update`. This macro is normally empty but will contain actions (like setting the `\baselineskip`) that have to be carried out when the font size, the base `\baselineskip` or the `\baselinestretch` have changed.

```
147     \size@update
```


A similar function is called to handle anything related to encoding updates. This one is changed from `\relax` by `\fontencoding`.

```
148   \enc@update
```

Just before ending this macro we have to pop the tracing stack if it was pushed before.

```
149 <+debug> \poptracing
150   }
```

(End of definition for `\selectfont`.)

selectfont Declare the hook used in `selectfont` in the kernel, but not inside the `tracefnt` package.

```
151 <-trace>\NewHook{selectfont}
```

(End of definition for `selectfont`.)

If `\tracingfonts` is greater than 2 we also show the font switch inside `\selectfont`. We do this by adding this code to the hook in the `tracefnt` package: macro might redefine `\font@name`.

```
152 <*trace>
153 \AddToHook{selectfont}
154   {\ifnum \tracingfonts>\tw@
155     \font@info{Switching to \font@name}\fi}
156 </trace>
157 </2ekernel | latexrelease | package>
158 <latexrelease>\EndIncludeInRelease
```

With `\selectfont` having different definitions in different kernels we also have to provide them in the `tracefnt` package to support rollback. In packages that works a bit differently and therefore we have to provide an empty block there.

```
159 <package>\IncludeInRelease{2021/06/01}%
160 <package>           {\selectfont}{Add hook to \selectfont}%
161 <package>\EndIncludeInRelease

162 <latexrelease | package>\IncludeInRelease{0000/00/00}%
163 <latexrelease | package>           {\selectfont}{Add hook to \selectfont}%
164 <latexrelease | package>
165 <latexrelease | package>\DeclareRobustCommand\selectfont
166 <latexrelease | package>  {%
167 <latexrelease | package>    \ifx\f@linespread\baselinestretch \else
168 <latexrelease | package>    \set@fontsize\baselinestretch\f@size\f@baselineskip \fi
169 <latexrelease | package>    \xdef\font@name{%
170 <latexrelease | package>      \csname\curr@fontshape/\f@size\endcsname}%
171 <latexrelease | package>    \pickup@font
172 <latexrelease | package>    \font@name
173 <latexrelease | package>    \size@update
174 <latexrelease | package>    \enc@update
175 <latexrelease | package>    }
176 <latexrelease | package>
177 <latexrelease | package>\EndIncludeInRelease
```

\set@fontsize The macro `\set@fontsize` does the actual work. First it assigns new values to `\f@size`, `\f@baselineskip` and `\f@linespread`.

```
178 <*2ekernel | package>
179 \def\set@fontsize#1#2#3{%
180   \@defaultunits\@tempdimb#2pt\relax\@nnil
```

```

181 \edef\f@size{\strip@pt\@tempdimb}%
182 \@defaultunits\@tempskipa#3pt\relax\@nnil
183 \edef\f@baselineskip{\the\@tempskipa}%
184 \edef\f@linespread{#1}%

```

For backward compatibility and for later testing within `\selectfont` the internal value of `\f@linespread` is passed back to `\baselinestretch`.

```
185 \let\baselinestretch\f@linespread
```

Additional processing will happen within `\selectfont`. For this reason the macro `\size@update` (which will be called in `\selectfont`) will be defined to be:

```
186 \def\size@update{%
```

First calculate the new `\baselineskip` and also store it in `normalbaselineskip`

```

187 \baselineskip\f@baselineskip\relax
188 \baselineskip\f@linespread\baselineskip
189 \normalbaselineskip\baselineskip

```

then to set up a new `\strutbox`

```

190 \setbox\strutbox\hbox{%
191 \vrule\@height.7\baselineskip
192 \@depth.3\baselineskip
193 \@width\z@}%

```

We end with a bit of tracing information.

```

194 <*trace>
195 \ifnum \tracingfonts>\tw@
196 \ifx\f@linespread\@empty
197 \let\reserved@a\@empty
198 \else
199 \def\reserved@a{\f@linespread x}%
200 \fi
201 \@font@info{Changing size to \f@size/\reserved@a
202 \f@baselineskip}%
203 \aftergroup\type@restoreinfo \fi
204 </trace>

```

When all this is processed `\size@update` redefines itself to `\relax` so that in later calls of `\selectfont` no extra code will be executed.

```

205 \let\size@update\relax}%
206 }

```

Instead of defining this macro internally we might speed things up by placing the code into a separate macro and use `\let!`

(End of definition for \set@fontsize.)

`\size@update` Normally this macro does nothing; it will be redefined by `\set@fontsize` to initiate an update.

```
207 \let\size@update\relax
```

(End of definition for \size@update.)

`\type@restoreinfo` This macro produces some info when a font size and/or baseline change will get restored.

```

208 <*trace>
209   \def\type@restoreinfo{%
210     \ifx\f@linespread\@empty
211       \let\reserved@a\@empty
212     \else
213       \def\reserved@a{\f@linespread x}%
214     \fi
215     \@font@info{Restoring size to
216               \f@size/\reserved@a\f@baselineskip}}
217 </trace>

```

(End of definition for `\type@restoreinfo`.)

`\glb@settings` The macro `\glb@settings` globally selects all math fonts for the current size if necessary.

`\glb@currsiz` `\def\glb@settings{%`

When `\glb@settings` gains control a size change was requested and all previous font assignments need to be replaced. Therefore the old values of the fonts are no longer needed. For every *math group* the new assignments are appended to `\math@fonts`. But this happens only if the `\math@fonts` switch is set to true. However, we always set up the correct math sizes for script and scriptscript fonts since they may be needed even if we don't set up the whole math machinery.

Here we set the math size, script size and scriptscript size. If the `S@...` macro is not defined we have to first calculate the three sizes.

```

219   \expandafter\ifx\csname S@\f@size\endcsname\relax
220     \calculate@math@sizes
221   \fi

```

The effect of this is that `\calculate@math@sizes` may or may not define the `S@...` macro. In the first case the next time the same size is requested this macro is used, otherwise `\calculate@math@sizes` is called again. This also sets the `\math@fonts` switch. If it is true we must switch the math fonts.

```

222   \csname S@\f@size\endcsname
223   \ifmath@fonts
224 <*trace>
225     \ifnum \tracingfonts>\tw@
226       \@font@info{Setting up math fonts for
227                 \f@size/\f@baselineskip}\fi
228 </trace>

```

Inside a group we execute the macro for the current math *version*. This sets `\math@fonts` to a list of `\textfont...` assignments. `\getanddefine@fonts` (which may be called at this point) needs the `\escapechar` parameter to be set to `-1`.

```

229   \begingroup
230     \escapechar\m@ne
231     \csname mv@\math@version \endcsname

```

Then we set `\globaldefs` to 1 so that all following changes are done globally. The math font assignments recorded in `\math@fonts` are executed and `\glb@currsiz` is set equal to `\f@size`. This signals that the fonts for math in this size are set up.

```

232     \globaldefs\@ne
233     \math@fonts

```

```

234         \let \glb@currsize \f@size
235     \endgroup

```

Finally we execute any code that is supposed to happen whenever the math font setup changes. This register will be executed in local mode which means that everything that is supposed to have any effect should be done globally inside. We can't execute it within `\globaldefs\@ne` as we don't know what ends up inside this register, e.g., it might contain calculations which use some local registers to calculate the final (global) value.

```

236     \the\every@math@size

```

Otherwise we announce that the math fonts are not set up for this size.

```

237 < *trace >
238     \else
239         \ifnum \tracingfonts > \tw@
240             \@font@info{No math setup for
241                 \f@size/\f@baselineskip}\fi
242 < /trace >
243     \fi
244 }
245 < /2ekernel | package >

```

(End of definition for `\glb@settings` and `\glb@currsize`.)

`\baselinestretch` In `\selectfont` we used `\baselinestretch` as a factor when assigning a value to `\baselineskip`. We use 1 as a default (i.e. no stretch).

```

246 < *2ekernel >
247 \def\baselinestretch{1}

```

(End of definition for `\baselinestretch`.)

`\every@math@size` We must still define the hook `\every@math@size` we used in `\glb@settings`. We initialize it to nothing. It is important to remember that everything that goes into this hook should to global updates, local changes will have weird effects.

```

248 \newtoks\every@math@size
249 \every@math@size={ }
250 < /2ekernel >

```

(End of definition for `\every@math@size`.)

5.2 Math fonts setup

5.2.1 Outline of algorithm for math font sizes

\TeX uses the math fonts that are current when the end of a formula is reached. If we don't want to keep font setups local to every formula (which would result in an enormous overhead), we have to be careful not to end up with the wrong setup in case formulas are nested, e.g., we need to be able to handle

$$\$ a=b+c \mbox{\small for all } b \text{ and } c \text{ in } \mathbb{Z} \$$$

Here the inner formulae b and $c \text{ in } \mathbb{Z}$ are typeset in `\small` but we have to return to `\normalsize` before we reach the closing `\$` of the outer formula.

This is handled in the following way:

1. At any point in the document the global variable `\glb@currsize` contains the point size for which the math fonts currently are set up.

2. Whenever we start a formula we compare its value with the local variable `\f@size` that describes the current text font size.
3. If both are the same we assume that we can use the current math font setup without adjustment.
4. If they differ we call `\glb@settings` which changes the math font setup and updates `\glb@currsiz`.
 - (a) If we are recursively inside another formula (`\if@inmath`) we ensure that `\glb@settings` is executed again in the outer formula, so that the old setup is automatically restored.
 - (b) Otherwise, we set the switch `@inmath` locally to `true` so that all nested formulae will be able to detect that they are nested in some outer formula.

The above algorithm has the following features:

- For sizes which are not containing any formula no math setup is done. Compared to the original algorithm of NFSS this results in the following savings:
 - No unnecessary loading of math fonts for sizes that are not used to typeset any math formulae (explicit or implicit ones).
 - No time overhead due to unnecessary changes of the math font setup on entrance and exit of the text font size.
- Math font setup changes for top-level formulae will survive (there is no restoration after the formula) thus any following formula in the same size will be directly typesettable. Compared to original implementation in NFSS2 the new algorithm has the overhead of one test per formula to see if the current math setup is valid (in the original algorithm the setup was always valid, thus no test was necessary).
- In nested formulae the math font setup is restored in the outer formula by a series of `\aftergroup` commands and checks. Compared to the original algorithm this involves additional checks ($2 \times \langle \text{non-math levels} \rangle$ per inner formula).

5.2.2 Code for math font size setting

`\check@mathfonts` In the `\check@mathfonts` macros we implement the steps 2 to 4 except that instead of a switch the macro `\init@restore@glb@settings` is used.

```

251 <*2kernel | package>
252 \def\check@mathfonts{%
253   \ifx \glb@currsiz \f@size
254   <*trace>
255     \ifnum \tracingfonts>\thr@@
256       \@font@info{*** MATH: no change \f@size\space
257         curr/global (\curr@math@size/\glb@currsiz)}\fi
258   </trace>
259   \else
260   <*trace>
261     \ifnum \tracingfonts>\thr@@
262       \@font@info{*** MATH: setting up \f@size\space
263         curr/global (\curr@math@size/\glb@currsiz)}\fi
264   </trace>

```

```

265     \glb@settings
266     \init@restore@glb@settings
267   \fi
268   \let\curr@math@size\f@size
269   \def\init@restore@glb@settings{\aftergroup\restglb@settings}%
270 }

```

(End of definition for `\check@mathfonts`.)

`\init@restore@glb@settings` This macro does by default nothing but get redefined inside `\check@mathfonts` to initiate fontsize restoring in nested formulas.

```

271 <-trace>\let\init@restore@glb@settings\relax
272 <*trace>
273 \def\init@restore@glb@settings{%
274     \ifnum \tracingfonts>\thr@@
275     \font@info{*** MATH: no resetting (not in
276         nested math)}\fi
277 }
278 </trace>

```

(End of definition for `\init@restore@glb@settings`.)

`\restglb@settings` This macro will be executed the first time after the current formula.

```

279 \def\restglb@settings{%
280 <*trace>
281     \ifnum \tracingfonts>\thr@@
282     \font@info{*** MATH: restoring}\fi
283 </trace>
284     \begingroup
285     \let\f@size\curr@math@size
286     \ifx\glb@currsiz \f@size
287 <*trace>
288     \ifnum \tracingfonts>\thr@@
289     \font@info{*** MATH: ... already okay (\f@size)}\fi
290 </trace>
291     \else
292 <*trace>
293     \ifnum \tracingfonts>\thr@@
294     \font@info{*** MATH: ... to \f@size}\fi
295 </trace>
296     \glb@settings
297     \fi
298     \endgroup
299 }

```

(End of definition for `\restglb@settings`.)

5.2.3 Other code for math

`\use@mathgroup` The `\use@mathgroup` macro should be used in user macros to select a math group. Depending on whether or not the `margid` option is in force it has two or three arguments. For this reason it should be called as the last macro.

First we test if we are inside math mode since we don't want to apply a useless definition.

```

300 \def\use@mathgroup#1#2{\relax\ifmmode

```

```

301 <*trace>
302   \ifnum \tracingfonts>\tw@
303     \count@#2\relax
304     \@font@info{Using \noexpand\mathgroup
305               (\the\count@) #2}\fi
306 </trace>

```

If so we first call the ‘=’ macro (i.e. argument three) to set up special things for the selected math group. Then we call `\mathgroup` to select the group given by argument two and finally we place `#1` (i.e., the argument of the `<math alphabet identifier>`) at the end. This part of the code is surrounded by two commands which behave like `\begingroup` and `\endgroup` if we want `<math alphabet identifier>`s but will expand into `\@empty` if we want simply switches to a new math group. Since argument number 2 may be a digit instead of a control sequence we add a `\relax`. Otherwise something like `\mit{1}` would switch to math group 11 (and back) instead of printing an oldstyle 1.

```

307   \math@bgroup
308     \expandafter\ifx\csname M@\f@encoding\endcsname#1\else
309     #1\fi
310     \mathgroup#2\relax

```

Before we reinsert the swallowed token (arg. three) into the input stream, in the case that the `<math alphabet identifier>` isn’t called in math mode, we remove the `\fi` with the `\expandafter` trick. This is necessary if the token is actually an macro with arguments. In such a case the `\fi` will be misinterpreted as the first argument which would be disastrous.

```

311   \expandafter\math@egroup\fi}%

```

The surrounding macros equal `\begingroup` and `\endgroup`. But using internal names makes it possible to overwrite their meaning in certain cases. This is for example used in $\mathcal{A}\mathcal{M}\mathcal{S}$ - $\mathcal{T}\mathcal{E}\mathcal{X}$ macros for placing accents.

(End of definition for \use@mathgroup.)

`\math@egroup` If the `margid` option is in force (which can be tested by looking at the definition of `\math@bgroup`) we change the `\math@egroup` command a bit to display the current `<math group number>` after it closes the scope of `<math alphabet>` with `\endgroup`.

```

312 <*trace>
313   \ifx\math@bgroup\bgroup
314     \def\math@egroup#1{#1\egroup
315       \ifnum \tracingfonts>\tw@
316         \@font@info{Restoring \noexpand\mathgroup
317               (\ifnum\mathgroup=\m@ne default\else \the\mathgroup \fi)%
318               }\fi}
319   \fi
320 </trace>

```

(End of definition for \math@egroup.)

`\getanddefine@fonts` `\getanddefine@fonts` has two arguments: the `<math group number>` and the *family/series/shape* name as a control sequence.

```

321 \def\getanddefine@fonts#1#2{%

```

First we turn of tracing when `\tracingfonts` is less than 4.

```

322 <+debug> \pushtracing
323 <+debug> \ifnum\tracingfonts<4 \tracingoff
324 <+debug> \else \tracingon\getanddefine@fonts \fi

325 <*trace>
326 \ifnum \tracingfonts>\tw@
327 \count@#1\relax
328 \font@info{\noexpand\mathgroup (\the\count@) #1 :=\MessageBreak
329 \string#2 \tf@size/\sf@size/\ssf@size}\fi
330 </trace>

```

We append the current `\tf@size` to `#2` to obtain the font name.³⁶ Again, `font@name` is defined globally, for the reasons explained in the description of `\wrong@fontshape`.

```

331 \xdef\font@name{\csname \string#2/\tf@size\endcsname}%

```

Then we call `\pickup@font` to load it if necessary. We remember the internal name as `\textfont@name`.

```

332 \pickup@font \let\textfont@name\font@name

```

Same game for `\scriptfont` and `\scriptscriptfont`:

```

333 \xdef\font@name{\csname \string#2/\sf@size\endcsname}%
334 \pickup@font \let\scriptfont@name\font@name
335 \xdef\font@name{\csname \string#2/\ssf@size\endcsname}%
336 \pickup@font

```

Then we append the new `\textfont...` assignments to the `\math@fonts`.

```

337 \edef\math@fonts{\math@fonts
338 \textfont#1\textfont@name
339 \scriptfont#1\scriptfont@name
340 \scriptscriptfont#1\font@name}%

```

Just before ending this macro we have to pop the tracing stack if it was pushed before.

```

341 <+debug> \poptracing
342 }
343 </2ekernel | package>

```

(End of definition for \getanddefine@fonts.)

6 Scaled font extraction

`\ifnot@nil` We begin with a simple auxiliary macro. It checks whether its argument is the token `\@nil`. If so, it expands to `\@gobble` which discards the following argument, otherwise it expands to `\@firstofone` which reproduces its argument.

```

344 <*2ekernel>
345 \def\ifnot@nil#1{\def\reserved@a{#1}%
346 \ifx\reserved@a\@nnil \expandafter\@gobble
347 \else \expandafter\@firstofone\fi}

```

(End of definition for \ifnot@nil.)

³⁶One might ask why this expansion does not generate a macro name that starts with an additional backslash character. The solution is that `\escapechar` is set to `-1` before `\getanddefine@fonts` is called.

`\remove@to@nnil` Three other auxiliary macros will be needed in the following: `\remove@to@nnil` gobbles up everything up to, and including, the next `\@nnil` token, and `\remove@angles` and `\remove@star` do the same for the character `>` and `*`, respectively, instead of `\@nnil`.

```

348 \def\remove@to@nnil#1\@nnil{}
349 \def\remove@angles#1>{\set@simple@size@args}
350 \def\remove@star#1*{#1}

```

(End of definition for `\remove@to@nnil`, `\remove@angles`, and `\remove@star`.)

`\extract@sizefn` This macro takes a size specification and parses it into size function and the optional and mandatory arguments.

```

351 \def\extract@sizefn#1*#2\@nnil{%
352   \if>#2>\set@size@funct@args#1\@nnil
353     \let\sizefn@info\@empty
354   \else\expandafter\set@size@funct@args\remove@star#2\@nnil
355     \def\sizefn@info{#1}\fi
356   }

```

(End of definition for `\extract@sizefn`.)

`\try@simple@size` This function tries to extract the given size (specified by `\f@size`) for the requested font shape. The font information must already be present in `\font@info`. The central macro that does the real work is `\extract@fontinfo`. We will first give a simple example how this macro works, and describe it in full generality later.

Assume that the requested parameters are: *encoding scheme* ‘OT1’, *family* ‘cm’, *series* ‘sansserif’, *shape* ‘normal’, and *size* ‘12’. The corresponding font definitions have already been extracted from the macro `\OT1/cm/sansserif/normal` and stored in `font@info`. (Otherwise `\extract@fontinfo` doesn’t get called.) This information consists of a token list made of characters of category code 12 of the form

```
<10*>cmss10<12*>cmss12<17*>cmss17
```

For reasonable packages one usually needs more sizes but this is sufficient to get the flavour. We will define a macro `\extract@fontinfo` to find the external font name (‘cmss12’) for us:

```

\def\extract@fontinfo#1<12*#2>#3<#4\@nnil{%
  \set@simple@size@args#3<#4\@nnil
  \execute@size@function{#2}}

```

so that when it gets called via

```
\extract@fontinfo<10*>cmss10<12*>cmss12<17*>cmss17\@nnil
```

`#1` will contain all characters before `<12*>`, `#2` will be empty, `#3` will be exactly `cmss12`, and `#4` will be `17>cmss17`. The expansion is therefore

```

\set@simple@size@args cmss12<17*>cmss17\@nnil
\execute@size@function{}

```

This means: the default (empty) size function will be executed, with its optional argument set to empty and its mandatory argument set to `cmss12` by `\set@simple@size@args`. As we discussed earlier, the effect of the default size function is to load the given external font (`cmss12`) at the specified size (12)—which is exactly what was intended.

But this is only part of the whole story. It may be that the size requested does not occur in the token list `\font@info`. And the simple definition of `\extract@fontinfo` we gave above does not allow to specify give more than one size specification in front of the external font name.

Let's address these two problems separately. The first one is solved with the following trick: We define `\extract@fontinfo` as follows:

```
\def\extract@fontinfo#1<12*#2>#3<#4\@nnil{%
  \ifnot@nil{#3}%
    {\set@simple@size@args#3<#4\@nnil
      \execute@size@function{#2}%
    }%
}
```

How does this work? We call `\extract@fontinfo` via

```
\expandafter\extract@fontinfo\font@info<12*>\@nil\@nnil
```

i.e. by appending `<12*>\@nil\@nnil`. If the size ('12' in this case) appears in `\font@info` everything works as explained above, the only difference being that argument #4 of `\extract@fontinfo` additionally gets the tokens `<12*>\@nil\@nnil`. However, if the size is not found everything up to the final `<12*>` is in argument #1, #3 gets `\@nil`, and #2 and #4 are empty. The macro `\ifnot@nil` will discard the calls to `\set@simple@size@args` and `execute@size@function`, and hence `\font@info` will continue to be equal to `\@empty`. This means that no simple size specification matching the requested size could be found.

The second problem (more than one simple size specification for one external font name) will be addressed in `\set@simple@size@args` below.

The macros are hidden inside other control sequences so that we have to build `\extract@fontinfo` in several steps.

So here's the actual definition of `\extract@font` in `\try@simple@size`.

```
357 % % this could be replaced by \try@size@range making the subst slower!
358 \def\try@simple@size{%
```

`\reserved@a` is made an abbreviation for the head of the definition of the macro `\extract@fontinfo`.

```
359   \def\reserved@a{\def\extract@fontinfo###1}%
```

Now we can define `\extract@fontinfo`. Here we handle a small but convenient variation: in case of the default (empty) size function it is allowed to omit the `*` character.

```
360   \expandafter\reserved@a\expandafter<\f@size>##2<##3\@nnil{%
361     \ifnot@nil{##2}%
362       {\set@simple@size@args##2<##3\@nnil
363         \execute@size@function\sizefn@info
364       }%
}
```

Now we call `\extract@fontinfo`. Note the `<\@nil` tokens at the end.

```
365   \expandafter\expandafter
366   \expandafter\extract@fontinfo\expandafter\font@info
367   \expandafter<\f@size>\@nil\@nnil
368 }
```

(End of definition for `\try@simple@size`.)

`\set@simple@size@args` As promised above, the macro `\set@simple@size@args` will handle the case of several size specifications in a row. If another size specification follows, the very first token of its argument list is the character `<`. By starting the definition as follows,

```
369 \def\set@simple@size@args#1<{%
```

parameter `#1` is empty in this case, and contains the size function's arguments otherwise. We distinguish these two cases (Note that the character `<` cannot appear in `#1`) by calling `\remove@angles` for empty `#1` and `\extract@sizefn` otherwise. In the latter case we have to take care of the remaining character tokens and discard them. This is done by `\remove@to@nnil`. Note also the use of Kabelschacht's method.

```
370     \if<#1<{%
371         \expandafter\remove@angles
372     \else
373         \extract@sizefn#1*\@nil
374         \expandafter\remove@to@nnil
375     \fi}
```

(End of definition for `\set@simple@size@args`.)

Now, we are through with the case of a simple size, except for calling the size function. This will be handled later, as it is the same mechanism for all types of size specification. We will now proceed to macros for extraction of size range specification.

`\extract@rangefontinfo` `\extract@rangefontinfo` goes through a font shape definition in the input until it recognizes the tokens `<\@nil->`. It looks for font ranges with font size functions. It's operation is rather simple: it discards everything up to the next size specification and passes this on to `\is@range` for inspection. The specification (parameter `#2`) is inserted again, in case it is needed later.

```
376 \def\extract@rangefontinfo#1<#2>{%
377     \is@range#2->\@nil#2>}
```

(End of definition for `\extract@rangefontinfo`.)

`\is@range` `\is@range` is again a sort of dispatcher macro: if the size specification it is looking at is not a range specification it discards it and calls `\extract@rangefontinfo` to continue the search. Otherwise it calls `\check@range` to check the requested size against the specified range.

From the way `\is@range` is called inside `\extract@rangefontinfo` we see that `#2` is the character `>` if the size specification found is a simple one (that does not contain a `-` character). This is checked easily enough and `\extract@rangefontinfo` called again. Note that the extra tokens inserted after the `\@nil` in the call to `\is@range` appear at the beginning of the first argument to `\extract@rangefontinfo` and are hence ignored.

```
378 \def\is@range#1-#2\@nil{%
379     \if>#2\expandafter\check@single\else
380     \expandafter\check@range\fi}
```

(End of definition for `\is@range`.)

`\check@range` `\check@range` takes lower bound as parameter `#1`, upper bound as `#2`, size function as `#3` and the size function's arguments as `#4`. If `#3` is the special token `\@nil \font@info` is exhausted and we can stop searching.

```
381 \def\check@range#1-#2>#3<#4\@nnil{%
382     \ifnot@nil{#3}{%
```

If #3 wasn't \@nil we have a range. We start by assuming that we have to recurse. Note that we have to reinsert an < as it was already removed by scanning.

```
383 \def\reserved@f{\extract@rangefontinfo<#4\@nnil}%
```

We have to make sure that both boundaries are present, if not we have to set them. Here we check the upper bound. If \upper@bound is zero after the assignment we set it to \maxdimen (upper open range). We need to use a <dimen> register for the scan since we may have a decimal number as the boundary.

```
384 \upper@bound0#2\p@
385 \ifdim\upper@bound=z@ \upper@bound\maxdimen\fi
```

Now we check the upper boundary against \f@size. If it is larger or equal than \f@size this range is no good and we have to recurse.

```
386 \ifdim \f@size \p@<\upper@bound
```

Otherwise we have to check the lower bound. This time it is not necessary to scan the boundary value into a register because if it is empty we get zero as desired. We could even omit the 0 which would result in 1pt as default lower boundary. If \f@size is smaller than the boundary we have to recurse.

```
387 \lower@bound0#1\p@
388 \ifdim \f@size \p@<\lower@bound
389 \else
```

If both tests are passed we can try executing the size function.

```
390 \set@simple@size@args#3<#4\@nnil
391 \execute@size@function\sizefn@info
```

If the function was successful it should have left an external font name in \external@font. We use this to see if we can stop scanning. Otherwise we recurse.

```
392 \ifx\external@font\@empty
393 \else
394 \let\reserved@f\@empty
395 \fi
396 \fi
397 \fi
398 \reserved@f}}
```

(End of definition for \check@range.)

\lower@bound We use two dimen registers \lower@bound and \upper@bound to store the lower and upper endpoints of the range we found.

```
399 \newdimen\lower@bound
400 \newdimen\upper@bound
```

(End of definition for \lower@bound and \upper@bound.)

\check@single \check@single takes the size as parameter #1, size function as #2 and the size function's arguments as #3. We can assume that there is always something in the pipeline since the very last entry is a faked range (see above).

```
401 \def\check@single#1>#2<#3\@nnil{%
```

We start by assuming that we have to recurse. Note that we have to reinsert an < as it was already removed by scanning.

```
402 \def\reserved@f{\extract@rangefontinfo<#3\@nnil}%
```

Now we check the size against `\f@size`. If it is not equal `\f@size` it is no good and we have to recurse.

```
403     \ifdim \f@size \p@=#1\p@
```

Otherwise if this test is passed we can try executing the size function.

```
404     \set@simple@size@args#2<#3\@nnil
405     \execute@size@function\sizefn@info
```

If the function was successful it should have left an external font name in `\external@font`. We use this to see if we can stop scanning. Otherwise we recurse.

```
406     \ifx\external@font\@empty
407     \else
408     \let\reserved@f\@empty
409     \fi
410     \fi
411     \reserved@f}
```

(End of definition for `\check@single`.)

`\set@size@funct@args` This macro sets the optional and mandatory arguments for a size function. If the optional argument is not present it is set to the empty token list. The mandatory argument is delimited by the token `\@nil`.

```
412 \def\set@size@funct@args{\@ifnextchar[%
413   \set@size@funct@args@\set@size@funct@args@[]]}
414 \def\set@size@funct@args@#1#2\@nil{%
415   \def\mandatory@arg{#2}%
416   \def\optional@arg{#1}}
417 </2ekernel)
```

(End of definition for `\set@size@funct@args` and `\set@size@funct@args@`.)

`\DeclareSizeFunction` This function defines a new size function hiding the internal from the designer. The body of the size function may use `\optional@arg` and `\mandatory@arg` denoting the optional and mandatory argument that may follow the size specification `<...>`.

```
418 <*2ekernel)
419 \def\DeclareSizeFunction#1#2{\@namedef{s@fct@#1}{#2}}
420 \@onlypreamble\DeclareSizeFunction
421 </2ekernel)
```

(End of definition for `\DeclareSizeFunction`.)

`\execute@size@function` This macro is very simple. The only point worth noting is that calling an undefined size function will do nothing (actually execute a `\relax`).

```
422 <*2ekernel | package)
423 \def\execute@size@function#1{%
424 <*trace)
425     \ifundefined{s@fct@#1}%
426     {\errmessage{Undefined font size function #1}%
427     \s@fct@}%
428     {\csname s@fct@#1\endcsname}%
429 </trace)
430 <-trace)     \csname s@fct@#1\endcsname
431 }
432 </2ekernel | package)
```

(End of definition for `\execute@size@function`.)

`\try@size@range` This macro tries to find a suitable range for requested size (specified by `\f@size`) in `\font@info`. All the relevant action is done in `\extract@rangefontinfo`. All that needs to be done is to stuff in the token list in `\font@info` so that `\extract@rangefontinfo` can inspect it. Note the `<*\@nil>` token at the end to stop scanning.

```
433 {*2kernel)
434 \def\try@size@range{%
435     \expandafter\extract@rangefontinfo\font@info <*\@nil\@nnil
436 }
```

(End of definition for `\try@size@range`.)

`\try@size@substitution` This is the last thing that can be tried. If the desired `\f@size` is found neither among the simple size specifications nor in one of the ranges the whole list of size specifications is searched for a nearby simple size.

```
437 \gdef\try@size@substitution{%
```

First we do some initializations. `\@tempdimb` will hold the difference between the wanted size and the best solution found so far, so we initialise it with `\maxdimen`. The macro `\best@size` will hold the best size found, nothing found is indicated by the empty value.

```
438     \@tempdimb \maxdimen
439     \let \best@size \empty
```

Now we loop over the specification

```
440     \expandafter \try@simples \font@info <\number\M>\@nil\@nnil
441 }
```

(End of definition for `\try@size@substitution`.)

`\font@submax` The macro `\font@submax` records the maximal deviation from the desired size encountered so far. Its value is used in a warning message at `\end{document}`. The macro `\fontsubfuzz` contains the amount that will not cause terminal warnings (warnings still go into the transcript file).

```
442 \def\font@submax{0pt}
443 \def\fontsubfuzz{.4pt}
444 {/2kernel)
445 {+package}\def\fontsubfuzz{0pt}
```

(End of definition for `\font@submax` and `\fontsubfuzz`.)

`\try@simples` `\try@simples` goes through a font shape definition in the input until it recognizes the tokens `<*\@nil>`. It looks for simple sizes to determine the two closest sizes. It is assumed that simple sizes are in increasing order.

```
446 {*2kernel)
447 \gdef\try@simples#1<#2>{%
448     \tryif@simple#2->\tryif@simple}
```

(End of definition for `\try@simples`.)

`\tryis@simple` `\tryis@simple` is similar to `\is@range`. If it sees a simple size, it checks it against the value of `\f@size` and sets `\lower@font@size` or `\higher@font@size`. In the latter case, it stops the iteration. By adding `<\number\@M>` at the end of the line we always have an end point. This is a hack which probably should be corrected.

First it checks whether it is finished already, then whether the size specification in question is a simple one.

```
449 \gdef\tryif@simple#1-#2\tryif@simple{%
```

Most common case for `\reserved@f` first:

```
450 \let \reserved@f \try@simples
451 \if>#2%
```

If so, it compares it to the value of `\f@size`. This is done using a `dimen` register since there may be fractional numbers.

```
452 \dimen@ #1\p@
453 \ifdim \dimen@<\@M\p@
```

If `\dimen@` is `\@M\p@` we have reached the end of the fontspec (hopefully) otherwise we compare the value with `\f@size` and compute in `\@tempdimc` the absolute value of the difference between the two values.

```
454 \ifdim \f@size\p@<\dimen@
455 \@tempdimc \dimen@
456 \advance\@tempdimc -\f@size\p@
457 \else
458 \@tempdimc \f@size\p@
459 \advance\@tempdimc -\dimen@
460 \fi
```

The result is then compared with the smallest difference we have encountered, if the new value (in `\@tempdimc` is smaller) we have found a size which is a better approximation so we make it the `\best@size` and adjust `\@tempdimb`.

```
461 \ifdim \@tempdimc<\@tempdimb
462 \@tempdimb \@tempdimc
463 \def \best@size{#1}%
464 \fi
```

When we have reached the end of the fontspec we substitute the best size found (if any). We code this inline to save macro space; in the past this was done by a macro called `\subst@size`.

```
465 \else
```

`\subst@size` This macro substitutes the size recorded in `\best@size` for the unavailable size `\f@size`. `\font@submax` records the maximum difference between desired size and selected size in the whole run.

```
466 % \subst@size %% coded inline
467 % \def\subst@size{%
468 \ifx \external@font\@empty
469 \ifx \best@size\@empty
470 \else
471 \ifdim \@tempdimb>\font@submax \relax
472 \xdef \font@submax {\the\@tempdimb}%
473 \fi
474 \let \f@user@size \f@size
475 \let \f@size \best@size
```

```

476     \ifdim \@tempdimb>\fontsubfuzz\relax
477     \font@warning{Font\space shape\space
478     ‘\curr@fontshape’\space in\space size\space
479     <\f@user@size>\space not\space available\MessageBreak
480     size\space <\f@size>\space substituted}%
481     \fi
482     \try@simple@size
483     \do@subst@correction
484     \fi
485     \fi
486 % %}

```

This brings us back into the main part of `\tryif@simple`. Finally we get rid of any rubbish left over on the input stack.

```

487     \let \reserved@f \remove@to@nnil
488     \fi
489     \fi

```

If it's a range iterate also.

```

490     \reserved@f}

```

(End of definition for \tryis@simple and \subst@size.)

6.1 Sizefunctions

In the following we define some useful size functions.

`\s@fct@` This is the default size function. Mandatory argument is an external font name, optional argument a scale factor. The font is scaled to `\f@size` if no optional argument is present, and to `\f@size` multiplied by the optional argument otherwise.

```

491 \DeclareSizeFunction{}{\empty@sfcnt\@font@warning}
492 \DeclareSizeFunction{s}{\empty@sfcnt\@font@info}
493 \def\empty@sfcnt#1{%
494     \@tempdimb \f@size\p@
495     \ifx\optional@arg\@empty
496     \else
497     \@tempdimb \optional@arg\@tempdimb
498     #1{Font\space shape\space ‘\curr@fontshape’\space
499     will\space be\MessageBreak
500     scaled\space to\space size\space \the\@tempdimb}%
501     \fi
502     \edef\external@font{\mandatory@arg\space at\the\@tempdimb}}

```

(End of definition for \s@fct@.)

`\s@fct@gen` This size function generates the external name from the mandatory argument and the requested user size, and thus can be used for external names where the size is encoded in the font name. The optional argument a scale factor. The font is scaled to `\f@size` if no optional argument is present, and to `\f@size` multiplied by the optional argument otherwise.

```

503 \DeclareSizeFunction{gen}{\gen@sfcnt\@font@warning}
504 \DeclareSizeFunction{sgen}{\gen@sfcnt\@font@info}

```



```

505 \def\gen@sfcnt{%
506     \edef\mandatory@arg{\mandatory@arg\f@size}%
507     \empty@sfcnt}

```

(End of definition for \sfct@gen and \sfct@sgen.)

`\sfct@gen` This size function is similar to `gen`, but for fonts where the size is encoded in the font name in centipoints, as in the DC fonts version 1.2. The font is scaled to `\f@size` if no optional argument is present, and to `\f@size` multiplied by the optional argument otherwise.

```

508 \DeclareSizeFunction{genb}{\genb@sfcnt\@font@warning}
509 \DeclareSizeFunction{sgenb}{\genb@sfcnt\@font@info}

510 \def\genb@sfcnt{%
511     \edef\mandatory@arg{\mandatory@arg\expandafter\genb@x\f@size..\@{}}%
512     \empty@sfcnt}

```

(End of definition for \sfct@genb and \sfct@sgenb.)

`\genb@x` The auxiliary macros `\genb@x` and `\genb@y` are used to convert the `\f@size` into centipoints.

```

513 \def\genb@x#1.#2.#3\@{\two@digits{#1}\genb@y#200\@{}}
514 \def\genb@y#1#2#3\@{\@{#1#2}}

```

(End of definition for \genb@x and \genb@y.)

`\sfct@sub` This size function handles font substitution. The mandatory argument is a family/series/shape combination, the optional argument (if present) is ignored. The font encoding scheme cannot be changed. Therefore, the first thing we do is to prepend the encoding scheme.

```

515 \DeclareSizeFunction{sub}{\sub@sfcnt\@font@warning}
516 \DeclareSizeFunction{ssub}{\sub@sfcnt\@font@info}

517 \def\sub@sfcnt#1{%
518     \edef\mandatory@arg{\f@encoding/\mandatory@arg}%

```

Next action is split the arg into its individual components and allow for a late font shape load.

```

519     \begingroup
520     \expandafter\split@name\mandatory@arg/\@nil
521     \try@load@fontshape
522     \endgroup

```

Then we record the current `\f@size` since it may get clobbered.

```

523     \let\f@user@size\f@size

```

Then we check whether this new combination is defined and give an error message if not. In this case we also switch to `\error@fontshape`.

```

524     \expandafter
525     \ifx\csname\mandatory@arg\endcsname\relax
526         \errmessage{No\space declaration\space for\space
527             shape\space \mandatory@arg}%
528     \error@fontshape
529     \else

```

Otherwise we warn the user about the substitution taking place.

```

530     #1{Font\space shape\space '\curr@fontshape'\space in\space
531         size\space <\f@size>\space not\space available\MessageBreak
532         Font\space shape\space '\mandatory@arg'\space tried\space
533         instead}%
534     \expandafter\split@name\mandatory@arg/\@nil
535     \fi

```

Then we restart the font specification scan by calling `\get@external@font`.

```

536     \edef\f@size{\f@user@size}%
537     \get@external@font

```

Finally `\do@subst@correction` is called to get the font name right.

```

538     \do@subst@correction
539 }

```

(End of definition for `\s@fct@sub`.)

`\@font@aliasinfo` Sometimes a substitution is only done to map a long font name to a standard shape or series, e.g.,

```
DeclareFontShape{T1}{Roboto-LF}{b}{it}{<-> alias * Roboto-LF/bold/it}{}
```

Using the `ssub` function in that case will give a strange (and incorrect) warning. As an alternative we therefore offer the size function `alias`. It will still add some info into the `.log` file, but no longer complains that the font shape is not available. It is implemented by grabbing the default warning text and replacing it with a new one.

```

540 </2ekernel>
541 <*2ekernel | latexrelease>
542 <latexrelease>\IncludeInRelease{2020/02/02}%
543 <latexrelease>           {\@font@aliasinfo}{alias size function}%
544 \DeclareSizeFunction{alias}{\sub@sfcnt \@font@aliasinfo}
545 \def \@font@aliasinfo#1{%
546     \@font@info{Font\space shape\space '\curr@fontshape'\space
547         aliased\space to\MessageBreak '\mandatory@arg'}%
548 }
549 </2ekernel | latexrelease>
550 <latexrelease>\EndIncludeInRelease
551 <latexrelease>\IncludeInRelease{0000/00/00}%
552 <latexrelease>           {\@font@aliasinfo}{alias size function}%
553 <latexrelease>\let\s@fct@alias \@undefined
554 <latexrelease>\let \@font@aliasinfo \@undefined
555 <latexrelease>
556 <latexrelease>\EndIncludeInRelease
557 <*2ekernel>

```

(End of definition for `\@font@aliasinfo`.)

`\s@fct@subf` The `subf` size function allows substitution of another font. The mandatory argument is the external name of the font to be substituted, the optional argument a size scaling factor like in the default size function. The main difference to the default size function is the warning message.

```

558 \DeclareSizeFunction{subf}{\sub@sfcnt \@font@warning}
559 \DeclareSizeFunction{ssubf}{\sub@sfcnt \@font@info}

```

```

560 \def\subf@sfcnt#1{%
561     #1{Font\space shape\space '\curr@fontshape'\space in\space
562     size\space \f@size\space not\space available\MessageBreak
563     external\space font\space '\mandatory@arg'\space used}%
564     \empty@sfcnt#1%
565 }

```

(End of definition for \s@fct@subf.)

\s@fct@fixed The **fixed** size function is for using a font at a different size than requested. A warning message is printed, and the external font to be used is taken from the mandatory argument. If an optional argument is present it is used as the ‘at’ size for the font. Otherwise the font is loaded at its design size.

```

566 \DeclareSizeFunction{fixed}{\fixed@sfcnt\@font@warning}
567 \DeclareSizeFunction{sfixed}{\fixed@sfcnt\@font@info}
568 \def\fixed@sfcnt#1{%
569     \ifx\optional@arg\@empty
570         \let\external@font\mandatory@arg
571     \else
572         \edef\external@font{\mandatory@arg\space at\optional@arg pt}%
573     \fi
574     #1{External\space font\space '\external@font'\space loaded\space
575     for\space size\MessageBreak
576     <\f@size>}%
577 }
578 \</2ekernel>

```

(End of definition for \s@fct@fixed.)

File 27

ltfsscmp.dtx

This file contains the implementation of commands giving compatibility with the original ‘NFSS1’ release of the Font Selection Scheme.

Warning: The macro documentation is still basically the documentation from the first NFSS release and therefore in some cases probably not completely accurate.

Version 1 of NFSS is obsolete now for about 20 years (and was “current” only for a short intermediate time) so with the 2015 release these internal interface commands are removed from the kernel and made available via `latexrelease` package so that backward compatibility remains ensured for very old documents.

```
1 <(*latexrelease)
2 \IncludeInRelease{2015/01/01}{\new@fontshape}%
3                                     {NFSS version1 commands}%
4 \let\new@fontshape\@undefined
5 \let\warn@rel@i\@undefined
6 \let\scan@fontshape\@undefined
7 \let\scan@@fontshape\@undefined
8 \let\subst@fontshape\@undefined
9 \let\extra@def\@undefined
10 \let\default@mextra\@undefined
11 \let\preload@sizes\@undefined
12 \let\err@rel@i\@undefined
13 \let\newmathalphabet\@undefined
14 \let\newmathalphabet@\@undefined
15 \let\newmathalphabet@@@\@undefined
16 \let@if@no@font@opt\@undefined
17 \let\@no@font@optfalse\@undefined
18 \let\define@mathalphabet\@undefined
19 \let\define@mathgroup\@undefined
20 \let\addtoversion\@undefined
21 \EndIncludeInRelease
```

In older releases we provide the original definitions.

```
22 \IncludeInRelease{0000/00/00}{\new@fontshape}%
23                                     {NFSS version1 commands}%
```

`\new@fontshape` The interface is now `\DeclareFontShape`.

```
24 \gdef\new@fontshape#1#2#3#4{%
25     \warn@rel@i\new@fontshape\DeclareFontShape
26     \expandafter\scan@fontshape\@gobble#4<\@nil><<%
27     \DeclareFontShape U{#1}{#2}{#3}\reserved@f}%
28 \@onlypreamble\new@fontshape
```

(End of definition for \new@fontshape.)

`\warn@rel@i` The warning message used above.

```
29 \gdef\warn@rel@i#1#2{%
30     \@font@warning{*** NFSS release 1 command
```

```

31         \noexpand#1found\MessageBreak
32     *** Update by using release 2 command
33         \string#2.\MessageBreak
34     *** Recovery is probably possible}%
35 }%
36 \@onlypreamble\warn@rel@i

```

(End of definition for \warn@rel@i.)

\scan@fontshape This will scan the old font shape definition syntax.

```

37 \gdef\scan@fontshape{%
38     \let\reserved@f\@empty
39     \let\reserved@e\@empty %           holds last info
40     \scan@@fontshape
41 }%
42 \@onlypreamble\scan@fontshape

```

(End of definition for \scan@fontshape.)

\scan@@fontshape

```

43 \gdef\scan@@fontshape#1>#2#3<{%
44     \ifx\@nil#1%
45         \edef\reserved@f{\reserved@f\reserved@e}%
46     \else
47         \def\reserved@b{#1}%           nick names
48         \def\reserved@c{#3}%
49         \in@{ at}{#3}%
50         \ifin@
51             \in@{pt}{#3}% not a proof but a good chance
52         \ifin@

```

We grab also everything after pt and discard it if people have forgotten to place a percent sign there.

```

53         \def\reserved@a##1 at##2pt##3\@nil{%
54             \def\reserved@b{##2}%
55             \def\reserved@c{##1}%
56         }%
57     \reserved@a#3\@nil
58     \fi
59     \fi
60     \ifnum 0<0#2
61         \edef\reserved@d{subf*\reserved@c}%
62         \ifcase #2\or
63             \or
64             \else
65                 \errmessage{*** What's this? NFSS release 0? ***}%
66             \fi
67         \else
68             \edef\reserved@d{#2\reserved@c}%
69         \fi
70     \ifx\reserved@d\reserved@e
71         \edef\reserved@f{\reserved@f<\reserved@b>}%
72     \else
73         \edef\reserved@f{\reserved@f\reserved@e<\reserved@b>}%add old info
74         \let\reserved@e\reserved@d

```

```

75     \fi
76     \expandafter\scan@@fontshape
77     \fi
78 }%
79 \@onlypreamble\scan@@fontshape

```

(End of definition for \scan@@fontshape.)

`\subst@fontshape` This is now also handled by the extend syntax of `\DeclareFontShape`.

```

80 \gdef\subst@fontshape#1#2#3#4#5#6{%
81     \warn@rel@i\subst@fontshape\DeclareFontShape
82     \DeclareFontShape{U}{#1}{#2}{#3}{<->sub*#4/#5/#6}{}}%
83 \@onlypreamble\subst@fontshape

```

(End of definition for \subst@fontshape.)

`\extra@def` This was replaced by `\DeclareFontFamily`.

```

84 \gdef\extra@def#1#2#3{%
85     \warn@rel@i\extra@def\DeclareFontFamily
86     \DeclareFontFamily{U}{#1}{}%
87 }%
88 \@onlypreamble\extra@def

```

(End of definition for \extra@def.)

`\default@mextra` The new name is `\DeclareFontEncodingDefaults` but in this case we don't feel comfortable with this either.

```

89 \gdef\default@mextra{%
90     \warn@rel@i\default@mextra\DeclareFontEncodingDefaults

```

We pick up the argument to `\default@mextra` implicitly as the second argument of `\DeclareFontEncodingDefaults`.

```

91     \DeclareFontEncodingDefaults\relax
92 }%
93 \@onlypreamble\default@mextra

```

(End of definition for \default@mextra.)

`\preload@sizes` The new interface is `\DeclarePreloadSizes`.

```

94 \gdef\preload@sizes{%
95     \warn@rel@i\preload@sizes\DeclarePreloadSizes
96     \DeclarePreloadSizes U%
97 }%
98 \@onlypreamble\preload@sizes

```

(End of definition for \preload@sizes.)

`\err@rel@i` This macro is used in cases where emulation with NFSS2 features is not really possible.

```

99 \gdef\err@rel@i#1#2{%
100     \@latex@error{*** NFSS release 1 command \noexpand#1found%
101         ^^J*** Recovery not possible. Use \string#2}%
102     {The new release of NFSS doesn't support the
103     \noexpand#1command^^Jany longer.
104     Please upgrade your file to the syntax of NFSS
105     release 2^^Jusing the \noexpand#2command.}%

```

Let's die.

```
106 \batchmode\input.\relax
107 }%
108 \@onlypreamble\err@rel@i
```

(End of definition for \err@rel@i.)

```
\newmathalphabet \newmathalphabet is the old form.
\newmathalphabet@@ 109 \gdef\newmathalphabet{%
\newmathalphabet@@@ 110 \ifno@font@opt
111 \latex@error{*** NFSS release 1 command
112 \noexpand\newmathalphabet found%
113 ^^J \space*** Automatic recovery not possible.%
114 ^^J \space*** TYPE H for Help%
115 }%
116 {Please look at the file usrguide.tex for hints on
117 how to resolve this problem.}%
118 \else
119 \warn@rel@i\newmathalphabet\DeclareMathAlphabet
120 \fi
121 \@ifstar\newmathalphabet@@@
122 \newmathalphabet@@}%
123 \gdef\newmathalphabet@@#1{\DeclareMathAlphabet#1{U}{-}{-}}%
124 \gdef\newmathalphabet@@@#1#2#3#4{%
125 \DeclareMathAlphabet{#1}{U}{#2}{#3}{#4}}%
126 \@onlypreamble\newmathalphabet
127 \@onlypreamble\newmathalphabet@@
128 \@onlypreamble\newmathalphabet@@@
```

(End of definition for \newmathalphabet, \newmathalphabet@@, and \newmathalphabet@@@.)

```
\ifno@font@opt
\@no@font@optfalse 129 \global\let\ifno@font@opt\iftrue
130 \gdef\@no@font@optfalse{\let\ifno@font@opt\iffalse}%
```

(End of definition for \ifno@font@opt and \@no@font@optfalse.)

```
\define@mathalphabet This is a case where dying is best.
131 \gdef\define@mathalphabet{%
132 \err@rel@i\define@mathalphabet\DeclareMathAlphabet
133 }%
134 \@onlypreamble\define@mathalphabet
```

(End of definition for \define@mathalphabet.)

```
\define@mathgroup And here is another one
135 \gdef\define@mathgroup{%
136 \err@rel@i\define@mathgroup\DeclareSymbolFont
137 }%
138 \@onlypreamble\define@mathgroup
```

(End of definition for \define@mathgroup.)

```
\addtoversion \addtoversion is the old form.
139 \def\addtoversion#1#2{%
140   \warn@rel@i\addtoversion\SetMathAlphabet
141   \SetMathAlphabet#2{#1}{U}}%
142 \@onlypreamble\addtoversion

(End of definition for \addtoversion.)
    Finishing off this huge \IncludeInRelease argument:
143 \EndIncludeInRelease
144 </latexrelease>
```


File 28

ltfssdcl.dtx

This file contains the main implementation of the font selection scheme commands. See other parts of the L^AT_EX distribution, or *The L^AT_EX Companion* for higher level documentation of these commands.

Warning: The macro documentation is still basically the documentation from the first NFSS release and therefore in some cases probably not completely accurate.

1 Interface Commands

```
1 <*2kernel>
2 \message{NFSS declarative interface,}

\in@ \@in is a utility macro with two arguments. It determines whether its first argument
\ifin@ occurs in its second and sets the switch \ifin@ accordingly. The first argument may not
contain braces nor # (more precisely, tokens of category code 1, 2, or 6).

3 \def\in@#1#2%
4 {%
5   \begingroup
6     \def\in@@##1#1{}%
7     \toks@\expandafter{\in@@#2{}-{}#1}%
8     \edef\in@@{\the\toks@}%
9     \expandafter\endgroup
10    \ifx\in@@@empty
11      \in@false
12    \else
13      \in@true
14    \fi
15  }
16 \newif\ifin@
```

(End of definition for \in@ and \ifin@.)

Before the `\begin{document}` command several *math versions* and *math alphabet identifiers* may be declared. In principle, there should be exactly one family/series/shape combination be declared for each version/alphabet pair. But we want to allow for defaults as well for automagical filling of holes.

While building the tables for math alphabet identifiers and math versions we keep several lists:

- the list of all math versions, `\version@list`, each entry prefixed by the control sequence `\version@elt`, i.e. this list has the following form

```
\version@elt<version1>\version@elt<version2>...
\version@elt<versionn>
```

- the list of all math alphabet identifiers. Here every entry has the form:

```
\group@elt<math group number>
{\{<default family>\}{<default series>\}{<default shape>\}}
```

- Each defined math alphabet identifier holds a list containing information about the *versions* for which it is defined. This list has a more complicated structure: it looks as follows:

```

\set@alpha<the alphabet identifier itself>
  \reserved@c<math version><font info>
  ...
\@nil

```

where ** is either `\reserved@e` (if the combination is not defined yet) or

```

{\<family>}{\<series>}{\<shape>}}

```

`\version@list` We initialize the version list to be empty.

```

17 \let\version@list=\@empty
18 \@onlypreamble\version@list

```

(End of definition for `\version@list`.)

`\version@elt`

```

19 \let\version@elt\relax
20 \@onlypreamble\version@elt

```

(End of definition for `\version@elt`.)

`\new@mathversion` The macro `\new@mathversion` is called with the version control sequence as its argument.

```

21 %\def\new@mathversion#1{%

```

The first thing this macro does is to check if the version identifier is already present in `\version@list`. We enclose `\version@list` in braces since it might be empty (if no *version* is defined yet). But this means that we need a suitable number of `\expandafter` primitives.

```

22 % \expandafter\in@\expandafter#1\expandafter{\version@list}%
23 % \ifin@

```

If so it prints an error message. The `\next` macro is used to get rid of the four characters `\mv@` that would otherwise appear at the begin of the version name in the error message.

```

24 % \latex@error{Math version
25 % \expandafter\@gobblefour\string#1'
26 % already defined}\@eha

```

Otherwise we have a new version, and we can proceed with entering it into the tables. We add it to `\version@list`. This is very easy: we define `\version@elt` (which is the delimiter in `\version@list`) to protect itself and the following token from being expanded and simply redefine `\version@list`.

```

27 % \else
28 % \global\expandafter\newcount\csname c@\expandafter
29 % \gobble\string#1\endcsname
30 % \global\csname c@\expandafter
31 % \gobble\string#1\endcsname\@ne
32 % \def\version@elt{\noexpand\version@elt\noexpand}%
33 % \edef\version@list{\version@list\version@elt#1}%

```

Then we prepare to enter the new version into all math alphabet identifier lists. Remember that these lists use `\reserved@c` as delimiter, and that there appears the control sequence `\reserved@e` that must not be expanded. Therefore we take suitable precautions.

```
34 % \def\reserved@c{\noexpand\reserved@c\noexpand}%
35 % \let\reserved@e\relax
```

We will now go through the `\alpha@list` to process every *math alphabet identifier* in turn. Since this list has `\group@elt` as a delimiter we define this control sequence. It has three arguments as every entry consists of three items (as explained above).

```
36 % \def\group@elt##1##2##3{%
```

The first of these arguments is the *math alphabet identifier*. We redefine it by appending the information about the new version at the end of the list contained in it. However, there is one subtlety: the definitions for `\reserved@c` and `\reserved@e` made above prevent the main part of the list from being expanded. But we still have to take care of the header and the trailer. To do this we remove the trailer by means of the macro `\remove@nil` which also protect the header from being expanded. Its definition is given below. Now we can prepare to add the new version.

```
37 % \edef##1{\expandafter\remove@nil##1%
38 % \reserved@c
39 % #1%
40 % \reserved@e
41 % \noexpand\@nil}}%
```

Finally we call `\alpha@list` which will now execute the macro `\group@elt` once for every defined *math alphabet identifier*. And that's all for now.

```
42 % \alpha@list
43 % \fi}
```

(End of definition for \new@mathversion.)

`\alpha@list` As we explained above every entry in `\alpha@list` has the form
`\alpha@elt`
`\alphabet identifier\internal group number\default font assignments`...
We initialize it to `\@empty`.

```
44 \let\alpha@list\@empty
45 \@onlypreamble\alpha@list
```

(End of definition for \alpha@list.)

`\alpha@elt`

```
46 \let\alpha@elt\relax
47 \@onlypreamble\alpha@elt
```

(End of definition for \alpha@elt.)

`\newgroup` Start the group (fam) allocation at 0. (Doesn't belong here.)

```
48 \count18=-1
```

(End of definition for \newgroup.)

`\stepcounter`

(End of definition for \stepcounter.)

`\select@group` We surround `\select@group` with braces so that functions using it can be used directly after `_` or `^`. However, if we use oldstyle syntax where the math alphabet doesn't have arguments (ie if `\math@bgroup` is not `\bgroup`) we need to get rid of the extra group.

```

49 </2ekernel>
50 <latexrelease>\IncludeInRelease{2015/01/01}
51 <latexrelease>          {\select@group}{\select@group}%
52 <*2ekernel | latexrelease>
53 \def\select@group#1#2#3#4{%
54   \ifx\math@bgroup\bgroup\else\relax\expandafter\@firstofone\fi
55   {%
56     \ifmmode
57       \ifnum\csname c@mv@\math@version\endcsname<\e@mathgroup@top
58         \begingroup
59           \escapechar\m@ne
60           \getanddefine@fonts{\csname c@mv@\math@version\endcsname}#3%
61           \globaldefs\@ne \math@fonts
62         \endgroup
63         \init@restore@version
64         \xdef#1{\noexpand\use@mathgroup\noexpand#2%
65             {\number\csname c@mv@\math@version\endcsname}}%
66         \global\advance\csname c@mv@\math@version\endcsname\@ne
67       \else
68         \let#1\relax
69         \@latex@error{Too many math alphabets used in
70             version \math@version}%
71         \@eha
72       \fi
73     \else \expandafter\non@alpherr\fi
74     #1{#4}%
75   }%
76 }
77 </2ekernel | latexrelease>
78 <latexrelease>\EndIncludeInRelease
79 <latexrelease>\IncludeInRelease{0000/00/00}
80 <latexrelease>          {\select@group}{\select@group}%
81 <latexrelease>\def\select@group#1#2#3#4{%
82 <latexrelease> \ifx\math@bgroup\bgroup\else\relax\expandafter\@firstofone\fi
83 <latexrelease> {%
84 <latexrelease> \ifmmode
85 <latexrelease>   \ifnum\csname c@mv@\math@version\endcsname<\sixt@n
86 <latexrelease>     \begingroup
87 <latexrelease>       \escapechar\m@ne
88 <latexrelease>       \getanddefine@fonts
89 <latexrelease>         {\csname c@mv@\math@version\endcsname}#3%
90 <latexrelease>       \globaldefs\@ne \math@fonts
91 <latexrelease>     \endgroup
92 <latexrelease>     \init@restore@version
93 <latexrelease>     \xdef#1{\noexpand\use@mathgroup\noexpand#2%
94 <latexrelease>         {\number\csname c@mv@\math@version\endcsname}}%
95 <latexrelease>     \global\advance\csname c@mv@\math@version\endcsname\@ne
96 <latexrelease>   \else
97 <latexrelease>     \let#1\relax
98 <latexrelease>     \@latex@error{Too many math alphabets used in
99 <latexrelease>         version \math@version}%

```

```

100 <latexrelease>      \@eha
101 <latexrelease>     \fi
102 <latexrelease>    \else \expandafter\non@alpherr\fi
103 <latexrelease>   #1{#4}%
104 <latexrelease>  }%
105 <latexrelease>}
106 <latexrelease>\EndIncludeInRelease
107 <*2ekernel>
108 \@onlypreamble\restore@mathversion

```

(End of definition for \select@group.)

`\init@restore@version`

```

109 \def\init@restore@version{%
110     \global\let\init@restore@version\relax
111     \xdef\restore@mathversion
112         {\expandafter\noexpand\csname mv@\math@version\endcsname
113          \global\csname c@mv@\math@version\endcsname
114          \number\csname c@mv@\math@version\endcsname\relax}%
115     \aftergroup\dorestore@version
116 }
117 \@onlypreamble\init@restore@version

```

(End of definition for \init@restore@version.)

`\non@alpherr`

```

118 \gdef\non@alpherr#1{\@latex@error{%

```

The command here will have a space at the end of its name, so we make sure not to insert an extra one.

```

119     \string#1allowed only in math mode}\@ehd}

```

(End of definition for \non@alpherr.)

`\dorestore@version`

```

120 \def\dorestore@version
121 {\ifmmode
122     \aftergroup\dorestore@version
123 \else
124     \gdef\init@restore@version{%
125         \global\let\init@restore@version\relax
126         \xdef\restore@mathversion
127             {\expandafter\noexpand\csname mv@\math@version\endcsname
128              \global\csname c@mv@\math@version\endcsname
129              \number\csname c@mv@\math@version\endcsname\relax}%
130         \aftergroup\dorestore@version
131     }%
132     \begingroup
133         \let\getanddefine@fonts@gobbletwo
134         \restore@mathversion
135     \endgroup
136 \fi}%
137 \@onlypreamble\dorestore@version

```

(End of definition for \dorestore@version.)

`\c@localmathalphabets` To avoid hitting the “no more math fams available” limit of 16, we keep a defined number of math alphabets flexible/local. If we have to allocate any of those we roll back the allocation after the formula has ended, so the next formula can use other alphabets in the slot(s). This makes the processing a bit slower if you are working at the limit, but that is better than dying with “out of memory”.

```

138 \endkernel
139 \latexrelease\IncludeInRelease{2021/11/15}
140 \latexrelease {\document@select@group}{\document@select@group}%
141 \*2ekernel | latexrelease

```

We don’t really undo the declaration on rollback (as that would be hard to maintain), so rolling forward needs to check if the declaration was already made.

```
142 \ifx\c@localmathalphabets\@undefined
```

There is no need to have this counter as part of the include checkpoints, given that it makes little sense to alter its settings mid document. All we want is the ability to change it using the `\setcounter` interface.

By default we keep two math fams flexible.

```

143 \newcount\c@localmathalphabets
144 \setcounter{localmathalphabets}{2}
145 \fi

```

(End of definition for \c@localmathalphabets.)

`\document@select@group` The `\document@select@group` command is the version of `\select@group` (inside math versions) that is used in the document body to set up math alphabets (if used).

```

146 \def\document@select@group#1#2#3#4{%
147 \ifx\math@bgroup\bgroup\else\relax\expandafter\@firstofone\fi
148 {%
149 \ifmode

```

We first check if there is still room for allocating another mathgroup. If there is, we check if it can be globally allocated or if we have reached the limit which is given by `\e@mathgroup@top` with `\c@localmathalphabets` subtracted.

```

150 \ifnum\csname c@mv@\math@version\endcsname<\e@mathgroup@top
151 \ifnum \numexpr\e@mathgroup@top-\c@localmathalphabets
152 >\csname c@mv@\math@version\endcsname
153 \else

```

If we are past this point we freeze the current state of the math version so that we can return to it after the formula has ended. Of course, that should be done only once, so we check if `\mv@<version>@frozen` already exists.

```
154 \ifcsname mv@\math@version @frozen\endcsname \else
```

We have to pass the current value of `\math@version` not the macro itself, because some of the processing is delayed to a point where the value may have changed again—not doing this caused a puzzling error in one setup.

```

155 \expandafter\freeze@math@version\expandafter{\math@version}%
156 \fi
157 \fi
158 \begingroup
159 \escapechar\m@ne
160 \getanddefine@fonts{\csname c@mv@\math@version\endcsname}#3%
161 \globaldefs\@ne \math@fonts

```

```

162 \endgroup
163 \expandafter\extract@alph@from@version
164 \csname mv@math@version\expandafter\endcsname
165 \expandafter{\number\csname
166 c@mv@math@version\endcsname}%
167 #1%
168 \global\advance\csname c@mv@math@version\endcsname\@ne
169 \else
170 \let#1\relax
171 \@latex@error{Too many math alphabets used in
172 version \math@version}%
173 \@eha
174 \fi

```

Extra `\expandafter` to remove the `\expandafter` added below

```

175 \else \expandafter\expandafter\expandafter\non@alpherr\fi

```

We surround `\select@group` with braces so that functions using it can be used directly after `_` or `^`.

If the legacy interface is used, e.g., `\sf -1` the math alphabet `#1` does not take an argument so we better do not surround `#4` with braces, because then we get `{\relax}` into the formula and introduce an extra Ord atom. The two different cases can be distinguished by looking at the current value of `\math@bgroup`.

```

176 \expandafter#1\ifx\math@bgroup\bgroup{#4}\else#4\fi
177 }%
178 }

```

```

179 </2kernel | latexrelease>
180 <latexrelease>\EndIncludeInRelease
181 <latexrelease>\IncludeInRelease{2020/10/01}
182 <latexrelease> {\document@select@group}{\document@select@group}%
183 <latexrelease>
184 <latexrelease>\def\document@select@group#1#2#3#4{%
185 <latexrelease> \ifx\math@bgroup\bgroup\else\relax\expandafter\@firstofone\fi
186 <latexrelease> {%
187 <latexrelease> \ifmmode
188 <latexrelease> \ifnum\csname c@mv@math@version\endcsname<\e@mathgroup@top
189 <latexrelease> \begingroup
190 <latexrelease> \escapechar\m@ne
191 <latexrelease> \getanddefine@fonts{\csname c@mv@math@version\endcsname}#3%
192 <latexrelease> \globaldefs\@ne \math@fonts
193 <latexrelease> \endgroup
194 <latexrelease> \expandafter\extract@alph@from@version
195 <latexrelease> \csname mv@math@version\expandafter\endcsname
196 <latexrelease> \expandafter{\number\csname
197 <latexrelease> c@mv@math@version\endcsname}%
198 <latexrelease> #1%
199 <latexrelease> \global\advance\csname c@mv@math@version\endcsname\@ne
200 <latexrelease> \else
201 <latexrelease> \let#1\relax
202 <latexrelease> \@latex@error{Too many math alphabets used
203 <latexrelease> in version \math@version}%
204 <latexrelease> \@eha
205 <latexrelease> \fi
206 <latexrelease> \else \expandafter\expandafter\expandafter\non@alpherr\fi

```

```

207 <latexrelease> \expandafter#1\ifx\math@bgroup\bgroup{#4}\else#4\fi
208 <latexrelease> }%
209 <latexrelease>}
210 <latexrelease>\EndIncludeInRelease
211 <latexrelease>\IncludeInRelease{2015/01/01}
212 <latexrelease> {\document@select@group}{\document@select@group}%
213 <latexrelease>
214 <latexrelease>\def\document@select@group#1#2#3#4{%
215 <latexrelease> \ifx\math@bgroup\bgroup\else\relax\expandafter\@firstofone\fi
216 <latexrelease> {%
217 <latexrelease> \ifmmode
218 <latexrelease> \ifnum\csname c@mv@\math@version\endcsname<\e@mathgroup@top
219 <latexrelease> \begingroup
220 <latexrelease> \escapechar\m@ne
221 <latexrelease> \getanddefine@fonts{\csname c@mv@\math@version\endcsname}#3%
222 <latexrelease> \globaldefs\@ne \math@fonts
223 <latexrelease> \endgroup
224 <latexrelease> \expandafter\extract@alph@from@version
225 <latexrelease> \csname mv@\math@version\expandafter\endcsname
226 <latexrelease> \expandafter{\number\csname
227 <latexrelease> c@mv@\math@version\endcsname}%
228 <latexrelease> #1%
229 <latexrelease> \global\advance\csname c@mv@\math@version\endcsname\@ne
230 <latexrelease> \else
231 <latexrelease> \let#1\relax
232 <latexrelease> \latex@error{Too many math alphabets used
233 <latexrelease> in version \math@version}%
234 <latexrelease> \@eha
235 <latexrelease> \fi
236 <latexrelease> \else \expandafter\non@alpherr\fi
237 <latexrelease> #1{#4}%
238 <latexrelease> }%
239 <latexrelease>}
240 <latexrelease>\EndIncludeInRelease
241 <latexrelease>
242 <latexrelease>\IncludeInRelease{0000/00/00}
243 <latexrelease> {\document@select@group}{\document@select@group}%
244 <latexrelease>
245 <latexrelease>\def\document@select@group#1#2#3#4{%
246 <latexrelease> \ifx\math@bgroup\bgroup\else\relax\expandafter\@firstofone\fi
247 <latexrelease> {%
248 <latexrelease> \ifmmode
249 <latexrelease> \ifnum\csname c@mv@\math@version\endcsname<\sixt@n
250 <latexrelease> \begingroup
251 <latexrelease> \escapechar\m@ne
252 <latexrelease> \getanddefine@fonts
253 <latexrelease> {\csname c@mv@\math@version\endcsname}#3%
254 <latexrelease> \globaldefs\@ne \math@fonts
255 <latexrelease> \endgroup
256 <latexrelease> \expandafter\extract@alph@from@version
257 <latexrelease> \csname mv@\math@version\expandafter\endcsname
258 <latexrelease> \expandafter{\number\csname
259 <latexrelease> c@mv@\math@version\endcsname}%
260 <latexrelease> #1%

```



```

261 <latexrelease> \global\advance\cscname c@mv@\math@version\endcscname\@ne
262 <latexrelease> \else
263 <latexrelease> \let#1\relax
264 <latexrelease> \@latex@error{Too many math alphabets used
265 <latexrelease> in version \math@version}%
266 <latexrelease> \@eha
267 <latexrelease> \fi
268 <latexrelease> \else \expandafter\non@alpherr\fi
269 <latexrelease> #1{#4}%
270 <latexrelease> }%
271 <latexrelease>}
272 <latexrelease>\EndIncludeInRelease
273 <*2ekernel>

```

(End of definition for \document@select@group.)

`\freeze@math@version` This command stores the current state of the math version and sets things up to return to it after each formula from now on. We use L3 programming layer code to set it up.

```

274 </2ekernel>
275 <*2ekernel | latexrelease>
276 <latexrelease>\IncludeInRelease{2022/11/01}%
277 <latexrelease> \freeze@math@version}{freeze math version}%
278 \ExplSyntaxOn
279 \cs_new_protected:Npn\freeze@math@version #1 {

```

Save the current `\mv@<version>` code and the number of allocated mathgroups inside.

```

280 \font@info{Freeze~ math~ alphabet~ allocation~ in~ version~
281 #1.\MessageBreak
282 Allocated~math~groups:~\int_use:c{ c@mv@ #1 }~
283 (local:~ \int_use:N\c@localmathalphabets) }
284 \cs_gset_eq:cc { mv@#1@frozen }{ mv@#1 }
285 \tl_gset:cx { g__nfss_frozen_mv_ #1 _tl }{ \int_use:c { c@mv@#1 } }

```

Here is the definition of `\mv@<version>@reset`. If there has been no new math alphabet allocation, doing a reset would just cause a lot of unnecessary processing, so we do a quick check upfront for this.

```

286 \cs_gset:cpn{mv@#1@reset}
287 {

```

If we are back at top-level, or more precisely outside of any (nested) math, we may have to reset the math version back to its frozen version, if that exists and has fewer alphabets allocated.

```

288 \int_compare:nNnTF \@math@level = 0
289 {
290 \int_compare:nNnTF { \int_use:c{c@mv@#1} } >
291 { \tl_use:c{g__nfss_frozen_mv_ #1 _tl} }
292 {
293 \font@info{ Undo~ math~ alphabet~ allocation~ in~ version~ #1 }

```

If the undo is necessary, we restore the `\mv@<version>` code.

```

294 \cs_gset_eq:cc { mv@#1 }{ mv@#1@frozen }
295 \int_gset:cn { c@mv@#1 }{ \tl_use:c {g__nfss_frozen_mv_ #1 _tl} }

```

But we also should undo changes to the top-level math alphabet definitions. We therefore run this code with a modified definition for `\getanddefine@fonts` because there is no need to do anything to the symbol fonts that are permanently allocated. However, we do this only if the resetting was for the current math version, because otherwise we would give the top-level definitions the values for the last math version being resetted (e.g., bold if there are just two).

```

296         \tl_if_eq:NnTF \math@version {#1}
297         {
298             \group_begin:
299             \cs_set_eq:NN \getanddefine@fonts \use_none:nn
300             \use:c {mv@ \math@version }
301             \group_end:
302         }
303     {

```

Once we have hit the boundary and have to revert to the frozen version that happens quite often, but typically only for the normal version. Thus, it is better to report only when we do not revert the top-level definition as this is the exceptional case and happens only if several math versions are used in parallel.

```

304         \font@info{ ...~ but~ do~ not~ reset~ the~
305             top-level~ math~ alphabet~ definitions }
306     }
307 }
308 {

```

If there was no change, we report that in the log (but this branch could go completely).

```

309         \font@info{ No~ math~ alphabet~ change~
310             to~ frozen~ version~ #1 }
311     }
312 }
313 {
314     \font@info{ Nested~ math:~ keeping~ math~ alphabet~
315         allocation~ in~ version~ #1 }
316 }

```

If this is executed after a math display, we may have to arrange for ignoring spaces, because they are now hidden if the tokens from above intervene. This is signaled by the 2e switch `@ignore` which is set in `\frozen@everymath` and `\frozen@everydisplay`.

This is all 2e code so we use that syntax.

```

317     \if@ignore \ignorespaces \fi
318 }
319 }
320 \ExplSyntaxOff
321 </2ekernel | latexrelease>
322 <latexrelease>\EndIncludeInRelease
323 <latexrelease>\IncludeInRelease{2021/11/15}
324 <latexrelease>         {\freeze@math@version}{freeze math version}%
325 <latexrelease>
326 <latexrelease>\ExplSyntaxOn
327 <latexrelease>\cs_set_protected:Npn\freeze@math@version #1 {
328 <latexrelease> \font@info{Freeze~ math~ alphabet~ allocation~ in~ version~
329 <latexrelease>         #1.\MessageBreak
330 <latexrelease>         Allocated-math-groups:~\int_use:c{ c@mv@ #1 }~
331 <latexrelease>         (local:~ \int_use:N\c@localmathalphabets)      }

```

```

332 <latexrelease> \cs_gset_eq:cc { mv@#1@frozen }{ mv@#1 }
333 <latexrelease> \tl_gset:cx { g__nfss_frozen_mv_ #1 _tl }{ \int_use:c { c@mv@#1 } }
334 <latexrelease> \group_insert_after:N \__nfss_init_mv_freeze:N
335 <latexrelease> \exp_after:wN \group_insert_after:N \cs:w mv@#1@reset \cs_end:
336 <latexrelease> \tl_gput_right:No \check@mathfonts
337 <latexrelease> {
338 <latexrelease> \exp_after:wN \group_insert_after:N \cs:w mv@#1@reset \cs_end:
339 <latexrelease> }
340 <latexrelease> \cs_gset:cpn{mv@#1@reset}
341 <latexrelease> {
342 <latexrelease> \int_compare:nNnTF { \int_use:c{c@mv@#1} } >
343 <latexrelease> { \tl_use:c{g__nfss_frozen_mv_ #1 _tl} }
344 <latexrelease> {
345 <latexrelease> \@font@info{Undo~ math~ alphabet~ allocation~ in~ version~ #1}
346 <latexrelease> \cs_gset_eq:cc { mv@#1 }{ mv@#1@frozen }
347 <latexrelease> \int_gset:cn { c@mv@#1 }{ \tl_use:c {g__nfss_frozen_mv_ #1 _tl} }
348 <latexrelease> \group_begin:
349 <latexrelease> \cs_set_eq:NN \getanddefine@fonts \use_none:n
350 <latexrelease> \use:c {mv@#1}
351 <latexrelease> \group_end:
352 <latexrelease> }
353 <latexrelease> {
354 <latexrelease> \@font@info{No~ math~ alphabet~ change~ to~ frozen~ version~ #1}
355 <latexrelease> }
356 <latexrelease> \if@ignore \ignorespaces \fi
357 <latexrelease> }
358 <latexrelease>}
359 <latexrelease>\cs_set_protected:Npn \__nfss_init_mv_freeze:N #1 {%
360 <latexrelease> \mode_if_math:T { \group_insert_after:N \__nfss_init_mv_freeze:N
361 <latexrelease> \group_insert_after:N } #1
362 <latexrelease>}
363 <latexrelease>\ExplSyntaxOff
364 <latexrelease>
365 <latexrelease>\EndIncludeInRelease
366 <*2ekernel>

```

(End of definition for \freeze@math@version.)

\process@table

```

367 \def\process@table{%
368 \def\cdp@elt##1##2##3##4{%
369 \@font@info{Checking defaults for
370 ##1/##2/##3/##4}%
371 \expandafter
372 \ifx\csname##1/##2/##3/##4\endcsname\relax

```

Grouping is important for two reasons, first \cdp@elt will get redefined if \Declare... functions are executed within the external .fd file and secondly \try@load@fontshape changes a lot of catcodes without surrounding itself with a group.

```

373 \begingroup
374 \def\f@encoding{##1}\def\f@family{##2}%
375 \try@load@fontshape
376 \endgroup
377 \fi
378 \expandafter

```

```

379     \ifx\csname##1/##2/##3/##4\endcsname\relax
380         \@latex@error{This NFSS system isn't set up properly}%
381             {For encoding scheme ##1 the defaults
382               ##2/##3/##4 do not form a valid font shape}%
383     \else
384         \@font@info{... okay}%
385     \fi}%
386 \cdp@list

```

Now we make sure that `\error@fontshape` is okay.

```

387 \begingroup
388   \escapechar\m@ne
389   \error@fontshape
390   \expandafter\ifx\csname \curr@fontshape\endcsname\relax
391     \begingroup
392       \try@load@fontshape
393     \endgroup
394   \fi
395   \expandafter\ifx\csname \curr@fontshape\endcsname\relax
396     \@latex@error{This NFSS system isn't set up properly}%
397       {The system maintainer forgot to specify a suitable
398         substitution
399         font shape using the \noexpand\DeclareErrorFont
400         command}%
401   \fi
402 \endgroup

```

Set `\select@group` to its meaning used within the document body.

```

403 \let\select@group\document@select@group

```

Install the default font attributes as they are currently pointing to error font face. We can speed up the process by just using `\edef`, thereby avoiding all kind of extra processing. Don't use `\reset@font` since that would trigger `\selectfont`.

```

404 \fontencoding\encodingdefault
405 \edef\f@family{\familydefault}%
406 \edef\f@series{\seriesdefault}%
407 \edef\f@shape{\shapedefault}%

```

Drop stuff not longer needed. We need to add many more!!!!

```

408 \everyjob{}%
409 }
410 \@onlypreamble\process@table

```

(End of definition for \process@table.)

```

411 %\@onlypreamble\set@mathradical

```

`\DeclareMathVersion`

```

412 </2kernel>
413 <*2kernel | latexrelease>
414 <latexrelease>\IncludeInRelease{2022/11/01}%
415 <latexrelease>           {\DeclareMathVersion}{local alphabets}%
416 \def\DeclareMathVersion#1{%

```

When declaring a new math version we need to instantiate an L3 variable that is used when we freeze the version, because too many alphabets got allocated. If we don't do this, L3 programming layer complains if it is run in checking mode.

```
417 \namedef{g_nfss_frozen_mv_#1_tl}{}%
```

We also extend `\check@mathfonts` to call a version reset (once frozen) after a formula has finished.

```
418 \expandafter\ifx\csname mv@#1\endcsname \relax
419 \expandafter \g@addto@macro \expandafter \check@mathfonts
420 \expandafter {\expandafter \aftergroup \csname mv@#1@reset\endcsname}
```

Initially this macro does nothing. It is, however, important that it doesn't stop any `\ignorespaces`, so we make it expandable and not `\relax`.

```
421 \namedef{mv@#1@reset}{}%
422 \fi

423 \expandafter\new@mathversion\csname mv@#1\endcsname}
424 \@onlypreamble\DeclareMathVersion
425 </2ekernel|latexrelease>
426 <latexrelease>\EndIncludeInRelease

427 <latexrelease>\IncludeInRelease{2021/11/15}%
428 <latexrelease> \DeclareMathVersion{local alphabets}%
429 <latexrelease>\def\DeclareMathVersion#1{%
430 <latexrelease> \namedef{g_nfss_frozen_mv_#1_tl}{}%
431 <latexrelease> \expandafter\new@mathversion\csname mv@#1\endcsname}
432 <latexrelease>\EndIncludeInRelease
433 <latexrelease>\IncludeInRelease{0000/00/00}%
434 <latexrelease> \DeclareMathVersion{local alphabets}%
435 <latexrelease>\def\DeclareMathVersion#1{%
436 <latexrelease> \expandafter\new@mathversion\csname mv@#1\endcsname}
437 <latexrelease>\EndIncludeInRelease
438 <*2ekernel>
```

(End of definition for \DeclareMathVersion.)

`\new@mathversion`

```
439 \def\new@mathversion#1{%
440 \expandafter\in@\expandafter#1\expandafter{\version@list}%
441 \ifin@
442 \font@info{Redeclaring math version
443 \expandafter\@gobblefour\string#1'}%
444 \else
445 \expandafter\newcount\csname c@\expandafter
446 \@gobble\string#1\endcsname
447 \def\version@elt{\noexpand\version@elt\noexpand}%
448 \edef\version@list{\version@list\version@elt#1}%
449 \fi
```

`\toks@` is used to gather all tokens for the math version. `\count@` will be used to count the math groups we add to this version.

```
450 \toks@{}%
451 \count@\z@
```

Now we loop over `\group@list` to add all math groups defined so far to the version and at the same time to count them.

```

452 \def\group@elt##1##2{%
453     \advance\count@\@ne
454     \addto@hook\toks@\getanddefine@fonts##1##2}%
455     }%
456 \group@list

```

We set the counter for this math version to the number of math groups found in `\group@list`.

```

457 \global\csname c@\expandafter\@gobble\string#1\endcsname\count@

```

Now we loop over `\alpha@list` to add all math alphabets known so far. We have to distinguish the case that an alphabet by default should produce an error in new versions.

```

458 \def\alpha@elt##1##2##3{%
459     \ifx##2\no@alphabet@error
460     \toks@\expandafter{\the\toks@\install@mathalphabet##1%
461         {\no@alphabet@error##1}}%
462     \else
463     \toks@\expandafter{\the\toks@\install@mathalphabet##1%
464         {\select@group##1##2##3}}%
465     \fi
466     }%
467 \alpha@list

```

Finally we define the math version to expand to the contents of `\toks@`.

```

468 \xdef#1{\the\toks@}%
469 }
470 \@onlypreamble\new@mathversion

```

(End of definition for \new@mathversion.)

`\DeclareSymbolFont` First drop any surplus `m` from the series argument then do what has been done since 1994.

```

471 </2ekernel>
472 <*2ekernel | latexrelease>
473 <latexrelease>\IncludeInRelease{2022/11/01}%
474 <latexrelease>           {\DeclareSymbolFont}{maybe drop m}%
475 \def\DeclareSymbolFont #1#2#3#4#5{%
476     \def\reserved@a{\DeclareSymbolFont@m@dropped{#1}{#2}{#3}}%
477     \edef\reserved@b{#4}%
478     \series@maybe@drop@one@m\reserved@b\reserved@b
479     \expandafter\reserved@a\expandafter{\reserved@b}{#5}%
480 }

481 \def\DeclareSymbolFont@m@dropped #1#2#3#4#5{%
482     \@tempswafalse
483     \def\reserved@b{#2}%
484     \def\cdp@elt##1##2##3##4{\def\reserved@c{##1}%
485         \ifx\reserved@b\reserved@c \@tempswatrue\fi}%
486     \cdp@list
487     \if@tempswa
488         \@ifundefined{sym#1}{%

```

```

489     \ifnum\count18<15 %
490         \expandafter\new@mathgroup\csname sym#1\endcsname
491         \expandafter\new@symbolfont\csname sym#1\endcsname
492             {#2}{#3}{#4}{#5}%
493     \else
494         \latex@error{Too many symbol fonts declared}\@eha
495     \fi
496 }%
497 {%
498     \@font@info{Redefining symbol font ‘#1’}%

```

Update the group list.

```

499     \def\group@elt##1##2{%
500         \noexpand\group@elt\noexpand##1%
501         \expandafter\ifx\csname sym#1\endcsname##1%
502             \expandafter\noexpand\csname#2/#3/#4/#5\endcsname
503         \else
504             \noexpand##2%
505         \fi}%
506     \xdef\group@list{\group@list}%

```

Update the version list.

```

507     \def\version@elt##1{%
508         \expandafter
509         \SetSymbolFont\expandafter##1\csname#2/#3/#4/#5\endcsname
510             \csname sym#1\endcsname
511         }%
512     \version@list
513 }%
514 \else
515     \latex@error{Encoding scheme ‘#2’ unknown}\@eha
516 \fi
517 }
518 \@onlypreamble\DeclareSymbolFont
519 </2ekernel | latexrelease>
520 <latexrelease>\EndIncludeInRelease
521 <latexrelease>\IncludeInRelease{0000/00/00}%
522 <latexrelease>         {\DeclareSymbolFont}{maybe drop m}%
523 <latexrelease>
524 <latexrelease>\let\DeclareSymbolFont\DeclareSymbolFont@m@dropped
525 <latexrelease>\let\DeclareSymbolFont@m@dropped\@undefined
526 <latexrelease>
527 <latexrelease>\EndIncludeInRelease
528 <*2ekernel>

```

(End of definition for \DeclareSymbolFont.)

\group@list

```

529 \let\group@list\@empty
530 \@onlypreamble\group@list

```

(End of definition for \group@list.)

\group@elt

```

531 \let\group@elt\relax
532 \@onlypreamble\group@elt

```

(End of definition for \group@elt.)

\new@symbolfont

```
533 \def\new@symbolfont#1#2#3#4#5{%
534   \toks@\expandafter{\group@list}%
535   \edef\group@list{\the\toks@\noexpand\group@elt\noexpand#1%
536     \expandafter\noexpand\csize#2/#3/#4/#5\endcsname}%
537   \def\version@elt##1{\toks@\expandafter{##1}%
538     \edef##1{\the\toks@\noexpand\getanddefine@font#
539       #1\expandafter\noexpand\csize#2/#3/#4/#5\endcsname}%
540     \global\advance\csize c@\expandafter
541       \gobble\string##1\endcsname\@ne
542     }%
543   \version@list
544 }
545 \@onlypreamble\new@symbolfont
```

(End of definition for \new@symbolfont.)

\SetSymbolFont First drop any surplus m from the series argument then do what has been done since 1994.

```
546 </2ekernel>
547 <*2ekernel | latexrelease>
548 <latexrelease> \IncludeInRelease{2022/11/01}%
549 <latexrelease>   {\SetSymbolFont}{maybe drop m}%
550 \def\SetSymbolFont #1#2#3#4#5#6{%
551   \def\reserved@a{\SetSymbolFont@m@dropped{#1}{#2}{#3}{#4}}%
552   \edef\reserved@b{#5}%
553   \series@maybe@drop@one@m\reserved@b\reserved@b
554   \expandafter\reserved@a\expandafter{\reserved@b}{#6}%
555 }

556 \def\SetSymbolFont@m@dropped#1#2#3#4#5#6{%
557   \@tempswafalse
558   \edef\reserved@b{#3}%
559   \def\cdp@elt##1##2##3##4{\def\reserved@c{##1}%
560     \ifx\reserved@b\reserved@c \@tempswatrue\fi}%
561   \cdp@list
562   \if@tempswa
563     \expandafter\SetSymbolFont@
564       \csize mv@#2\expandafter\endcsname\csize#3/#4/#5/#6\expandafter
565       \endcsname \csize sym#1\endcsname
566   \else
567     \@latex@error{Encoding scheme ‘#3’ unknown}\@eha
568   \fi
569 }
570 \@onlypreamble\SetSymbolFont
571 </2ekernel | latexrelease>
572 <latexrelease>\EndIncludeInRelease
573 <latexrelease>\IncludeInRelease{0000/00/00}%
574 <latexrelease>   {\SetSymbolFont}{maybe drop m}%
575 <latexrelease>
576 <latexrelease>\let\SetSymbolFont\SetSymbolFont@m@dropped
577 <latexrelease>\let\SetSymbolFont@m@dropped\@undefined
578 <latexrelease>
```



```

579 \latexrelease)\EndIncludeInRelease
580 (*2ekernel)

```

(End of definition for \SetSymbolFont.)

\SetSymbolFont@

```

581 \def\SetSymbolFont@#1#2#3{%
582   \expandafter\in@\expandafter#1\expandafter{\version@list}%
583   \ifin@
584     \expandafter\in@\expandafter#3\expandafter{\group@list}%
585     \ifin@
586       \begingroup
587         \expandafter\get@cdp\string#2\@nil\reserved@a
588         \toks@{}%
589         \def\install@mathalphabet##1##2{%
590           \addto@hook\toks@{\install@mathalphabet##1{##2}}%
591           }%
592         \def\getanddefine@fonts##1##2{%
593           \ifnum##1=#3%
594             \addto@hook\toks@{\getanddefine@fonts#3#2}%
595             \expandafter\get@cdp\string##2\@nil\reserved@b
596             \ifx\reserved@a\reserved@b\else
597               \@font@info{Encoding '\reserved@b' has changed
598                 to '\reserved@a' for symbol font\MessageBreak
599                 '\expandafter\@gobblefour\string#3' in the
600                 math version '\expandafter
601                 \@gobblefour\string#1'}%
602             \fi
603             \@font@info{%
604               Overwriting symbol font
605               '\expandafter\@gobblefour\string#3' in
606               version '\expandafter
607               \@gobblefour\string#1'\MessageBreak
608               \@spaces \expandafter\@gobble\string##2 -->
609                 \expandafter\@gobble\string#2}%
610             \else
611               \addto@hook\toks@{\getanddefine@fonts##1##2}%
612             \fi}%
613         #1%
614         \xdef#1{\the\toks@}%
615       \endgroup
616     \else
617       \@latex@error{Symbol font '\expandafter\@gobblefour\string#3'
618         not defined}\@eha
619     \fi
620   \else
621     \@latex@error{Math version '\expandafter\@gobblefour\string#1'
622       is not
623       defined}{You probably misspelled the name of the math
624       version.^^JOr you have to specify an additional package.}%
625   \fi
626 }
627 \@onlypreamble\SetSymbolFont@

```

(End of definition for \SetSymbolFont@.)

`\get@cdp`

```
628 \def\get@cdp#1#2/#3\@nil#4{\def#4{#2}}
629 \@onlypreamble\get@cdp
```

(End of definition for \get@cdp.)

`\DeclareMathAlphabet`

```
630 \def\DeclareMathAlphabet#1#2#3#4#5{%
631   \@tempswafalse
632   \edef\reserved@b{#2}%
633   \def\cdp@elt##1##2##3##4{\def\reserved@c{##1}%
634     \ifx\reserved@b\reserved@c \@tempswatruetrue\fi}%
635   \cdp@list
636   \if@tempswa
637     \expandafter\ifx
638     \csname\expandafter\@gobble\string#1\endcsname
639     \relax
640     \new@mathalphabet#1{#2}{#3}{#4}{#5}%
641   \else
```

Check if it is already a math alphabet.

```
642   \edef\reserved@a{\noexpand\in@{\string\select@group}%
643     {\expandafter\meaning\csname \expandafter
644       \@gobble\string#1\space\endcsname}}%
645   \reserved@a
646   \ifin@
647     \@font@info{Redeclaring math alphabet \string#1}%
648     \def\version@elt##1{%
649       \expandafter\SetMathAlphabet@\expandafter
650         ##1\csname#2/#3/#4/#5\expandafter\endcsname
651
652       \csname M@#2\expandafter\endcsname
653       \csname \expandafter\@gobble\string#1\space\endcsname#1}%
654     \version@list
655   \else
```

Check if it is a math alphabet defined via `\DeclareSymbolFontAlphabet`.

```
655   \edef\reserved@a{\noexpand\in@{\string\use@mathgroup}%
656     {\expandafter\meaning\csname \expandafter
657       \@gobble\string#1\space\endcsname}}%
658   \reserved@a
659   \ifin@
```

In that case overwriting is simple since there is nothing inserted in the math version macros.

```
660     \@font@info{Redeclaring math alphabet \string#1}%
661     \new@mathalphabet#1{#2}{#3}{#4}{#5}%
```

Otherwise panic.

```
662   \else
663     \@latex@error{Command ‘\string#1’ already defined}\@eha
664   \fi
665 \fi
666 \fi
667 \else
668   \@latex@error{Encoding scheme ‘#2’ unknown}\@eha
```

```

669 \fi
670 }
671 \@onlypreamble\DeclareMathAlphabet

```

(End of definition for \DeclareMathAlphabet.)

\new@mathalphabet

```

672 \def\new@mathalphabet#1#2#3#4#5{%
673   \toks@\expandafter{\alpha@list}%
674   \edef#1{\expandafter\noexpand\csname \expandafter
675     \@gobble\string#1\space\endcsname
676     \if/#5/%
677       \noexpand\no@alphabet@error
678       \noexpand\no@alphabet@error
679     \else
680       \expandafter\noexpand\csname M@#2\endcsname
681       \expandafter\noexpand\csname#2/#3/#4/#5\endcsname
682     \fi
683   }%
684   \toks2\expandafter{#1}%
685   \edef\alpha@list{\the\toks@\noexpand\alpha@elt\the\toks2}%
686   \def\version@elt##1{\toks@\expandafter{##1}%
687     \edef##1{\the\toks@\install@mathalphabet
688       \expandafter\noexpand
689       \csname \expandafter\@gobble
690         \string#1\space\endcsname
691       {\if/#5/%
692         \noexpand\no@alphabet@error
693         \noexpand#1%
694       \else
695         \noexpand\select@group\the\toks2
696         \fi}}%
697     }%
698   \version@list
699   \expandafter\edef\csname \expandafter\@gobble
700     \string#1\space\endcsname{\if/#5/%
701     \noexpand\no@alphabet@error
702     \noexpand#1%
703   \else
704     \noexpand\select@group\the\toks2
705   \fi}%
706   \edef#1{\noexpand\protect
707     \expandafter\noexpand\csname \expandafter
708     \@gobble\string#1\space\endcsname}%
709 }
710 \@onlypreamble\new@mathalphabet

```

(End of definition for \new@mathalphabet.)

\SetMathAlphabet

```

711 \def\SetMathAlphabet#1#2#3#4#5#6{%
712   \@tempswafalse
713   \edef\reserved@b{#3}%
714   \def\cdp@elt##1##2##3##4{\def\reserved@c{##1}%
715     \ifx\reserved@b\reserved@c \@tempswatruetrue\fi}%

```

```

716 \cdp@list
717 \if@tempswa
718 \expandafter\SetMathAlphabet@
719 \csname mv@#2\expandafter\endcsname\csname#3/#4/#5/#6\expandafter
720 \endcsname \csname M@#3\expandafter\endcsname
721 \csname \expandafter@gobble\string#1\space\endcsname#1%
722 \else
723 \@latex@error{Encoding scheme ‘#3’ unknown}\@eha
724 \fi
725 }
726 \@onlypreamble\SetMathAlphabet

```

(End of definition for \SetMathAlphabet.)

\SetMathAlphabet@

```

727 \def\SetMathAlphabet@#1#2#3#4#5{%
728 \expandafter\in@\expandafter#1\expandafter{\version@list}%
729 \ifin@
730 \expandafter\in@\expandafter#4\expandafter{\alpha@list}%
731 \ifin@
732 \begingroup
733 \toks@{%
734 \def\getanddefine@fonts##1##2{%
735 \addto@hook\toks@{\getanddefine@fonts##1##2}%
736 }%
737 \def\reserved@c##1##2##3##4{% % for message below
738 \expandafter@gobble\string##4}%
739 \def\install@mathalphabet##1##2{%
740 \ifx##1#4%
741 \addto@hook\toks@
742 {\install@mathalphabet#4{\select@group#4#3#2}}%
743 \@font@info{Overwriting math alphabet
744 ‘\string#5’ in version ‘\expandafter
745 \@gobblefour\string#1’\MessageBreak
746 \@spaces \reserved@c##2 -->
747 \expandafter@gobble\string#2}%
748 \else
749 \addto@hook\toks@{\install@mathalphabet##1{##2}}%
750 \fi
751 }%
752 #1%
753 \xdef#1{\the\toks@}%
754 \endgroup
755 \else

```

If the math alphabet was defined via \DeclareSymbolFontAlphabet we have remove its external definition and add it as a normal math alphabet to every version before trying to change it in one version.

```

756 \edef\reserved@a{%
757 \noexpand\in@{\string\use@mathgroup}{\meaning#4}}%
758 \reserved@a
759 \ifin@
760 \def\reserved@b##1\use@mathgroup##2##3{%
761 \def\reserved@b{##3}\def\reserved@c{##2}}%
762 \expandafter\reserved@b#4%

```

```

763     \begingroup
764     \def\install@mathalphabet##1##2{%
765         \addto@hook\toks@{\install@mathalphabet##1{##2}}%
766         }%
767     \def\getanddefine@fonts##1##2{%
768         \addto@hook\toks@{\getanddefine@fonts##1##2}%
769         \ifnum##1=\reserved@b
770             \expandafter
771             \addto@hook\expandafter\toks@
772             \expandafter{\expandafter\install@mathalphabet
773             \expandafter#4\expandafter
774             {\expandafter\select@group\expandafter
775             #4\reserved@c##2}}%
776         \fi
777     }%
778     \def\version@elt##1{%
779         \toks@{}%
780         ##1%
781         \xdef##1{\the\toks@}%
782     }%
783     \version@list
784     \endgroup

```

Put it into the `\alpha@list` with default ‘error’

```

785     \expandafter\gdef\expandafter\alpha@list\expandafter
786     {\alpha@list
787     \alpha@elt #4\no@alphabet@error \no@alphabet@error}%
788     \gdef#4{\no@alphabet@error #5}% fake things :-)

```

Then call the internal setting routine again:

```

789     \SetMathAlphabet@{#1}{#2}{#3}#4#5%
790     \else
791     \@latex@error{Command ‘\string#5’ not defined as a
792     math alphabet}%
793     {Use \noexpand\DeclareMathAlphabet to define it.}%
794     \fi
795     \fi
796     \else
797     \@latex@error{Math version ‘\expandafter@gobblefour\string#1’
798     is not
799     defined}{You probably misspelled the name of the math
800     version.^^JOr you have to specify an additional package.}%
801     \fi
802     }
803     \onlypreamble\SetMathAlphabet@

```

(End of definition for \SetMathAlphabet@.)

`\DeclareMathAccent` Could do with more checks like allowing single number in #4 lowercase in #4 etc

```

804     </2ekernel>
805     <*2ekernel | latexrelease>
806     <latexrelease>\IncludeInRelease{2019/10/01}%
807     <latexrelease>           {DeclareMathAccent}{Make math accents robust}%
808     \def\DeclareMathAccent#1#2#3#4{%
809     \expandafter\in@\csname sym#3\expandafter\endcsname

```

```

810     \expandafter{\group@list}%
811 \ifin@
812   \begingroup
813     \count\z@=#4\relax
814     \count\tw@\count\z@
815     \divide\count\z@\sist@@n
816     \count@\count\z@
817     \multiply\count@\sist@@n
818     \advance\count\tw@-\count@
819     \if\relax\noexpand#1% is command?
820     \edef\reserved@a{\noexpand\in@
821       {\expandafter@gobble\string\mathaccent}
822       {\expandafter\meaning
823         \csname\expandafter@gobble\string#1\space\endcsname}}%
824     \reserved@a
825     \ifin@
826       \expandafter\let
827         \csname\expandafter@gobble\string#1\space\endcsname
828         \@undefined
829       \expandafter\set@mathaccent
830         \csname sym#3\endcsname#1#2%
831         {\hexnumber@\count\z@}\hexnumber@\count\tw@}}%
832       \@font@info{Redefining math accent \string#1}%
833     \else
834       \expandafter\ifx
835         \csname\expandafter@gobble\string#1\endcsname
836         \relax
837         \expandafter\set@mathaccent
838           \csname sym#3\endcsname#1#2%
839           {\hexnumber@\count\z@}\hexnumber@\count\tw@}}%
840       \else
841         \@latex@error{Command '\string#1' already defined}\@eha
842       \fi
843     \fi
844     \else
845       \@latex@error{Not a command name: '\noexpand#1'}\@eha
846     \fi
847   \endgroup
848 \else
849   \@latex@error{Symbol font '#3' is not defined}\@eha
850 \fi
851 }
852 </2ekernel | latexrelease>
853 <latexrelease>\EndIncludeInRelease
854 <latexrelease>\IncludeInRelease{0000/00/00}%
855 <latexrelease>      {DeclareMathAccent}{Make math accents robust}%
856 <latexrelease>\def\DeclareMathAccent#1#2#3#4{%
857 <latexrelease>  \expandafter\in@\csname sym#3\expandafter\endcsname
858 <latexrelease>    \expandafter{\group@list}%
859 <latexrelease>  \ifin@
860 <latexrelease>    \begingroup
861 <latexrelease>      \count\z@=#4\relax
862 <latexrelease>      \count\tw@\count\z@
863 <latexrelease>      \divide\count\z@\sist@@n

```

```

864 <latexrelease> \count@\count\z@
865 <latexrelease> \multiply\count@\sixt@@n
866 <latexrelease> \advance\count\tw@-\count@
867 <latexrelease> \if\relax\noexpand#1% is command?
868 <latexrelease> \edef\reserved@a{\noexpand\in@
869 <latexrelease> \expandafter\@gobble\string\mathaccent}{\meaning#1}}%
870 <latexrelease> \reserved@a
871 <latexrelease> \ifin@
872 <latexrelease> \expandafter\set@mathaccent
873 <latexrelease> \csname sym#3\endcsname#1#2%
874 <latexrelease> {\hexnumber@\count\z@}\hexnumber@\count\tw@}}%
875 <latexrelease> \@font@info{Redefining math accent \string#1}%
876 <latexrelease> \else
877 <latexrelease> \expandafter\ifx
878 <latexrelease> \csname\expandafter\@gobble\string#1\endcsname
879 <latexrelease> \relax
880 <latexrelease> \expandafter\set@mathaccent
881 <latexrelease> \csname sym#3\endcsname#1#2%
882 <latexrelease> {\hexnumber@\count\z@}\hexnumber@\count\tw@}}%
883 <latexrelease> \else
884 <latexrelease> \@latex@error{Command '\string#1' already defined}\@eha
885 <latexrelease> \fi
886 <latexrelease> \fi
887 <latexrelease> \else
888 <latexrelease> \@latex@error{Not a command name: '\noexpand#1'}\@eha
889 <latexrelease> \fi
890 <latexrelease> \endgroup
891 <latexrelease> \else
892 <latexrelease> \@latex@error{Symbol font '#3' is not defined}\@eha
893 <latexrelease> \fi
894 <latexrelease>}
895 <latexrelease>\EndIncludeInRelease
896 <*2ekernel)

```

```

897 \@onlypreamble\DeclareMathAccent

```

(End of definition for \DeclareMathAccent.)

\set@mathaccent

```

898 </2ekernel)
899 <*2ekernel | latexrelease)
900 <latexrelease> \IncludeInRelease{2019/10/01}%
901 <latexrelease> {\set@mathaccent}{makemath accents robust}%
902 \def\set@mathaccent#1#2#3#4{%
903 \xdef#2{\mathaccent"\mathchar@type#3\hexnumber@#1#4\relax}%
904 \MakeRobust#2%
905 }
906 \@onlypreamble\set@mathaccent
907 </2ekernel | latexrelease)
908 <latexrelease>\EndIncludeInRelease
909 <latexrelease>\IncludeInRelease{0000/00/00}%
910 <latexrelease> {\set@mathaccent}{makemath accents robust}%
911 <latexrelease>
912 <latexrelease>\def\set@mathaccent#1#2#3#4{%
913 <latexrelease> \xdef#2{\mathaccent"\mathchar@type#3\hexnumber@#1#4\relax}}

```

```

914 <latexrelease>
915 <latexrelease>\EndIncludeInRelease
916 <*2kernel>

```

(End of definition for \set@mathaccent.)

\DeclareMathSymbol

```

917 \def\DeclareMathSymbol#1#2#3#4{%
918   \expandafter\in@\csname sym#3\expandafter\endcsname
919   \expandafter{\group@list}%
920   \ifin@
921     \begingroup
922       \count\z@=#4\relax
923       \count\tw@\count\z@
924       \divide\count\z@\sist@@n
925       \count@\count\z@
926       \multiply\count@\sist@@n
927       \advance\count\tw@-\count@
928       \if\relax\noexpand#1% is command?

```

Store the command name with a space attached inside \reserved@@b in case we look at a robust definition.

```

929       \edef\reserved@b{\expandafter\noexpand
930         \csname\expandafter@gobble\string#1\space\endcsname}%

```

Test both #1 and #1_□ for containing mathchar.

```

931       \edef\reserved@a
932         {\noexpand\in@{\expandafter@gobble\string\mathchar}%
933          {\meaning#1\expandafter\meaning\reserved@b}}%
934       \reserved@a

```

Drop #1_□ in case it was defined before.

```

935       \global\expandafter\let\reserved@b\@undefined
936       \ifin@
937         \expandafter\set@mathsymbol
938           \csname sym#3\endcsname#1#2%
939           {\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
940         \font@info{Redeclaring math symbol \string#1}%
941       \else
942         \expandafter\ifx
943           \csname\expandafter@gobble\string#1\endcsname
944           \relax
945         \expandafter\set@mathsymbol
946           \csname sym#3\endcsname#1#2%
947           {\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
948         \else
949           \@latex@error{Command '\string#1' already defined}\@eha
950         \fi
951       \fi
952     \else
953       \expandafter\set@mathchar
954         \csname sym#3\endcsname#1#2
955         {\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
956       \fi
957     \endgroup

```



```

958 \else
959 \latex@error{Symbol font ‘#3’ is not defined}\@eha
960 \fi
961 }
962 \@onlypreamble\DeclareMathSymbol

```

(End of definition for \DeclareMathSymbol.)

`\set@mathchar`

```

963 \def\set@mathchar#1#2#3#4{%
964 \global\mathcode‘#2="\mathchar@type#3\hexnumber@#1#4\relax}
965 \@onlypreamble\set@mathchar

```

(End of definition for \set@mathchar.)

`\set@mathsymbol`

```

966 \def\set@mathsymbol#1#2#3#4{%
967 \global\mathchardef#2"\mathchar@type#3\hexnumber@#1#4\relax}
968 \@onlypreamble\set@mathsymbol

```

(End of definition for \set@mathsymbol.)

```

969 %\def\mathsymbol#1#2#3{%
970 % \tempcnta=#3\relax
971 % \tempcntb\tempcnta
972 % \divide\tempcnta\sixt@n
973 % \count@\tempcnta
974 % \multiply\count@\sixt@n
975 % \advance\tempcntb-\count@
976 % \mathchar"\mathchar@type#1\hexnumber@#2%
977 % \hexnumber@\tempcnta\hexnumber@\tempcntb\relax}
978 %
979 %\def\DeclareMathAlphabetCharacter#1#2#3{%
980 % \DeclareMathSymbol{#1}7{#2}{#3}

```

`\DeclareMathDelimiter`

```

981 \def\DeclareMathDelimiter#1{%
982 \if\relax\noexpand#1%
983 \expandafter\DeclareMathDelimiter
984 \else
985 \expandafter\@xxDeclareMathDelimiter
986 \fi
987 #1}
988 \@onlypreamble\DeclareMathDelimiter

```

(End of definition for \DeclareMathDelimiter.)

`\@xxDeclareMathDelimiter`

This macro checks if the second arg is a “math type” such as `\mathopen`. The undocumented original code didn’t use math types when the delimiter was a single letter. For this reason the coding is a bit strange as it tries to support the undocumented syntax for compatibility reasons.

```

989 \def\@xxDeclareMathDelimiter#1#2#3#4{%

```

7 is the default value returned in the case that `\mathchar@type` is passed something unexpected, like a math symbol font name. We locally move `\mathalpha` out of the way so if you use that the right branch is taken. This will still fail if an explicit number 7 is used!

```

990 \begingroup
991 \let\mathalpha\mathord
992 \ifnum7=\mathchar@type{#2}%
993 \endgroup

```

If this branch is taken we have old syntax (5 arguments).

```

994 \expandafter\@firstofone
995 \else

```

If this branch is taken `\mathchar@type` is different from 7 so we assume new syntax. In this case we also use the arguments to set up the letter as a math symbol for the case where it is not used as a delimiter.

```

996 \endgroup
997 \DeclareMathSymbol#1{#2}{#3}{#4}%

```

Then we arrange that `\@xxDeclareMathDelimiter` only gets #1, #3, #4 ... as it does not expect a math type as argument.

```

998 \expandafter\@firstoftwo
999 \fi
1000 {\@xxDeclareMathDelimiter#1}{#2}{#3}{#4}}
1001 \@onlypreamble\@xxDeclareMathDelimiter

```

(End of definition for \@xxDeclareMathDelimiter.)

`\@DeclareMathDelimiter`

```

1002 \def\@DeclareMathDelimiter#1#2#3#4#5#6{%
1003 \expandafter\in@\csname sym#3\expandafter\endcsname
1004 \expandafter{\group@list}%
1005 \ifin@
1006 \expandafter\in@\csname sym#5\expandafter\endcsname
1007 \expandafter{\group@list}%
1008 \ifin@
1009 \begingroup
1010 \count\z@=#4\relax
1011 \count\tw@\count\z@
1012 \divide\count\z@\sist@@n
1013 \count@\count\z@
1014 \multiply\count@\sist@@n
1015 \advance\count\tw@-\count@
1016 \edef\reserved@c{\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
1017 %
1018 \count\z@=#6\relax
1019 \count\tw@\count\z@
1020 \divide\count\z@\sist@@n
1021 \count@\count\z@
1022 \multiply\count@\sist@@n
1023 \advance\count\tw@-\count@
1024 \edef\reserved@d{\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
1025 %
1026 \edef\reserved@a{\noexpand\in@
1027 {\expandafter\@gobble\string\delimiter}{\meaning#1}}%

```

```

1028     \reserved@a
1029     \ifin@
1030         \expandafter\set@mathdelimiter
1031             \csname sym#3\expandafter\endcsname
1032             \csname sym#5\endcsname#1#2%
1033             \reserved@c\reserved@d
1034             \@font@info{Redeclaring math delimiter \string#1}%
1035     \else
1036         \expandafter\ifx
1037             \csname\expandafter\@gobble\string#1\endcsname
1038             \relax
1039             \expandafter\set@mathdelimiter
1040                 \csname sym#3\expandafter\endcsname
1041                 \csname sym#5\endcsname#1#2%
1042                 \reserved@c\reserved@d
1043     \else
1044         \@latex@error{Command ‘\string#1’ already defined}\@eha
1045     \fi
1046 \fi
1047 \endgroup
1048 \else
1049     \@latex@error{Symbol font ‘#5’ is not defined}\@eha
1050 \fi
1051 \else
1052     \@latex@error{Symbol font ‘#3’ is not defined}\@eha
1053 \fi
1054 }
1055 \@onlypreamble\@DeclareMathDelimiter

```

(End of definition for \@DeclareMathDelimiter.)

\@xDeclareMathDelimiter

```

1056 \def\@xDeclareMathDelimiter#1#2#3#4#5{%
1057     \expandafter\in@\csname sym#2\expandafter\endcsname
1058     \expandafter{\group@list}%
1059     \ifin@
1060         \expandafter\in@\csname sym#4\expandafter\endcsname
1061         \expandafter{\group@list}%
1062     \ifin@
1063         \begingroup
1064             \count\z@=#3\relax
1065             \count\tw@\count\z@
1066             \divide\count\z@\sixt@n
1067             \count@\count\z@
1068             \multiply\count@\sixt@n
1069             \advance\count\tw@-\count@
1070             \edef\reserved@c{\hexnumber@\count\z@}\hexnumber@\count\tw@}%
1071         %
1072         \count\z@=#5\relax
1073         \count\tw@\count\z@
1074         \divide\count\z@\sixt@n
1075         \count@\count\z@
1076         \multiply\count@\sixt@n
1077         \advance\count\tw@-\count@

```

```

1078     \edef\reserved@d{\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
1079     \expandafter\set@mathdelimiter
1080     \csname sym#2\expandafter\endcsname\csname sym#4\endcsname#1%
1081     \reserved@c\reserved@d
1082   \endgroup
1083   \else
1084     \@latex@error{Symbol font ‘#4’ is not defined}\@eha
1085   \fi
1086   \else
1087     \@latex@error{Symbol font ‘#2’ is not defined}\@eha
1088   \fi
1089 }
1090 \@onlypreamble\@xDeclareMathDelimiter

```

(End of definition for \@xDeclareMathDelimiter.)

`\set@mathdelimiter` We have to end the definition of a math delimiter like `\lfloor` with a space and not with `\relax` as we did before, because otherwise constructs involving `\abovewithdelims` will prematurely end (pr/1329)

```

1091 </2kernel>
1092 <*2kernel | latexrelease>
1093 <latexrelease>\IncludeInRelease{2019/10/01}%
1094 <latexrelease>           {\set@mathdelimiter}{make delimiters robust}%
1095 \def\set@mathdelimiter#1#2#3#4#5#6{%

```

We use `\protected` not `\MakeRobust` so that `\bigl\lfloor` etc. works inside the argument of `\protected@edef`.

```

1096   \protected
1097   \xdef#3{\delimiter"\mathchar@type#4\hexnumber@#1#5%
1098     \hexnumber@#2#6 }%
1099   % \MakeRobust#3%
1100 }
1101 \@onlypreamble\set@mathdelimiter
1102 </2kernel | latexrelease>
1103 <latexrelease>\EndIncludeInRelease
1104 <latexrelease>\IncludeInRelease{0000/00/00}%
1105 <latexrelease>           {\set@mathdelimiter}{make delimiters robust}%
1106 <latexrelease>
1107 <latexrelease>\def\set@mathdelimiter#1#2#3#4#5#6{%
1108 <latexrelease> \xdef#3{\delimiter"\mathchar@type#4\hexnumber@#1#5%
1109 <latexrelease>   \hexnumber@#2#6 }}
1110 <latexrelease>
1111 <latexrelease>\EndIncludeInRelease
1112 <*2kernel>

```

(End of definition for \set@mathdelimiter.)

`\set@@mathdelimiter`

```

1113 \def\set@@mathdelimiter#1#2#3#4#5{%
1114   \global\delcode‘#3’=\hexnumber@#1#4\hexnumber@#2#5\relax}
1115 \@onlypreamble\set@@mathdelimiter

```

(End of definition for \set@@mathdelimiter.)

`\DeclareMathRadical`

```
1116 \def\DeclareMathRadical#1#2#3#4#5{%
Below is a crude fix to make this macro work if #1 is undefined or \relax. Should be
improved!
1117 \expandafter\ifx
1118 \csname\expandafter@gobble\string#1\endcsname
1119 \relax
1120 \let#1\radical
1121 \fi
1122 \edef\reserved@a{\noexpand\in@
1123 {\expandafter@gobble\string\radical}{\meaning#1}}%
\reserved@a
1124 \ifin@
1125 \ifin@
1126 \expandafter\in@\csname sym#2\expandafter\endcsname
1127 \expandafter{\group@list}%
1128 \ifin@
1129 \expandafter\in@\csname sym#4\expandafter\endcsname
1130 \expandafter{\group@list}%
1131 \ifin@
1132 \begingroup
1133 \count\z@=#3\relax
1134 \count\tw@\count\z@
1135 \divide\count\z@\sixt@@n
1136 \count@\count\z@
1137 \multiply\count@\sixt@@n
1138 \advance\count\tw@-\count@
1139 \edef\reserved@c{%
1140 \hexnumber@\count\z@}\hexnumber@\count\tw@}}%
1141 \count\z@=#5\relax
1142 \count\tw@\count\z@
1143 \divide\count\z@\sixt@@n
1144 \count@\count\z@
1145 \multiply\count@\sixt@@n
1146 \advance\count\tw@-\count@
1147 \edef\reserved@d{%
1148 \hexnumber@\count\z@}\hexnumber@\count\tw@}}%
Coded inline instead of using \set@mathradical
1149 % \expandafter\set@mathradical
1150 % \csname sym#2\expandafter\endcsname
1151 % \csname sym#4\endcsname#1%
1152 % \reserved@c\reserved@d
1153 \xdef#1{\radical"\expandafter\hexnumber@
1154 \csname sym#2\endcsname\reserved@c
1155 \expandafter\hexnumber@
1156 \csname sym#4\endcsname\reserved@d
1157 \relax}%
1158 \endgroup
1159 \else
1160 \@latex@error{Symbol font '#4' is not defined}\@eha
1161 \fi
1162 \else
1163 \@latex@error{Symbol font '#2' is not defined}\@eha
1164 \fi
```

```

1165 \else
1166 \latexerror{Command '\string#1' already defined}\@eha
1167 \fi
1168 }
1169 \@onlypreamble\DeclareMathRadical

```

(End of definition for \DeclareMathRadical.)

Definition below was wrong it contained \delimiter !

```

def\set@mathradical#1#2#3#4#5{%
\xdef#3{\radical"\hexnumber@#1#4\hexnumber@#2#5\relax}}

```

`\mathalpha` just a dummy currently

```

1170 \let\mathalpha\relax

```

(End of definition for \mathalpha.)

`\mathchar@type`

```

1171 \def\mathchar@type#1{%
1172 \ifodd 2#1#1 #1\else % is this non-negative number?
1173 \ifx#1\mathord 0\else
1174 \ifx#1\mathop 1\else
1175 \ifx#1\mathbin 2\else
1176 \ifx#1\mathrel 3\else
1177 \ifx#1\mathopen 4\else
1178 \ifx#1\mathclose 5\else
1179 \ifx#1\mathpunct 6\else
1180 7% % anything else is variable ord
1181 \fi
1182 \fi
1183 \fi
1184 \fi
1185 \fi
1186 \fi
1187 \fi
1188 \fi}
1189 \@onlypreamble\mathchar@type

```

(End of definition for \mathchar@type.)

`\DeclareSymbolFontAlphabet`

```

1190 \def\DeclareSymbolFontAlphabet#1#2{%
1191 \expandafter\DeclareSymbolFontAlphabet@
1192 \csname \expandafter@gobble\string#1\space\endcsname{#2}#1}
1193 \@onlypreamble\DeclareSymbolFontAlphabet

```

(End of definition for \DeclareSymbolFontAlphabet.)

`\DeclareSymbolFontAlphabet@`

```

1194 \def\DeclareSymbolFontAlphabet@#1#2#3{%
We use the switch \if@tempswa to decide if we can declare this symbol font alphabet.
1195 \@tempswatrue

```

First check if #2 is known to be a symbol font

```
1196 \expandafter\in@\csname sym#2\expandafter\endcsname
1197 \expandafter{\group@list}%
1198 \ifin@
```

Check if #1 is defined as a math alphabet defined via `\DeclareMathAlphabet:`

```
1199 \expandafter\in@\expandafter#1\expandafter{\alpha@list}%
1200 \ifin@
```

If so remove it from the `\alpha@list` and from all math version macros.

```
1201 \@font@info{Redefining math alphabet \string#3}%
1202 \toks@{}%
1203 \def\alpha@elt##1##2##3{%
1204 \ifx##1#1\else\addto@hook\toks@{\alpha@elt##1##2##3}\fi}%
1205 \alpha@list
1206 \xdef\alpha@list{\the\toks@}%
```

Now we loop over all versions and remove the math alphabet:

```
1207 \def\version@elt##1{%
1208 \begingroup
1209 \toks@{}%
1210 \def\getanddefine@fonts####1####2{%
1211 \addto@hook\toks@{\getanddefine@fonts####1####2}}%
1212 \def\install@mathalphabet####1####2{%
1213 \ifx####1#1\else
1214 \addto@hook\toks@{\install@mathalphabet
1215 ####1{####2}}\fi}%
1216 ##1%
1217 \xdef##1{\the\toks@}%
1218 \endgroup
1219 }%
1220 \version@list
1221 \else
```

If #3 is not defined as a math alphabet check if it is defined at all:

```
1222 \expandafter\ifx
1223 \csname\expandafter\@gobble\string#1\space\endcsname
1224 \relax
```

If it is undefined, fine otherwise check if it is a math alphabet defined via `\DeclareSymbolFontAlphabet:`

```
1225 \else
1226 \edef\reserved@a{%
1227 \noexpand\in@{\string\use@mathgroup}{\meaning#1}}%
1228 \reserved@a
1229 \ifin@
1230 \@font@info{Redefining math alphabet \string#3}%
1231 \else
```

Since the command #3 is defined to be something which is not a math alphabet we have to skip redefining it.

```
1232 \@tempwafalse
1233 \@latex@error{Command ‘\string#3’ already defined}\@eha
1234 \fi
1235 \fi
1236 \fi
1237 \else
```

Since the symbol font is not known we better skip defining this alphabet.

```
1238     \@tempswafalse
1239     \@latex@error{Unknown symbol font '#2'}\@eha
1240     \fi
1241     \if@tempswa
```

When we reach this point we are allowed to define #1 to be a symbol font math alphabet. This means that we have to set it to

```
\use@mathgroup <math-settings> \sym<name>
```

The <math-settings> are the one for the encoding that is used in the font shape where \sym<name> is pointing to. This means that we have to get it from the information stored in \group@list. Thus we loop through that list after defining \group@elt in a suitable way.

```
1242     \def\group@elt##1##2{%
1243         \expandafter\ifx\csname sym#2\endcsname##1%
1244         \expandafter\reserved@a\string##2\@nil
1245         \fi}%
1246     \def\reserved@a##1##2/##3\@nil{%
1247         \def\reserved@a{##2}}%
1248     \group@list
1249     \toks@{\relax\ifmmode \else \non@alpherr#1\fi}%
1250     \edef#1{\the\toks@
1251         \noexpand\use@mathgroup
1252         \expandafter\noexpand\csname M@\reserved@a\endcsname
1253         \csname sym#2\endcsname}%
1254     \def#3{\protect#1}%
1255     \fi
1256 }
1257 \@onlypreamble\DeclareSymbolFontAlphabet@
1258 </2ekernel)
```

(End of definition for \DeclareSymbolFontAlphabet@.)

File 29

ltfssini.dtx

This file contains the top level L^AT_EX interface to the font selection scheme commands. See other parts of the L^AT_EX distribution, or *The L^AT_EX Companion* for higher level documentation of these commands.

1 NFSS Initialization

Finally, there are six commands that are to be used in L^AT_EX and that we will therefore protect against expansion at the wrong point: `\fontfamily`, `\fontseries`, `\fontshape`, `\fontsize`, `\selectfont`, and `\mathversion`.

```
1 <*2kernel>
2 \message{NFSS initialization,}
```

1.1 Providing math *versions*

L^AT_EX provides two *versions*. We call them normal and bold, respectively.

```
3 \DeclareMathVersion{normal}
4 \DeclareMathVersion{bold}
```

Now we define the standard font change commands. We don't allow the use of `\rmfamily` etc. in math mode.

(Actually most are now defined further down in the file.)

First the changes to another *family*:

```
5 %\DeclareRobustCommand\rmfamily
6 %      {\not@math@alphabet\rmfamily\mathrm
7 %      \fontfamily\rmdefault\selectfont}
8 %\DeclareRobustCommand\sffamily
9 %      {\not@math@alphabet\sffamily\mathsf
10 %      \fontfamily\sfdefault\selectfont}
11 %\DeclareRobustCommand\ttfamily
12 %      {\not@math@alphabet\ttfamily\mathtt
13 %      \fontfamily\ttdefault\selectfont}
```

Then the commands changing the *series*:

```
14 %\DeclareRobustCommand\bfseries
15 %      {\not@math@alphabet\bfseries\mathbf
16 %      \fontseries\bfdefault\selectfont}
17 %\DeclareRobustCommand\mdseries
18 %      {\not@math@alphabet\mdseries\relax
19 %      \fontseries\mddefault\selectfont}
20 \DeclareRobustCommand\upshape
21      {\not@math@alphabet\upshape\relax
22      \fontshape\updefault\selectfont}
```

Then the commands changing the *shape*:

```
23 \DeclareRobustCommand\slshape
24      {\not@math@alphabet\slshape\relax
25      \fontshape\sldefault\selectfont}
26 \DeclareRobustCommand\scshape
```

```

27         {\not@math@alphabet\scshape\relax
28          \fontshape\scdefault\selectfont}
29 \DeclareRobustCommand\itshape
30         {\not@math@alphabet\itshape\mathit
31          \fontshape\itdefault\selectfont}

```

2 Custom series settings for main document families

This section was introduced 2020/02/02 and for now we support a full rollback (may need splitting later).

```

32 </2ekernel>
33 <*2ekernel | latexrelease>
34 <latexrelease>\IncludeInRelease{2021/11/15}%
35 <latexrelease>          {\DeclareFontSeriesDefault}{Custom series}%

```

One problem with the NFSS approach of handling the series axis turned out to be that (especially with respect to “boldness”) different font families implemented different strategies. For example, with Computer Modern fonts you normally only have **bx** whereas most PostScript fonts offered only **b** but not **bx**. As a result L^AT_EX’s standard setting for `\bfdefault` didn’t work with such fonts, but if it got changed to produce **b**, then that didn’t work with Computer Modern if the fonts got combined (e.g., using Computer Modern Typewriter with such fonts).

The solution back then was to provide substitution rules in the font `.fd` such that if a **bx** series got requested the **b** series got used. While this works in that particular case, it isn’t a very general solution. For example, if you happen to have a font family that has several weights you may want to typeset the whole document in a somewhat lighter or darker font but if you then modify `\mddefault` to allow for this, then of course your change only works with that particular family but not with the typewriter or sans serif family you also want to use.

A better solution was provided by the `mweights` package by Bob Tennent that offers defaults on the level of the three main font families in the document: for “`rm`”, “`sf`” and “`tt`” so that font packages could define defaults for the sans serif document font by providing `\bfseries@sf` which then was used when `\bfseries` got executed and the current family was the `\sffamily`.

`\DeclareFontSeriesDefault` We now support this concept directly from within L^AT_EX and for use in font packages (or the document preamble) we offer `\DeclareFontSeriesDefault`. This declaration takes three arguments:

document family interface: Can either be `rm`, `sf` or `tt`. This is optional and if not given the overall default.

document series interface: Can be `md` or `bf`.

series value: This is the value that is going to be used with the combination is requested.

For example, `\DeclareFontSeriesDefault[rm]{bf}{sb}` would use **sb** (semi-bold) when `\rmfamily \bfseries` is asked for.

If used without the optional argument, e.g., `\DeclareFontSeriesDefault{bf}{b}` then this is like redefining `\bfdefault` or `\mddefault`.

If some family specify defaults aren't given, e.g. if there are no declarations for, say, `tt` then the format defaults of `\mddefault` and `\bfdefault` are assumed. If those are later changed this is *not* reflected!³⁷

`\DeclareFontSeriesDefault` The command to declare font series defaults for the “rm”, “sf” or “tt” family.

```

36 \let\DeclareFontSeriesDefault\undefined      % for rollback
37 \newcommand\DeclareFontSeriesDefault[3] []{%
38   \expand@font@defaults
39   \maybe@update@bfseries@defaults
40   \maybe@update@mdseries@defaults
41   \def\reserved@a{#1}%

```

No optional argument: set up general default.

```

42   \ifx\reserved@a\@empty
43     \ifcsname #2series\endcsname              % supported are
44                                           % \[md/bf]default

```

Adding `\@empty` allows us to detect if the default gets redefined with `\renewcommand` or `\def` by the user.

```

45     \expandafter\def
46       \csname #2default\endcsname{#3\@empty}%
47     \expandafter\def
48       \csname #2default@previous\endcsname{#3\@empty}%
49   \else
50     \@latex@error{Wrong syntax for \string\DeclareFontSeriesDefault}%
51     {Mandatory first argument must be 'md' or 'bf'.}
52   \fi

```

Optional argument given, set up specific default.

```

53   \else
54     \ifcsname #2series@#1\endcsname          % supported are
55                                           % \[md/bf]series@[rm/sf/tt]
56     \expandafter\edef
57       \csname #2series@#1\endcsname{#3}%

```

If the interface is used we remove the frozen kernel default. This way, we know that something was explicitly set up (even if the setup has the same value as the default).

```

58     \expandafter\let
59       \csname #2series@#1@kernel\endcsname\@undefined
60   \else
61     \@latex@error{Wrong syntax for \string\DeclareFontSeriesDefault}%
62     {Optional argument must be 'rm', 'sf', or 'tt'. \MessageBreak
63     Mandatory first argument must be 'md' or 'bf'.}
64   \fi
65 \fi
66 }

```

³⁷I see no easy way to achieve this without compromising compatibility with existing packages that currently use `mweights` and directly define (some) of the `\mdseries@..` commands but not others.

(End of definition for `\DeclareFontSeriesDefault`.)

```
67 </2ekernel | latexrelease>
68 <latexrelease>\EndIncludeInRelease
69 <latexrelease>\IncludeInRelease{2020/02/02}%
70 <latexrelease>          {\DeclareFontSeriesDefault}{Custom series}%
71 <latexrelease>
72 <latexrelease>\let\DeclareFontSeriesDefault\@undefined      % for rollback
73 <latexrelease>\newcommand\DeclareFontSeriesDefault[3] [] {%
74 <latexrelease>  \def\reserved@a{#1}%
75 <latexrelease>  \ifx\reserved@a\@empty
76 <latexrelease>    \ifcsname #2series\endcsname          % supported are
77 <latexrelease>                                     % \[md/bf]default
78 <latexrelease>    \expandafter\def
79 <latexrelease>      \csname #2default\endcsname{#3\@empty}%
80 <latexrelease>    \expandafter\def
81 <latexrelease>      \csname #2default@previous\endcsname{#3\@empty}%
82 <latexrelease>  \else
83 <latexrelease>    \@latex@error{Wrong syntax for \string\DeclareFontSeriesDefault}%
84 <latexrelease>      {Mandatory first argument must be 'md' or 'bf'.}
85 <latexrelease>  \fi
86 <latexrelease> \else
87 <latexrelease>  \ifcsname #2series@#1\endcsname          % supported are
88 <latexrelease>                                     % \[md/bf]series@[rm/sf/tt]
89 <latexrelease>    \expandafter\edef
90 <latexrelease>      \csname #2series@#1\endcsname{#3}%
91 <latexrelease>    \expandafter\let
92 <latexrelease>      \csname #2series@#1@kernel\endcsname\@undefined
93 <latexrelease>  \else
94 <latexrelease>    \@latex@error{Wrong syntax for \string\DeclareFontSeriesDefault}%
95 <latexrelease>      {Optional argument must be 'rm', 'sf', or 'tt'. \MessageBreak
96 <latexrelease>      Mandatory first argument must be 'md' or 'bf'.}
97 <latexrelease>  \fi
98 <latexrelease> \fi
99 <latexrelease>}
100 <latexrelease>
101 <latexrelease>\EndIncludeInRelease
102 <latexrelease>\IncludeInRelease{0000/00/00}%
103 <latexrelease>          {\DeclareFontSeriesDefault}{Custom series}%
104 <latexrelease>
105 <latexrelease>\let\DeclareFontSeriesDefault\@undefined
106 <latexrelease>\let\bfseries@rm\@undefined
107 <latexrelease>\let\bfseries@sf\@undefined
108 <latexrelease>\let\bfseries@tt\@undefined
109 <latexrelease>\let\bfseries@rm@kernel\@undefined
110 <latexrelease>\let\bfseries@sf@kernel\@undefined
111 <latexrelease>\let\bfseries@tt@kernel\@undefined
112 <latexrelease>\let\mdseries@rm\@undefined
113 <latexrelease>\let\mdseries@sf\@undefined
114 <latexrelease>\let\mdseries@tt\@undefined
115 <latexrelease>\expandafter\let\csname ver@weights.sty\endcsname\@undefined
116 <latexrelease>
117 <latexrelease>\let\@meta@family@list\@undefined
118 <latexrelease>\let\prepare@family@series@update\@undefined
119 <latexrelease>\let@update@series@target@value\@undefined
```

120 <latexrelease>

This is always called in `\document` so don't make it undefined.

121 <latexrelease>\let\init@series@setup\relax

122 <latexrelease>

123 <latexrelease>\EndIncludeInRelease

124 <*2ekernel>

125 </2ekernel>

126 <*2ekernel | latexrelease>

127 <latexrelease>\IncludeInRelease{2020/02/02}%

128 <latexrelease> \mdseries@rm}{Custom series}%

`\mdseries@rm` We initialize the family specific default at the end of the format generation. Later on
`\mdseries@sf` they may get overwritten in the preamble or a package via `\DeclareFontSeriesDefault`
`\mdseries@tt` (or possibly directly).

`\bfseries@rm` Conceptual change: The `\bfdefault` will be `b` not `bx` because that is what it should
`\bfseries@sf` be really for nearly every font except Computer/Latin Modern.

`\bfseries@tt` To account for the fact that by default we typeset in CM or LM we set up the
`\bfseries@..` defaults to use `bx` instead.

This means that it behaves like before because if the default fonts are used then
`\bfseries@rm` etc kick in and make `\textbf` use `bx`. However, if the font gets changed
then `\bfdefault` will get used.

129 \def\bfseries@rm{bx}

130 \def\bfseries@sf{bx}

131 \def\bfseries@tt{bx}

Frozen version of the kernel defaults so we can see if they have changed.

132 \let\bfseries@rm@kernel\bfseries@rm

133 \let\bfseries@sf@kernel\bfseries@sf

134 \let\bfseries@tt@kernel\bfseries@tt

The default for the medium series is `m` and this will be interpreted as resetting both
weight and width. To reset only one of them the virtual value `?m` and `m?` are available.

135 \def\mdseries@rm{m}

136 \def\mdseries@sf{m}

137 \def\mdseries@tt{m}

(End of definition for `\mdseries@rm` and others.)

`\series@change@debug` For debugging, but right now none of this code is extracted. The idea is to have a separate
package with debugging code one day.

138 <*debug>

139 \let\series@change@debug\typeout

140 \let\series@change@debug@gobble

141 </debug>

(End of definition for `\series@change@debug`.)

`\prepare@family@series@update` This is core command that prepares for the family update. The big difference to the
documented code above is that the nested `\ifx` statements seem to be missing. Instead
we loop through an internal list that holds the names of the three meta families. This
approach allows us to extend the mechanism at a later stage to allow for additional named
meta families.

Here is the current definition of that list:

```

\@meta@family@list 142 \def\@meta@family@list{\@elt{rm}\@elt{sf}\@elt{tt}}
143 \def\prepare@family@series@update#1#2{%
144   \if@forced@series
145   <+debug> \series@change@debug{No series preparation (forced \f@series)\on@line}%
146     \fontfamily#2%
147   \else
148   <+debug> \series@change@debug{Preparing for switching to #1 (#2)\on@line}%
149     \expand@font@defaults

```

We prepare for changing the current series. We have to find it before changing the family as discussed above.

```

150   \let\target@series@value\@empty
151   \def\target@meta@family@value{#1}%

```

As the very last item in the meta family list we add \@elt{??} and define this pseudo meta family to be the current font family. So if none of the real meta families matched then this will match. This will cover the following case:

- \bfseries is called for a family using `bx` (e.g., `CMR`)
- Switch to a font family that is none of the meta families, e.g., via `\fontfamily{ptm}\allowbreak\vf`
- Then none of the real meta families, match but the final \@elt{??} will.
- Therefore if the current series is `\mddefault` or `\bfdefault` it will be detected and the corresponding target series selected.

```

152   \expandafter\edef\csname ??def@ult\endcsname{\f@family}%

```

To find it we loop over the meta family list with a suitable definition of \@elt.

```

153   \let\@elt@update@series@target@value
154     \@meta@family@list

```

Last resort pseudo meta family. Will only be looked at if none of the real ones have matched.

```

155     \@elt{??}%
156     \let\@elt\relax

```

That will figure out the correct series value to use without updating it. Now we can change the family.

```

157   \fontfamily#2%

```

After that we update the series. That code is again like the one above.

```

158   \ifx\target@series@value\@empty
159   <+debug> \series@change@debug{Target series still empty ...}%
160   \else
161     \ifx \f@series\target@series@value
162   <+debug> \series@change@debug{Target series unchanged:
163   <+debug>                                     \f@series \space = \target@series@value}%
164     \else
165     \maybe@load@fontshape
166   <+debug> \series@change@debug{Target series:
167   <+debug>                                     \f@series \space -> \target@series@value}%

```

The `\target@series@value` may contain something like `cm` (coming from a default) and so we can't directly assign it to `\f@series` we have to drop any surplus `m` first.

```

168 %      \let\f@series\target@series@value
169      \series@maybe@drop@one@m\target@series@value\f@series
170      \fi
171      \fi
172 \fi
173 }

```

(End of definition for `\prepare@family@series@update` and `\@meta@family@list`.)

`\update@series@target@value`

In this macro used in the look you basically find the nested `\ifx`s from the outline above. The only difference is that it is parameterized instead of being written out and only for one block of tests because the code is called repeatedly when looping over the meta family list. From the list we get each meta family name in turn.

```

174 \def\update@series@target@value#1{%

```

There is one additional test at the beginning, because the list contains all meta families and we need to ignore the case where current one from the list and target one are identical.

```

175   \def\reserved@a{#1}%
176   \ifx\target@meta@family@value\reserved@a   % rm -> rm do nothing
177   \else
178   <+debug> \series@change@debug{Trying to match #1: \csname#1def@ult\endcsname
179   <+debug>           \space = \f@family\space ?}%

```

We only “do” something if the current font family matches the current meta family.

```

180   \expandafter\ifx\csname#1def@ult\endcsname\f@family

```

If that's the case we know that this is the block that applies (only one meta family can match). So to speed things up we change `\@elt` so that the rest of the loop gets gobbled.

```

181       \let\@elt@gobble

```

Then we try to find the right new value for the series (as explained above). The two macros defined first are only there because we now need to use `\csname` and this way the code will be a little faster.

```

182       \expandafter\let\expandafter\reserved@b
183           \csname mdseries@\target@meta@family@value\endcsname
184       \expandafter\let\expandafter\reserved@c
185           \csname bfseries@\target@meta@family@value\endcsname
186   <+debug>\series@change@debug{Targets for mdseries and bfseries:
187   <+debug>           \reserved@b\space and \reserved@c}%

```

This here is now identical to the nested `\ifx` block from the outline, except that it there appeared twice in `\rmfamily`. This is now covered by looping and stopping the loop when a match was found.

We have to sanitize the default value first because it may contain something like `mc` and that would never match `\f@series` because there it would be called `c` with the `m` dropped. It would be probably better to do that differently these days, but it is hard to adjust without causing a lot of issues, so we do the dropping in various places instead.

```

188       \expandafter\series@maybe@drop@one@m
189       \csname mdseries@#1\endcsname\reserved@d
190       \ifx\reserved@d\f@series
191   <+debug>   \series@change@debug{mdseries@#1 matched -> \reserved@b}%
192           \let\target@series@value\reserved@b
193       \else

```

Again do some sanitizing.

```

194     \expandafter\series@maybe@drop@one@m
195     \csname bfseries@#1\endcsname\reserved@d
196     \ifx\reserved@d\f@series
197 <+debug> \series@change@debug{bfseries@#1 matched -> \reserved@c}%
198     \let\target@series@value\reserved@c
199     \else\ifx\f@series\mdef@ult \let\target@series@value\reserved@b
200 <+debug> \series@change@debug{mdef@ult matched -> \reserved@b}%
201     \else\ifx\f@series\bdef@ult \let\target@series@value\reserved@c
202 <+debug> \series@change@debug{bdef@ult matched -> \reserved@c}%
203     \fi\fi\fi\fi
204     \fi
205     \fi
206 }

```

(End of definition for \update@series@target@value.)

\init@series@setup This is code to be run at begin document ...

```

207 \def\init@series@setup{%

```

We only want `bx` in `\bfseries@rm` if the roman font is Computer Modern or Latin Modern, otherwise it should be `b`. It was set to `bx` in the kernel so that any font use with the default families in the preamble get this value. Now at the real document start we check if the fonts have been changed. If there was a `\DeclareFontSeriesDefault` declaration or `\bfseries@rm` was directly altered then it differs from `\bfseries@rm@kernel` and we do nothing. Otherwise we check if `\rmdefault` is one of the CM/LM font families and if so we keep `bx` otherwise we change it to `b`.

This approach doesn't cover one case: CM/LM got changed to a different family that supports `bx`, but the support package for that family used `\def\bfseries@rm{bx}` instead of using `\DeclareFontSeriesDefault`. In that case the code here changes it to `b`. Solution: use the `\DeclareFontSeriesDefault` interface.

```

208 \ifx\bfseries@rm@kernel\bfseries@rm
209   \expandafter\in@\expandafter{\rmdefault}%
210   {cmr,cmss,cmtt,lcms,lcmtt,lmr,lmss,lmtt}%
211 \ifin@ \else \def\bfseries@rm{b}\fi\fi

```

Same approach for `\bfseries@sf` and `\bfseries@tt`:

```

212 \ifx\bfseries@sf@kernel\bfseries@sf
213   \expandafter\in@\expandafter{\sfdefault}%
214   {cmr,cmss,cmtt,lcms,lcmtt,lmr,lmss,lmtt}%
215 \ifin@ \else \def\bfseries@sf{b}\fi\fi
216 \ifx\bfseries@tt@kernel\bfseries@tt
217   \expandafter\in@\expandafter{\ttdefault}%
218   {cmr,cmss,cmtt,lcms,lcmtt,lmr,lmss,lmtt}%
219 \ifin@ \else \def\bfseries@tt{b}\fi\fi

```

If the document preamble has changed the `\familydefault` or if the if the `\rmdefault` contains a new font family, we may have to adjust the series defaults accordingly, before starting typesetting.

Similarly, if the user has changed the `\mdef@ult` or the medium series for the family selected as document font we may also have to adjust the `\seriesdefault`.

On the other hand if the document font is still CM or LM then `\bdef@ult` is wrong, because it is now saying `b` and not `bx` as it should for such fonts.

To fix all this we first run `\reset@font` (the internal kernel name for `\normalfont`). This will set up the document encoding, family, series and shape based on the current values of `\encodingdefault`, `\familydefault`, `\seriesdefault` and `\shapedefault`. However, if the family (from `\familydefault`) has special medium default we should switch to that (and not use what is current value from `\seriesdefault`). This can be achieved by afterwards calling `\mediumseries` and then changing `\seriesdefault` to the now current series value (in `\f@series`).

But what should happen if `\seriesdefault` got explicitly changed? In that case the explicit change should survive and we should not alter `\seriesdefault`. This is solved by comparing the current value of `\seriesdefault` with a kernel version saved in the format and if they differ we do not call `\mdseries` or change `\seriesdefault`.

```

220 \reset@font
221 \ifx\seriesdefault\seriesdefault@kernel
222 \mdseries
223 \let\seriesdefault\f@series
224 \fi
225 ]%
```

(End of definition for `\init@series@setup`.)

As the kernel code now implements the same functionality as `mweights`, albeit internally coded slightly differently, that package shouldn't be loaded any more. We therefore pretend that it already got loaded. Thus, a font package that tries to load it and then sets `\mdseries@.`, etc. will continue to work but will now use the kernel code.

Of course, mid-term such package should probably use `\DeclareFontSeriesDefault` instead of making using low-level definitions.

```

226 \expandafter\let\csname ver@mweights.sty\endcsname\fmtversion
227 </2kernel | latexrelease>
228 <latexrelease>\EndIncludeInRelease
229 <latexrelease>\IncludeInRelease{0000/00/00}%
230 <latexrelease> \mdseries@rm}{Custom series}%
231 <latexrelease>
232 <latexrelease>\let\bfseries@rm\@undefined
233 <latexrelease>\let\bfseries@sf\@undefined
234 <latexrelease>\let\bfseries@tt\@undefined
235 <latexrelease>\let\bfseries@rm@kernel\@undefined
236 <latexrelease>\let\bfseries@sf@kernel\@undefined
237 <latexrelease>\let\bfseries@tt@kernel\@undefined
238 <latexrelease>\let\mdseries@rm\@undefined
239 <latexrelease>\let\mdseries@sf\@undefined
240 <latexrelease>\let\mdseries@tt\@undefined
241 <latexrelease>\expandafter\let\csname ver@mweights.sty\endcsname\@undefined
242 <latexrelease>
243 <latexrelease>\let\@meta@family@list\@undefined
244 <latexrelease>\let\prepare@family@series@update\@undefined
245 <latexrelease>\let@update@series@target@value\@undefined
246 <latexrelease>
```

This is always called in `\document` so don't make it undefined.

```

247 <latexrelease>\let\init@series@setup\relax
248 <latexrelease>
249 <latexrelease>\EndIncludeInRelease
250 <*2kernel>
```

```

251 </2ekernel>
252 <*2ekernel | latexrelease>
253 <latexrelease>\IncludeInRelease{2021/11/15}%
254 <latexrelease>                {\bfseries}{Custom series with hooks}%

```

`\bfseries` This document command switches to the bold series.

```

255 \DeclareRobustCommand\bfseries{%
256   \not@math@alphabet\bfseries\mathbf

```

In the original NFSS definition it then called `\fontseries` with the value `\bfdefault`. In the new scheme we have more alternatives and therefore check if the current family (`\f@family`) is the current `\rmdef@ult`, `\sfdef@ult` or `\ttdef@ult` and then select the correct family default in that case.

```

257   \expand@font@defaults
258   \maybe@update@bfseries@defaults

259   \ifx\f@family\rmdef@ult      \fontseries\bfseries@rm
260   \else\ifx\f@family\sfdef@ult \fontseries\bfseries@sf
261   \else\ifx\f@family\ttdef@ult \fontseries\bfseries@tt

```

If not `\bfdefault` is used.

```

262   \else                        \fontseries\bfdefault
263   \fi\fi\fi

```

This hook in contrast is always executed.

```

264   \UseHook{bfseries}%
265   \selectfont
266 }

```

(End of definition for `\bfseries`.)

`\maybe@update@bfseries@defaults` If `\bfdefault` and `\bfdefault@previous` are different then the default got changed directly through the legacy interface (i.e., via `\def` or `\renewcommand`). In that case we reset all meta family defaults so that the document behaves like it was the case before the new mechanism was introduced.

```

267 \def\maybe@update@bfseries@defaults{%
268   \ifx\bfdefault\bfdefault@previous\else

```

We add `\@empty` and then let `\bfdefault@previous` to `\bfdefault` so that we can detect any further change.

```

269     \expandafter\def\expandafter\bfdefault
270         \expandafter{\bfdefault\@empty}%
271     \let\bfdefault@previous\bfdefault

```

And we reset the meta family defaults (`\bfdef@ult` is an expanded version of `\bfdefault`).

```

272     \let\bfseries@rm\bfdef@ult
273     \let\bfseries@sf\bfdef@ult
274     \let\bfseries@tt\bfdef@ult

```

Formats that set up parallel fonts, e.g., for Japanese, can use this hook to add resets here. Note that this hook is only run when resets are necessary.

```

275     \UseHook{bfseries/defaults}%
276     \fi
277 }

```

(End of definition for `\maybe@update@bfseries@defaults`.)

`\mdseries` This document command switches to the medium series.

```
278 \DeclareRobustCommand\mdseries{%
279   \not@math@alphabet\mdseries\relax
280   \expand@font@defaults
281   \maybe@update@mdseries@defaults
282   \ifx\f@family\rmdef@ult      \fontseries\mdseries@rm
283     \else\ifx\f@family\sfdef@ult \fontseries\mdseries@sf
284     \else\ifx\f@family\ttdef@ult \fontseries\mdseries@tt
285     \else                       \fontseries\mddefault
286   \fi\fi\fi
287   \UseHook{mdseries}%
288   \selectfont
289 }
```

(End of definition for \mdseries.)

`\maybe@update@mdseries@defaults`

```
290 \def\maybe@update@mdseries@defaults{%
291   \ifx\mddefault\mddefault@previous\else
292     \expandafter\def\expandafter\mddefault\expandafter{\mddefault\@empty}%
293     \let\mddefault@previous\mddefault
294     \let\mdseries@rm\mddef@ult
295     \let\mdseries@sf\mddef@ult
296     \let\mdseries@tt\mddef@ult
```

Formats that set up parallel fonts, e.g., for Japanese, can use this hook to add resets here.

```
297   \UseHook{mdseries/defaults}%
298   \fi
299 }
```

(End of definition for \maybe@update@mdseries@defaults.)

```
300 </2ekernel | latexrelease>
301 <latexrelease>\EndIncludeInRelease
302 <latexrelease>\IncludeInRelease{2020/10/01}%
303 <latexrelease>          {\bfseries}{Custom series with hooks}%
304 <latexrelease>
305 <latexrelease>\let\maybe@update@bfseries@defaults\@undefined
306 <latexrelease>\let\maybe@update@mdseries@defaults\@undefined
307 <latexrelease>
308 <latexrelease>\DeclareRobustCommand\bfseries{%
309 <latexrelease>  \not@math@alphabet\bfseries\mathbf
310 <latexrelease>  \expand@font@defaults
311 <latexrelease>  \ifx\bfdefault\bfdefault@previous\else
312 <latexrelease>    \expandafter\def\expandafter\bfdefault
313 <latexrelease>      \expandafter{\bfdefault\@empty}%
314 <latexrelease>    \let\bfdefault@previous\bfdefault
315 <latexrelease>    \let\bfseries@rm\bfdef@ult
316 <latexrelease>    \let\bfseries@sf\bfdef@ult
317 <latexrelease>    \let\bfseries@tt\bfdef@ult
318 <latexrelease>    \UseHook{bfseries/defaults}%
319 <latexrelease>  \fi
320 <latexrelease>    \ifx\f@family\rmdef@ult      \fontseries\bfseries@rm
321 <latexrelease>    \else\ifx\f@family\sfdef@ult \fontseries\bfseries@sf
```

```

322 <latexrelease> \else\ifx\f@family\ttdef@ult \fontseries\bfseries@tt
323 <latexrelease> \else \fontseries\bfdefault
324 <latexrelease> \fi\fi\fi
325 <latexrelease> \UseHook{bfseries}%
326 <latexrelease> \selectfont
327 <latexrelease>}
328 <latexrelease>
329 <latexrelease>\DeclareRobustCommand\mdseries{%
330 <latexrelease> \not@math@alphabet\mdseries\relax
331 <latexrelease> \expand@font@defaults
332 <latexrelease> \ifx\mddefault\mddefault@previous\else
333 <latexrelease> \expandafter\def\expandafter\mddefault\expandafter{\mddefault\@empty}%
334 <latexrelease> \let\mddefault@previous\mddefault
335 <latexrelease> \let\mdseries@rm\mddef@ult
336 <latexrelease> \let\mdseries@sf\mddef@ult
337 <latexrelease> \let\mdseries@tt\mddef@ult
338 <latexrelease> \UseHook{mdseries/defaults}%
339 <latexrelease> \fi
340 <latexrelease> \ifx\f@family\rmdef@ult \fontseries\mdseries@rm
341 <latexrelease> \else\ifx\f@family\sfdef@ult \fontseries\mdseries@sf
342 <latexrelease> \else\ifx\f@family\ttdef@ult \fontseries\mdseries@tt
343 <latexrelease> \else \fontseries\mddefault
344 <latexrelease> \fi\fi\fi
345 <latexrelease> \UseHook{mdseries}%
346 <latexrelease> \selectfont
347 <latexrelease>}
348 <latexrelease>\EndIncludeInRelease
349 <latexrelease>\IncludeInRelease{2020/02/02}%
350 <latexrelease> \bfseries}{Custom series with hooks}%
351 <latexrelease>
352 <latexrelease>
353 <latexrelease>\DeclareRobustCommand\bfseries{%
354 <latexrelease> \not@math@alphabet\bfseries\mathbf
355 <latexrelease> \expand@font@defaults
356 <latexrelease> \ifx\f@family\rmdef@ult \fontseries\bfseries@rm
357 <latexrelease> \else\ifx\f@family\sfdef@ult \fontseries\bfseries@sf
358 <latexrelease> \else\ifx\f@family\ttdef@ult \fontseries\bfseries@tt
359 <latexrelease> \else \fontseries\bfdefault
360 <latexrelease> \fi\fi\fi
361 <latexrelease> \selectfont
362 <latexrelease>}
363 <latexrelease>
364 <latexrelease>\DeclareRobustCommand\mdseries{%
365 <latexrelease> \not@math@alphabet\mdseries\relax
366 <latexrelease> \expand@font@defaults
367 <latexrelease> \ifx\f@family\rmdef@ult \fontseries\mdseries@rm
368 <latexrelease> \else\ifx\f@family\sfdef@ult \fontseries\mdseries@sf
369 <latexrelease> \else\ifx\f@family\ttdef@ult \fontseries\mdseries@tt
370 <latexrelease> \else \fontseries\mddefault
371 <latexrelease> \fi\fi\fi
372 <latexrelease> \selectfont
373 <latexrelease>}
374 <latexrelease>
375 <latexrelease>

```

```

376 <latexrelease>
377 <latexrelease>\EndIncludeInRelease
378 <latexrelease>\IncludeInRelease{0000/00/00}%
379 <latexrelease>                {\bfseries}{Custom series with hooks}%
380 <latexrelease>
381 <latexrelease>\DeclareRobustCommand\bfseries
382 <latexrelease>                {\not@math@alphabet\bfseries\mathbf}
383 <latexrelease>                \fontseries\bfdefault\selectfont}
384 <latexrelease>\DeclareRobustCommand\mdseries
385 <latexrelease>                {\not@math@alphabet\mdseries\relax
386 <latexrelease>                \fontseries\mddefault\selectfont}
387 <latexrelease>
388 <latexrelease>\EndIncludeInRelease
389 <*2ekernel>
390
391
392
393
394 </2ekernel>
395 <*2ekernel | latexrelease>
396 <latexrelease>\IncludeInRelease{2020/10/01}%
397 <latexrelease>                {\expand@font@defaults}{Custom series with hooks}%

```

`\expand@font@defaults` The family specific defaults are fully expanded, i.e., they are defined via `\edef` inside `\rm@def@ult` `\DeclareFontSeriesDefault`. However, the overall defaults, e.g., `\bfdefault` may have been redefined by the user and thus may not be fully expanded. So to enable reliable comparison we make expanded versions of them. That we rerun each time. The alternative would be to only allow for changes before begin document.

```

\rm@def@ult \def\expand@font@defaults{%
\sf@def@ult \edef\rmdef@ult{\rmdefault}%
\tt@def@ult \edef\sfdef@ult{\sfdefault}%
\md@def@ult \edef\ttdef@ult{\ttdefault}%
\bf@def@ult

```

The series defaults may contain some surplus `m` that we need to drop here.

```

402 \series@maybe@drop@one@m\bfdefault\bfdef@ult
403 \series@maybe@drop@one@m\mddefault\mddef@ult

```

Formats that set up parallel fonts, e.g., for Japanese, can use this hook to add additional code here.

```

404 \UseHook{expand@font@defaults}%
405 }

```

(End of definition for `\expand@font@defaults` and others.)

`\rmfamily` Here are the document level commands for changing the main font families, or rather, here is a documented outline of the code, the actual code is then streamlined and somewhat generalized.

```

DeclareRobustCommand\rmfamily{%
\not@math@alphabet\rmfamily\mathrm

```

If families are changed then we have to do a bit more work. In the original NFSS implementation a family change kept encoding, series shape and size unchanged but now we can't any longer simply reuse the current series value. Instead we may have to change it from one family default to the next.

```
\expand@font@defaults
```

We have to do the testing while the current family is still unchanged but we have to do the adjustment of the series after it got changed (because the new family might have different sets of shapes available and we certainly don't want to see substitution going on. So we use `\target@series@value` to hold the target series (if any).

```
\let\target@series@value\@empty
```

Thus, if the current family is the sans family

```
\ifx\f@family\sfdef@ult
```

and if we using the medium series of the sans family

```
\ifx\f@series\mdseries@sf
```

then lets switch to the medium series for the serif family

```
\let\target@series@value\mdseries@rm
```

and if we use the bold series of the sans family switch to the bold default of the serif family:

```
\else\ifx\f@series\bfseries@sf \let\target@series@value\bfseries@rm
```

However, the sans family may not have any specific defaults set, so we also compare with the overall defaults.

```
\else\ifx\f@series\mddef@ult \let\target@series@value\mdseries@rm
```

```
\else\ifx\f@series\bfdef@ult \let\target@series@value\bfseries@rm
```

If neither test was true we leave the series alone. This way a special manual setting such as `\fontseries{lc}` is not undone if the family changes (of course there may not be any support for it in the new family but then the NFSS substitution kicks in and sorts it out).

```
\fi\fi\fi\fi
```

We need to do the same if the current family is the typewriter family:

```
\else\ifx\f@family\ttdef@ult
```

```
\ifx\f@series\mdseries@tt \let\target@series@value\mdseries@rm
```

```
\else\ifx\f@series\bfseries@tt \let\target@series@value\bfseries@rm
```

```
\else\ifx\f@series\mddef@ult \let\target@series@value\mdseries@rm
```

```
\else\ifx\f@series\bfdef@ult \let\target@series@value\bfseries@rm
```

```
\fi\fi\fi\fi
```

```
\fi\fi
```

With these preparations for series out of the way we can now change the font family to `\rmdefault`.

```
\fontfamily\rmdefault
```

If `\target@series@value` is still empty there is nothing more to do other than selecting the new family. However, if not then we should update the font series now as well. But there is one further subtle issue. We may not have loaded an `.fd` file for our target font family yet. In the past that was done in `\selectfont` if necessary but since we are now doing all the comparisons in `\fontseries` we need to make sure that the font family specifications are already loaded prior to calling `\fontseries`.

```
\ifx\target@series@value\@empty \else
  \maybe@load@fontshape
```

Updating the series in this case means directly changing `\f@series` to the target value. We don't want to go through `\fontseries` because that would apply the mappings and then `bx + b` would keep `bx` instead of changing to `b` as desired. as

```
\let\f@series\target@series@value
\fi
\selectfont}
```

So now for the real definition: most of the code above gets delegated to a helper command `\prepare@family@series@update` so that the definition becomes again fairly short. In addition we add a hook, mainly for our Japanese friends so that the code can be extended prior to the call to `\selectfont`.

```
406 \DeclareRobustCommand\rmfamily{%
407   \not@math@alphabet\rmfamily\mathrm
```

This holds all the code discussed above, first argument is the meta family, i.e., `rm` in this case, and second argument is the default family name, e.g., `cmr` indirectly accessed via `\rmdefault`. This is calling `\fontfamily` and if necessary `\fontseries` as outline above.

```
408   \prepare@family@series@update{rm}\rmdefault
```

Then comes the hook code (by default a no-op) and finally the call to `\selectfont`.

```
409   \UseHook{rmfamily}%
410   \selectfont}
```

The definitions for `\sffamily` and `\ttfamily` are similar, the differences are only in what font families get checked.

```
\sffamily
\ttfamily
411 \DeclareRobustCommand\sffamily{%
412   \not@math@alphabet\sffamily\mathsf
413   \prepare@family@series@update{sf}\sfdefault
414   \UseHook{sffamily}%
415   \selectfont}

416 \DeclareRobustCommand\ttfamily{%
417   \not@math@alphabet\ttfamily\mathtt
418   \prepare@family@series@update{tt}\ttdefault
419   \UseHook{ttfamily}%
420   \selectfont}
```

(End of definition for `\rmfamily`, `\sffamily`, and `\ttfamily`.)

```

    rmfamily   Declare the hooks used above.
    sffamily   421 \NewHook{rmfamily}
    ttfamily   422 \NewHook{sffamily}
    normalfont 423 \NewHook{ttfamily}
expand@font@defaults 424 \NewHook{normalfont}
    bfseries   425 \NewHook{expand@font@defaults}
    bfseries/defaults 426 \NewHook{bfseries}
    mdseries   427 \NewHook{bfseries/defaults}
    mdseries/defaults 428 \NewHook{mdseries}
    mdseries/defaults 429 \NewHook{mdseries/defaults}

(End of definition for rmfamily and others.)

\@rmfamilyhook These four hooks have legacy versions used in 2020/02/02 so we should support them
\@sffamilyhook until they aren't any longer used.
\@ttfamilyhook   By default the hooks do nothing.
\@defaultfamilyhook 430 \let\@rmfamilyhook\@empty
                    431 \let\@sffamilyhook\@empty
                    432 \let\@ttfamilyhook\@empty
                    433 \let\@defaultfamilyhook\@empty   %FMI sort out

(End of definition for \@rmfamilyhook and others.)

434 </2ekernel | latexrelease>
435 <latexrelease>\EndIncludeInRelease
436 <latexrelease>\IncludeInRelease{2020/02/02}%
437 <latexrelease>           {\expand@font@defaults}{Custom series with hooks}%
438 <latexrelease>
439 <latexrelease>\def\expand@font@defaults{%
440 <latexrelease> \edef\rmdef@ult{\rmdefault}%
441 <latexrelease> \edef\sfdef@ult{\sfdefault}%
442 <latexrelease> \edef\tddef@ult{\ttdefault}%
443 <latexrelease> \edef\bfdef@ult{\bfdefault}%
444 <latexrelease> \edef\mddef@ult{\mddefault}%
445 <latexrelease> \edef\famdef@ult{\familydefault}%
446 <latexrelease>}
447 <latexrelease>
448 <latexrelease>
449 <latexrelease>\DeclareRobustCommand\rmfamily{%
450 <latexrelease> \not@math@alphabet\rmfamily\mathrm
451 <latexrelease> \prepare@family@series@update{rm}\rmdefault
452 <latexrelease> \@rmfamilyhook
453 <latexrelease> \selectfont}
454 <latexrelease>\DeclareRobustCommand\sffamily{%
455 <latexrelease> \not@math@alphabet\sffamily\mathsf
456 <latexrelease> \prepare@family@series@update{sf}\sfdefault
457 <latexrelease> \@sffamilyhook
458 <latexrelease> \selectfont}
459 <latexrelease>\DeclareRobustCommand\ttfamily{%
460 <latexrelease> \not@math@alphabet\ttfamily\mathtt
461 <latexrelease> \prepare@family@series@update{tt}\ttdefault
462 <latexrelease> \@ttfamilyhook
463 <latexrelease> \selectfont}
464 <latexrelease>\let\@rmfamilyhook\@empty
465 <latexrelease>\let\@sffamilyhook\@empty

```



```

466 <latexrelease>\let\@ttfamilyhook\@empty
467 <latexrelease>
468 <latexrelease>
469 <latexrelease>\EndIncludeInRelease
470 <latexrelease>\IncludeInRelease{0000/00/00}%
471 <latexrelease>          {\expand@font@defaults}{Custom series with hooks}%
472 <latexrelease>
473 <latexrelease>\let\expand@font@defaults\@undefined
474 <latexrelease>
475 <latexrelease>\DeclareRobustCommand\bfseries
476 <latexrelease>          {\not@math@alphabet\bfseries\mathbf
477 <latexrelease>          \fontseries\bfdefault\selectfont}
478 <latexrelease>\DeclareRobustCommand\mdseries
479 <latexrelease>          {\not@math@alphabet\mdseries\relax
480 <latexrelease>          \fontseries\mddefault\selectfont}
481 <latexrelease>\DeclareRobustCommand\rmfamily
482 <latexrelease>          {\not@math@alphabet\rmfamily\mathrm
483 <latexrelease>          \fontfamily\rmdefault\selectfont}
484 <latexrelease>\DeclareRobustCommand\sffamily
485 <latexrelease>          {\not@math@alphabet\sffamily\mathsf
486 <latexrelease>          \fontfamily\sfdefault\selectfont}
487 <latexrelease>\DeclareRobustCommand\ttfamily
488 <latexrelease>          {\not@math@alphabet\ttfamily\mathtt
489 <latexrelease>          \fontfamily\ttdefault\selectfont}
490 <latexrelease>
491 <latexrelease>\let\@rmfamilyhook\@undefined
492 <latexrelease>\let\@sffamilyhook\@undefined
493 <latexrelease>\let\@ttfamilyhook\@undefined
494 <latexrelease>
495 <latexrelease>\EndIncludeInRelease
496 (*2ekernel)

```

`\IfFontSeriesContextTF` With the ability for `\bfseries` or `\mdseries` to be mapped to different NFSS axis values it becomes important to have the ability to determine the current context as we can no longer look at `\f@series` to answer a question such as “am I currently typesetting in a bold typeface?”

This is provided by the test `\IfFontSeriesContextTF`. It takes three arguments:

- The context we try to check (either `bf` for bold or `md` for medium, i.e., the same that can go into the first mandatory argument of `\DeclareFontSeriesDefault`),
- what to do if we are in this context (true case) and
- what to do if we are not (false case).

This allows you to define commands like `\IfBold`, e.g.,

```
\NewDocumentCommand\IfBold{mm}{\IfFontSeriesContextTF{bf}{#1}{#2}}
```

and then do

```
This is \IfBold{bold}{non-bold} text.
```

and get the appropriate result.

```
497 </2ekernel>
498 <*2ekernel | latexrelease>
499 <latexrelease>\IncludeInRelease{2020/10/01}%
500 <latexrelease>          {\IfFontSeriesContextTF}{Font series context}%
501 \DeclareRobustCommand\IfFontSeriesContextTF[1]{%
502   \expand@font@defaults
```

In the beginning we haven't found the context we are looking for.

```
503   \@font@series@contextfalse
```

We store the requested context away for use in the tests.

```
504   \def\requested@test@context{#1}%
```

The next definition is there to ensure that get a final match during testing even if the current family is non of the meta families (`rm`, `sf` or `tt`). This will then basically tests if the current font family matches the overall default.

```
505   \expandafter\edef\csname ??def@ult\endcsname{\f@family}%
```

Then we run through the meta family list (currently containing just the three values) followed by the artificial meta family `??` and test each of them in turn using `\test@font@series@context` as the testing command.

```
506   \let\@elt@test@font@series@context
507         \@meta@family@list
508         \@elt{??}%
509   \let\@elt\relax
```

Following that we evaluate the status of `\if@font@series@context` to determine which of the remaining arguments (true/false case) we have to execute.

```
510   \if@font@series@context
511   \expandafter\@firstoftwo
512   \else
513   \expandafter\@secondoftwo
514   \fi
515 }
```

(End of definition for \IfFontSeriesContextTF.)

`\test@font@series@context` This tests the context (stored in `\requested@test@context`) and updates the boolean if the right context is found.

```
516 \def\test@font@series@context#1{%
```

First task is to figure out whether the current family matches `\rmfamily`, `\sffamily`, etc. so in `\reserved@a` we store the value of `\rmdef@ult` (or whatever the given meta family is) and compare that to `\f@family`.

```
517   \edef\reserved@a{\csname #1def@ult\endcsname}%
518   \ifx\f@family\reserved@a
```

If they match we have found the right meta family so we don't need to test any of the remaining meta family and therefore change `\@elt` to `\@gobble`.

```
519   \let\@elt\@gobble
```

Now we have to test if `\f@series` matches the requested context (e.g., whether `\bfseries@rm` has that value if the current meta family is `rm` and we are looking for the `bf` context).

```
520     \expandafter\ifx
521         \csname\requested@test@context series@#1\endcsname\f@series
```

If yes we change the boolean and are done.

```
522     \@font@series@contexttrue
```

If not then maybe the reason is that nothing special was set up for that meta family so we also check now check if `\f@series` matches the overall default (e.g., `\bfdef@ult` if we are looking for the bold context). If that matches we change the boolean.

```
523     \else
524     \expandafter\ifx
525         \csname\requested@test@context def@ult\endcsname\f@series
526     \@font@series@contexttrue
527 \fi\fi\fi
528 }
```

(End of definition for `\test@font@series@context`.)

`\if@font@series@context` The boolean to signal if we found the requested font series context.

```
529 \newif\if@font@series@context
```

(End of definition for `\if@font@series@context`.)

```
530 </2kernel | latexrelease>
531 <latexrelease>\EndIncludeInRelease
532 <latexrelease>\IncludeInRelease{0000/00/00}%
533 <latexrelease>         {\IfFontSeriesContextTF}{Font series context}%
534 <latexrelease>
535 <latexrelease>\let\IfFontSeriesContextTF\@undefined
536 <latexrelease>\let\test@font@series@context\@undefined
537 <latexrelease>\let\if@font@series@context\@undefined
538 <latexrelease>\let\@font@series@contexttrue\@undefined
539 <latexrelease>\let\@font@series@contextfalse\@undefined
540 <latexrelease>\EndIncludeInRelease
541 <*2kernel>
```

3 Supporting nested emphasis

By default $\text{\LaTeX} 2_{\epsilon}$ supports two levels of nested emphasis: if the current font has an upright shape then it switches to `\itshape` otherwise to `\emminnershape` (which defaults to `\upshape`). This means nested emphasis will oscillate between italic and upright shapes.

Sometimes it would be nice to allow for a more lengthy sequence, but instead of providing a fixed one \LaTeX now offers a general mechanism that allows to define arbitrary sequences.

```
\DeclareEmphSequence
\emforce
```

This declaration expects a comma separated list of (font) change declarations corresponding to increasing levels of emphasis. The mechanism tries to be “smart” and verifies that the declarations actually alter the font. If not it will ignore this level and tries the next one—the assumption being that there was a manual font change in the document

to the font that is now supposed to be used for emphasis. Of course, this only works if the declarations in the list actually change the font and not, say, just the color. In such a case one has to use `\emforce` to which directs the mechanism to use the level even if the font attributes haven't changed.

`\emreset` If the nesting is so deep, that the specified levels are exhausted then `\emreset` is used as a final set of declarations (which by default returns back to the upright shape). Any additional nesting levels will then reuse the list from its beginning.

`\DeclareEmphSequence` `\DeclareEmphSequence` expects a list of declaration. Spaces in the argument are dropped to avoid spurious spaces in the output. The declarations are additive. At the very end the shape is reset using `\emreset` and `\emforce` so that this case is never skipped.³⁸ Further nested calls restart at the beginning.

```
542 </2ekernel>
543 <*2ekernel | latexrelease>
544 <latexrelease>\IncludeInRelease{2020/02/02}%
545 <latexrelease>          {\DeclareEmphSequence}{Nested emph}%
546 \def\DeclareEmphSequence#1{%
547   \protected@edef\emfontdeclare@clist{\zap@space#1, \@empty\emforce\emreset}%
548 }
```

By default the it is empty, in which case `\emminershape` is used by L^AT_EX.

```
549 \let\emfontdeclare@clist\@empty
```

(End of definition for \DeclareEmphSequence.)

`\emrest` Reset the font to upright and upper/lower case. With the default rules using `\shapedefault` does that for us but to be on the safe side we do it like this:

```
550 \DeclareRobustCommand\emreset{\upshape\ulcshape}
```

(End of definition for \emrest.)

`\em` The new definition for `\em` (and implicitly `\emph`) is the same as before as long as `\emfontdeclare@clist` is empty.

```
551 \DeclareRobustCommand\em{%
552   \@nomath\em
553   \ifx\emfontdeclare@clist\@empty
554     \ifdim \fontdimen\@ne\font >\z@
555       \emminershape \else \itshape \fi
556   \else
```

But if not we use the list to decide how to do emphasis.

We use the current font to check if the declarations have any effect, so even a size change is allowed and identified as a modification (but a color change, for example, isn't). So first we save the current status.

```
557   \edef\em@currfont{\csname\curr@fontshape/\f@size\endcsname}%
558
```

Then we grab the next element from the list and check if it can be used.

```
559   \expandafter\do@emfont@update\emfontdeclare@clist\do@emfont@update
560   \fi
561 }
```

³⁸Maybe we should not add `\emforce` but allow that case to be skipped as well. Of course, that might result in an endless loop if somebody defines a sequence without any font change and without `\emforce` but ...

```
561 \def\eminnershape{\upshape}
```

(End of definition for \em.)

`\do@emfont@update` We know that the list (if not empty) has at least 2 elements separated by a comma, so we pick up the first in #1 and the rest in #2.

```
562 \def\do@emfont@update#1,#2\do@emfont@update{%
```

First action is to alter the list and move the first entry to the end

```
563 \def\emfontdeclare@clist{#2,#1}%
```

Then we execute the current declaration. Appending `\selectfont` means one can write just `\fontshape{it}` and that works then too.

```
564 % \typeout{Use: \detokenize{#1}}%
```

```
565 #1\selectfont
```

We then compare the current font with our saved version, but with a slight twist: we add `\em@force` at the end of the name. Normally this is empty so has no effect but if there was an `\emforce` as part of #1 it will append a / to the font name (making it invalid) thus this will then always fail the test.

If the test fails we are done and the declarations will be used. Otherwise we will try the next declaration in the sequence.

```
566 \expandafter\ifx\csname \curr@fontshape/\f@size\em@force
```

For the comparison with `\ifx` we have to expand `\em@currfont` once as the relevant info is inside.

```
567 \expandafter\endcsname
```

```
568 \em@currfont
```

```
569 \expandafter\do@emfont@update\emfontdeclare@clist\do@emfont@update
```

If `\emforce` was used, we have to undo its effect:

```
570 \else
```

```
571 \let\em@force\@empty
```

```
572 \fi
```

```
573 }
```

(End of definition for `\do@emfont@update`.)

`\emforce` `\em@force` The definition of `\emforce` is simple: change `\em@force` to make the above test always invalid.

```
574 \protected\def\emforce{\def\em@force{/}}
```

```
575 \let\em@force\@empty
```

```
576 </2ekernel | latexrelease>
```

```
577 <latexrelease>\EndIncludeInRelease
```

(End of definition for `\emforce` and `\em@force`.)

`\em` `\eminnershape` These are the older definitions for `\em`, prior to 2020.

We also have to define the *emphasize* font change command (i.e. `\em`). This command will look is the current font is sloped (i.e. has a positive `\fontdimen1`) and will then select either `\upshape` or `\itshape`.

```
578 <latexrelease>\IncludeInRelease{2015/01/01}{\DeclareEmphSequence}{Nested emph}%
```

```
579 <latexrelease>\let\DeclareEmphSequence\@undefined
```

```
580 <latexrelease>\let\emfontdeclare@clist\@undefined
```

```
581 <latexrelease>\let\emreset\@undefined
```

```
582 <latexrelease>\let\do@emfont@update\@undefined
```

```

583 <latexrelease>\let\emforce\@undefined
584 <latexrelease>\let\em@force\@undefined
585 <latexrelease>
586 <latexrelease>\DeclareRobustCommand\em
587 <latexrelease>          {\@nomath\em \ifdim \fontdimen\@ne\font >\z@
588 <latexrelease>          \emminnershape \else \itshape \fi}%
589 <latexrelease>\EndIncludeInRelease
590 <latexrelease>
591 <latexrelease>\IncludeInRelease{0000/00/00}{\DeclareEmphSequence}{Nested emph}%
592 <latexrelease>\DeclareRobustCommand\em
593 <latexrelease>          {\@nomath\em \ifdim \fontdimen\@ne\font >\z@
594 <latexrelease>          \upshape \else \itshape \fi}%
595 <latexrelease>\let\emminnershape\@undefined
596 <latexrelease>\EndIncludeInRelease
597 (*2ekernel)

```

(End of definition for \em and \emminnershape.)

`\not@math@alphabet` This function generates an error message when it is called in math mode. The same function should be defined in `newlfont.sty`.

```

598 \def\not@math@alphabet#1#2{%
599   \relax
600   \ifmmode
601     \@latex@error{Command \noexpand#1invalid in math mode}%
602     {%
603       Please
604       \ifx#2\relax
605         define a new math alphabet^^J%
606         if you want to use a special font in math mode%
607       \else

```

We have to a `\noexpand` below to prevent expansion of #2. In case of #1 we can omit this (due to the current definition of robust commands since they do come out right there :-).

```

608         use the math alphabet \noexpand#2instead of
609         the #1command%
610       \fi
611       .
612     }%
613 \fi}

```

(End of definition for \not@math@alphabet.)

Finally we provide two abbreviations to switch to the L^AT_EX versions.

```

614 \DeclareRobustCommand\boldmath{\@nomath\boldmath
615   \mathversion{bold}}
616 \DeclareRobustCommand\unboldmath{\@nomath\unboldmath
617   \mathversion{normal}}

```

Here we switch to the default math version by defining the internal macro `\math@version`. We dare not to call `\mathversion` at this place because this would call `\glb@settings`.

```

618 \def\math@version{normal}

```

3.1 Legacy

We start by defining a few macros that are part of standard L^AT_EX's user interface. The use of these functions is not encouraged, but they will allow to process older documents without changes to the source.

```
\newfont
619 \def\newfont#1#2{\@ifdefinable#1{\font#1=#2\relax}}
(End of definition for \newfont.)

\symbol
620 </2ekernel>
621 <*2ekernel | latexrelease>
622 <latexrelease>\IncludeInRelease{2020/10/01}%
623 <latexrelease>          {\symbol}{XeTeX change for math}%
624 \ifdefined\XeTeXversion
625   \DeclareRobustCommand\symbol[1]{\Ucharcat#1 12\relax}
626 \else
627   \DeclareRobustCommand\symbol[1]{\char#1\relax}
628 \fi
629 </2ekernel | latexrelease>

630 <latexrelease>\EndIncludeInRelease
631 <latexrelease>\IncludeInRelease{0000/00/00}%
632 <latexrelease>          {\symbol}{XeTeX change for math}%
633 <latexrelease>
634 <latexrelease>\DeclareRobustCommand\symbol[1]{\char#1\relax}
635 <latexrelease>
636 <latexrelease>\EndIncludeInRelease
637 <*2ekernel>

(End of definition for \symbol.)
```

3.2 Miscellaneous

```
\@setfontsize This abbreviation is used by LATEX's user level size changing commands, such as \large.
\@setsize    638 \def\@setfontsize#1#2#3{\@nomath#1%
For the benefit of people relying on keeping the name of the current font command saved
in \@currsiz we define it. To ensure that \@setfontsize keeps being robust we omit
this assignment during times where \protect differs from \@typeset@protect.
639   \ifx\protect\@typeset@protect
640     \let\@currsiz#1%
641   \fi
642   \fontsize{#2}{#3}\selectfont}
For compatibility we also define \@setsize the 209 command
643 <*compat>
644 \def\@setsize#1#2#3#4{\@setfontsize#1{#4}{#2}}
645 </compat>

(End of definition for \@setfontsize and \@setsize.)
```

`\hexnumber@` To set up L^AT_EX's special math character definitions we first provide a macro to generate hexadecimal numbers. It is a rather simple `\ifcase`.

```
646 \def\hexnumber@#1{\ifcase\number#1
647 0\or 1\or 2\or 3\or 4\or 5\or 6\or 7\or 8\or
648 9\or A\or B\or C\or D\or E\or F\fi}
```

(End of definition for \hexnumber@.)

`\nfss@text` In its simplest form `\nfss@text` is an `\mbox`. This will produce unbreakable text outside math and inside math you will get text with the same fonts as outside. The only drawback is that such item won't change sizes in subscripts. But this behavior can be easily changed. With the `amstex` style option one will get a sub style called `amstext` which will redefine the `\nfss@text` macro to produce correct text in all sizes.

We have to use `\def` instead of the shorter `\let` since `\mbox` is undefined when we reach this point.

```
649 \def\nfss@text#1{{\mbox{#1}}}
```

(End of definition for \nfss@text.)

`\copyright` The definition of `\copyright` was changed so that it works in other type styles, and to make it robust. We leave the family untouched so that the copyright notice will come out differently if a different font family is in use. This command is commented out, since it is now defined in `ltoutenc.dtx`.

```
650 %\DeclareRobustCommand\copyright
651 %    {\ooalign{\hfil
652 %        \raise.07ex\hbox{\mdseries\upshape c}\hfil\crcr
653 %        \mathhexbox20D}}}
```

(End of definition for \copyright.)

`\normalfont` The macro `\reset@font` is used in L^AT_EX to switch to a standard font, in order to initialize the current font in situations where typesetting is done in a new visual context (e.g. in a footnote). We define it here to allow the test for the new L^AT_EX version above but nevertheless are able to run all kind of mixtures.

The user interface name for `\reset@font` is `\normalfont`:

```
654 </2kernel>
655 <*2kernel | latexrelease>
656 <latexrelease>\IncludeInRelease{2021/06/01}%
657 <latexrelease>          {\normalfont}{Add hook to \normalfont}%
658 \DeclareRobustCommand\normalfont%
```

Instead of calling `\usefont`, as it was done in the past, we inline the code from `\usefont` as we want to add the hook before `\selectfont`, but after all the font attributes are set.

```
659 \fontencoding\encodingdefault
660 \edef\f@family{\familydefault}%
661 \edef\f@series{\seriesdefault}%
662 \edef\f@shape{\shapedefault}%
```

Any earlier `\fontseries`, etc. should be canceled and we should switch unconditionally to the requested font face so we drop any code that may have been stored in `\delayed@f@adjustment`.

```
663 \let\delayed@f@adjustment\@empty
```

```
664 \UseHook{normalfont}%
```


This is the old name for the hook introduced in 2020/02/02. It will be removed in one of the future releases!

```

665 \@defaultfamilyhook      % hookname from 2020/02 will vanish
666 \selectfont}

667 \let\reset@font\normalfont

(End of definition for \normalfont and \reset@font.)

668 % \changes{v3.2g}{2021/03/18}
669 %      {Add missing 2020/02/02 latexrelease entry.}
670 </2ekernel | latexrelease>
671 <latexrelease>\EndIncludeInRelease
672 <latexrelease>
673 <latexrelease>\IncludeInRelease{2020/10/01}%
674 <latexrelease>      {\normalfont}{Add hook to \normalfont}%
675 <latexrelease>
676 <latexrelease>\DeclareRobustCommand\normalfont{%
677 <latexrelease>  \fontencoding\encodingdefault
678 <latexrelease>  \edef\f@family{\familydefault}%
679 <latexrelease>  \edef\f@series{\seriesdefault}%
680 <latexrelease>  \edef\f@shape{\shapedefault}%
681 <latexrelease>  \UseHook{normalfont}%
682 <latexrelease>  \@defaultfamilyhook      % hookname from 2020/02 will vanish
683 <latexrelease>  \selectfont}
684 <latexrelease>
685 <latexrelease>\let\reset@font\normalfont
686 <latexrelease>
687 <latexrelease>\EndIncludeInRelease
688 <latexrelease>
689 <latexrelease>\IncludeInRelease{2020/02/02}%
690 <latexrelease>      {\normalfont}{Add hook to \normalfont}%
691 <latexrelease>
692 <latexrelease>\DeclareRobustCommand\normalfont{%
693 <latexrelease>  \fontencoding\encodingdefault
694 <latexrelease>  \edef\f@family{\familydefault}%
695 <latexrelease>  \edef\f@series{\seriesdefault}%
696 <latexrelease>  \edef\f@shape{\shapedefault}%
697 <latexrelease>  \@defaultfamilyhook
698 <latexrelease>  \selectfont}
699 <latexrelease>
700 <latexrelease>\let\reset@font\normalfont
701 <latexrelease>
702 <latexrelease>\let\@defaultfamilyhook\@empty
703 <latexrelease>
704 <latexrelease>\EndIncludeInRelease
705 <latexrelease>
706 <latexrelease>\IncludeInRelease{0000/00/00}%
707 <latexrelease>      {\normalfont}{Add hook to \normalfont}%
708 <latexrelease>
709 <latexrelease>\DeclareRobustCommand\normalfont
710 <latexrelease>      {\usefont\encodingdefault
711 <latexrelease>          \familydefault
712 <latexrelease>          \seriesdefault
713 <latexrelease>          \shapedefault

```

```

714 <latexrelease>          \relax}
715 <latexrelease>\let\reset@font\normalfont
716 <latexrelease>
717 <latexrelease>\let\@defaultfamilyhook\@undefined
718 <latexrelease>
719 <latexrelease>\EndIncludeInRelease
720 <*2ekernel>

```

We left out the special L^AT_EX fonts which are not automatically included in the base version of the font selection since these fonts contain only a few characters which are also included in the AMS fonts so anybody who is using these fonts doesn't need them. But for compatibility reasons we will define these symbols.

```

721 \def\not@base#1{\@latex@error
722   {Command \noexpand#1not provided in base LaTeX2e}%
723   {Load the latexsym or the amsfonts package to
724     define this symbol}}
725 \def\mho{\not@base\mho}
726 \def\Join{\not@base\Join}
727 \def\Box{\not@base\Box}
728 \def\Diamond{\not@base\Diamond}
729 \def\leadsto{\not@base\leadsto}
730 \def\sqsubset{\not@base\sqsubset}
731 \def\sqsupset{\not@base\sqsupset}
732 \def\lhd{\not@base\lhd}
733 \def\unlhd{\not@base\unlhd}
734 \def\rhd{\not@base\rhd}
735 \def\unrhd{\not@base\unrhd}

```

We now initialize all variables set by `\DeclareErrorFont`. These values are not really important since they will be overwritten later on by the definition in `fontdef.ltx`.

However, if `fontdef.cfg` is corrupted then at least a hopefully suitable error font is present.

```

736 \DeclareErrorFont{OT1}{cmr}{m}{n}{10}  %% don't modify this setting
737                                         %% overwrite it in fontdef.cfg
738                                         %% if necessary

```

We also set some default values for `\f@family` etc. Note that we don't yet have any encodings that comes later. In the past this was implicitly done by `\DeclareErrorFont`.

```

739 \fontfamily{cmr}

```

Previously the default values for series and shape were set by calling `\fontseries` and `\fontshape`, but their action is now delayed until `\selectfont` which isn't called inside the format (to avoid unnecessarily loading a font that may never get used). We therefore have to set `\f@series` and `\f@shape` directly instead.

```

740 \def\f@series{m}          % \fontseries{m}
741 \def\f@shape{n}         % \fontshape{n}
742 \fontsize{10}{10}

```

The initial fontenc package load list. This will get overwritten in `fonttext` and is only provided in case an old `fonttext.cfg` does not define the command:

```

743 \def\@fontenc@load@list{\@elt{T1,OT1}}

```

We now load the customizable parts of NFSS.

```

744 \InputIfFileExists{fonttext.cfg}
745     {\typeout{=====^^J%
746             ^^J%
747             Local config file fonttext.cfg used^^J%
748             ^^J%
749             =====}%
750     \def\@addtofilelist##1{\xdef\@filelist{\@filelist,##1}}%
751     }
752     {\input{fonttext.ltx}}
753 \let\@addtofilelist\@gobble

```

Ditto for math although I don't think that we will get a lot of customisation :-)

```

754 \InputIfFileExists{fontmath.cfg}
755     {\typeout{=====^^J%
756             ^^J%
757             Local config file fontmath.cfg used^^J%
758             ^^J%
759             =====}%
760     \def\@addtofilelist##1{\xdef\@filelist{\@filelist,##1}}%
761     }
762     {\input{fontmath.ltx}}
763 \let\@addtofilelist\@gobble

```

Then we preload several fonts. This file might be customized *without* changing the behavior of the format (i.e. necessary font definitions will be loaded at runtime if they are not preloaded). This is done in the file `preload.ltx`.

```

764 \InputIfFileExists{preload.cfg}
765     {\typeout{=====^^J%
766             ^^J%
767             Local config file preload.cfg used^^J%
768             ^^J%
769             =====}%
770     \def\@addtofilelist##1{\xdef\@filelist{\@filelist,##1}}%
771     }
772     {\input{preload.ltx}}
773 \let\@addtofilelist\@gobble

```

`\seriesdefault` After `\seriesdefault` got defined inside `fonttext.ltx` or a `.cfg` file overwriting it, we `\seriesdefault@kernel` alter its value by appending `\@empty` to it. This will vanish if expanded but allows us to check if the default gets altered (even to the same value) in the document preamble. All we have to do is to save the current value somewhere and later compare the two. For this we use `\seriesdefault@kernel`.

```

774 \expandafter\def\expandafter\seriesdefault\expandafter{\seriesdefault\@empty}
775 \let\seriesdefault@kernel\seriesdefault

```

(End of definition for `\seriesdefault` and `\seriesdefault@kernel`.)

`\@acci` We also save the values of some accents in `\@acci`, `\@accii` and `\@acciii` so they can `\@accii` be restored by a `minipage` inside a `tabbing` environment. `\@acciii`

```

776 \let\@acci\' \let\@accii\' \let\@acciii\=

```

(End of definition for `\@acci`, `\@accii`, and `\@acciii`.)

`\cal` Here were the two old `\langle alphabet identifiers \rangle`.
`\mit`

(End of definition for \ca1 and \mit.)

777 </2ekernel>

File 30

fontdef.dtx

1 Introduction

This file is used to generate the files `fonttext.ltx` (text font declarations) and `fontmath.ltx` (math font declarations), which are used during the format generation. It contains the declaration of the standard text encodings used at the site as well as a minimal subset of font shape groups that NFSS will look at to ensure that the specified encodings are valid.

The math part contains the setup for math encodings as well as the default math symbol declarations that belong to the encoding.

It is possible to change this setup (by using other fonts, or defaults) without losing the ability to process documents written at other sites. Portability in this sense means that a document will compile without errors. It does not mean, however, that identical output will be produced. For this it is necessary that the distributed setup is used at both installations.

2 Customization

You are not allowed to change this source file! If you want to change the default encodings and/or the font shape groups preloaded you should create a copy of `fonttext.ltx` under the name `fonttext.cfg` and change this copy. If $\text{\LaTeX} 2_{\epsilon}$ finds a file of this name it will use it, otherwise it uses the standard file which is `fontdef.ltx`.

If you don't plan to use Computer Modern much or at all, it might (!) be a good idea to make your own `fonttext.cfg`. Look at the comments below (docstrip module 'text') to see what should go into such a file.

To change the math font setup use a copy of `fontmath.ltx` under the name `fontmath.cfg` and change this copy. However, dealing with this interface is even more a job for an expert than changing the text font setup — in short, we don't encourage either.

Warning: please note that we don't support customised \LaTeX versions. Thus, before sending in a bug report please try your test file with a \LaTeX format which is not customised and send in the log from that version (unless the problem goes away).

Please note: the following standard encodings have to be defined in all local variants of `font...cfg` to guarantee that all \LaTeX installations behave in the same way.

T1	Cork \TeX text encoding
OT1	old \TeX text encoding
U	unknown encoding
OML	old \TeX math letters encoding
OMS	old \TeX math symbols encoding
OMX	old \TeX math extension symbols encoding
TU	Unicode

Notice that some of these encodings are ‘old’ in the sense that we hope that they will be superseded soon by encoding standards defined by the T_EX user community. Therefore this set of default encodings may change in the future.

The first candidate is OT1 which will soon be replaced by T1, the official T_EX text encoding.

Warning: If you add additional encodings to this file there is no guarantee any longer that files processable at your installation will also be processable at other installations. Thus, if you make use of such an encoding in your document, e.g. if you intend to typeset in Cyrillic (OT2 encoding), you need to specify this encoding in the preamble of your document prior to sending it to another installation. Once the encoding is specified in that place in your document, the document is processable at all L^AT_EX installations (provided they have suitable fonts installed).

For this reason we suggest that you define a short package file that sets up an additional encoding used at your site (rather than putting the encoding into this file) since this package can easily be shipped with your document.

3 The docstrip modules

The following modules are used to direct `docstrip` in generating external files:

<code>driver</code>	produce a documentation driver file
<code>text</code>	produce the file <code>fonttext.ltx</code>
<code>math</code>	produce the file <code>fontmath.ltx</code>
<code>cfgtext</code>	produce a dummy <code>fonttext.cfg</code> file
<code>cfgmath</code>	produce a dummy <code>fontmath.cfg</code> file

A typical `docstrip` command file would then have entries like:

```
generateFile{fonttext.ltx}{t}{\from{fontdef.dtx}{text}}
```

4 A driver for this document

The next bit of code contains the documentation driver file for T_EX, i.e. the file that will produce the documentation you are currently reading. It will be extracted from this file by the DOCSTRIP program.

```
1 <*driver>
2 \documentclass{ltxdoc}
3 \GetFileInfo{fontdef.dtx}
4 \begin{document}
5   \DocInput{fontdef.dtx}
6 \end{document}
7 </driver>
```

5 The fonttext.ltx file

The identification is done earlier on with a `\ProvidesFile` declaration.

```
8 <*text>
9 \typeout{=== Don't modify this file, use a .cfg file instead ===^^J}
```

5.1 Encodings

This file declares the standard encodings for text and math fonts. All others should be declared in packages or in the documents directly.

For every text encoding there are normally a number of encoding specific commands, e.g. accents, special characters, etc. (The definition for such a command might have to change when the encoding is changed, because the character is in a different position, or not available at all, or the accent is produced in a different way.) This is handled by a general mechanism which is described in `ltoutenc.dtx`.

By convention, text encoding specific declarations, including the `\DeclareFontEncoding` declaration, are kept in separate file of the form `<enc>enc.def`, e.g. `ot1enc.def`. This allows other applications to make use of the declarations as well.

Similar to the default encoding, the loading of the encoding files for the two major text encodings shouldn't be changed. In particular, the `inputenc` package depends on this.

```
10 \input {omlenc.def}
11 \input {omsenc.def}
```

Documents containing a lot of accented characters should really be using T1 fonts. We therefore load this after OT1 so that T1 encoding specific commands are executed as fast as possible (encoding files are no longer reloaded in `fontenc`).

```
12 \input {ot1enc.def}
13 \input {t1enc.def}
14 \input {ts1enc.def}

15 \ifx\Umathcode\@undefined
```

We then set the default text font encoding. This will hopefully change some day to T1. This setting should *not* be changed to produce a portable format.

```
16 \fontencoding{OT1}

  The initial fontenc package load list if an 8-bit TEX engine is used:
17 \def\@fontenc@load@list{\@elt{T1,OT1}}
18 \def\rmsubstdefault{cmr}
19 \def\sfsubstdefault{cmss}
20 \def\ttsubstdefault{cmtt}
21 \LoadFontDefinitionFile{TS1}{cmr}

22 \else
```

Unicode.

```
23 \input {tuenc.def}
24 \fontencoding{TU}

  The initial fontenc package load list if a Unicode engine is used:
25 \def\@fontenc@load@list{\@elt{TU}}
26 \DeclareFontSubstitution{TU}{lmr}{m}{n}
27 \LoadFontDefinitionFile{TU}{lmr}
28 \LoadFontDefinitionFile{TU}{lmss}
29 \LoadFontDefinitionFile{TU}{lmtt}

30 \def\rmsubstdefault{lmr}
31 \def\sfsubstdefault{lmss}
32 \def\ttsubstdefault{lmtt}
33 \LoadFontDefinitionFile{TS1}{lmr}
```

```
34 \DeclareFontSubstitution{TU}{lmr}{m}{n}
```

End of Unicode branch.

```
35 \fi
```

If different encodings for text fonts are in use one could put the common setup into `\DeclareFontEncodingDefaults`. There is now a better mechanism so using this interface is discouraged!

```
36 \DeclareFontEncodingDefaults{}{}
```

The default font substitution for an encoding is defined in the corresponding `...enc.def` file so for OT1, T1, and TS1 this is already defined.

```
37 %\DeclareFontSubstitution{T1}{cmr}{m}{n}
```

```
38 %\DeclareFontSubstitution{OT1}{cmr}{m}{n}
```

```
39 %\DeclareFontSubstitution{TS1}{cmr}{m}{n}
```

This release of L^AT_EX 2_ε assumes that the ec fonts are available. It is possible to change this to point to some other font family (e.g., Times with the appropriate encoding if it is available) without making documents non-portable. However, in such a case documents will produce different page breaks at other sites. The substitution defaults can all be changed without losing portability as long as there are font shape definitions for the selected substitutions.

For every encoding declaration, L^AT_EX 2_ε will try to verify that the given substitution information makes sense, i.e. that it is impossible to go into an endless loop if font substitution happens. This is done at the moment the `\begin{document}` is encountered. L^AT_EX 2_ε will then check that for every encoding the substitution defaults form a valid font shape group, which means that it will check if there is a `\DeclareFontShape` declaration for this combination. We will therefore load the corresponding `.fd` files now. If we don't do this they would be loaded at verification time (i.e. at `\begin{document}`) which would delay processing unnecessarily.

Warning: Please note that this means that you have to regenerate the format whenever you change any of these `.fd` files since L^AT_EX 2_ε will not read `.fd` files if it already knows about the encoding/family combination.

The `\nfss@catcodes` ensures that white space is ignored in any definitions made in the `fd` files.

```
40 \begingroup
```

```
41 \nfss@catcodes
```

```
42 \input {t1cmr.fd}
```

```
43 \input {ot1cmr.fd}
```

```
44 \input {ts1cmr.fd}
```

```
45 \endgroup
```

We also load some other font definition files which are normally needed in a document. This is only done for processing speed and you can comment the next two lines out to save some memory. If necessary these files are then loaded when your document is processed. (Loading `.fd` files is a less drastic step compared to preloading fonts because the number of fonts is limited 255 at (nearly) every T_EX installation, while the amount of main memory is not a limiting factor at most installations.)

```
46 \begingroup
```

```
47 \nfss@catcodes
```

```
48 \input {t1cmss.fd}
```



```

49 \input {t1cmtt.fd}
50 \input {ot1cmss.fd}
51 \input {ot1cmtt.fd}

52 \input {ts1cmss.fd}
53 \input {ts1cmtt.fd}
54 \endgroup

```

Even though Unicode engines default to `lm` load `ts1cmr` as this may be used for fallback for TS1 encoding.

We now load it in all engines above, so the next lines are no longer necessary. We keep them here if we stop loading other `fd` files in Unicode engines.

```

55 %\ifx\Umathcode\@undefined\else
56 %\begingroup
57 %\nfss@catcodes
58 %\input {ts1cmr.fd}
59 %\endgroup
60 %\fi

```

Even with all the precautions it is still possible that NFSS will run into problems, for example, when a `.fd` file contains corrupted data. To guard against such cases NFSS has a very low-level fallback font that is installed with the following line.

```

61 \DeclareErrorFont{OT1}{cmr}{m}{n}{10}

```

This means, “if everything else fails use Computer Modern Roman normal shape at 10pt in the old text encoding”. You can change the font used but the encoding should be the same as the one specified with `\fontencoding` above.

5.2 Defaults

To allow the use of `\rmfamily`, `\sffamily`, etc. in documents even if non-standard families are used we provide nine macros which hold the name of the corresponding families, series, and so on. This makes it easy to use other font families (like Times Roman, etc.). One simply has to redefine these defaults.

All these hooks have to be defined in this file but you can change their meaning (except for `\encodingdefault`) without making documents non-portable.

```

\encodingdefault The following three definitions set up the meaning for \rmfamily, \sffamily, and
  \rmdefault      \ttfamily.
  \sfdefault      62 \ifx\Umathcode\@undefined
  \ttdefault      63 \newcommand\encodingdefault{OT1}
                  64 \newcommand\rmdefault{cmr}
                  65 \newcommand\sfdefault{cmss}
                  66 \newcommand\ttdefault{cmtt}
                  67 \else
                  68 \newcommand\encodingdefault{TU}
                  69 \newcommand\rmdefault{lmr}
                  70 \fontfamily{\rmdefault}
                  71 \newcommand\sfdefault{lms}
                  72 \newcommand\ttdefault{lmtt}
                  73 \fi
                  74 \</text>
                  75 \<latexrelease>\IncludeInRelease{2017/01/01}%
                  76 \<latexrelease>          {\encodingdefault}{TU encoding default}%

```

```

77 <latexrelease>\ifx\Umathcode\@undefined
78 <latexrelease>\renewcommand\encodingdefault{OT1}
79 <latexrelease>\fontencoding{\encodingdefault}
80 <latexrelease>\renewcommand\rmdefault{cmr}
81 <latexrelease>\fontfamily{\rmdefault}
82 <latexrelease>\renewcommand\sfddefault{cmss}
83 <latexrelease>\renewcommand\ttdefault{cmtt}
84 <latexrelease>\else
85 <latexrelease>\renewcommand\encodingdefault{TU}
86 <latexrelease>%done in everyjob\fontencoding{\encodingdefault}
87 <latexrelease>\renewcommand\rmdefault{lmr}
88 <latexrelease>\fontfamily{\rmdefault}
89 <latexrelease>\renewcommand\sfddefault{lms}
90 <latexrelease>\renewcommand\ttdefault{lmtt}
91 <latexrelease>\fi
92 <latexrelease>\EndIncludeInRelease
93 <latexrelease>\IncludeInRelease{0000/00/00}%
94 <latexrelease>          {\encodingdefault}{TU encoding default}%
95 <latexrelease>\fontencoding{OT1}
96 <latexrelease>\renewcommand\encodingdefault{OT1}
97 <latexrelease>\fontencoding{\encodingdefault}
98 <latexrelease>\renewcommand\rmdefault{cmr}
99 <latexrelease>\fontfamily{\rmdefault}
100 <latexrelease>\renewcommand\sfddefault{cmss}
101 <latexrelease>\renewcommand\ttdefault{cmtt}
102 <latexrelease>\EndIncludeInRelease
103 <*text>

```

(End of definition for \encodingdefault and others.)

```

\bfdefault Series changing commands are influenced by the following hooks.
\mddefault 104 \newcommand\bfdefault{b} % overwritten below (for rollback)
105 \newcommand\mddefault{m} % overwritten below (for rollback)

```

(End of definition for \bfdefault and \mddefault.)

```

\itdefault Shape changing commands use the following hooks.
\sldefault 106 \newcommand\itdefault{it}
\scdefault 107 \newcommand\sldefault{sl}
\updefault 108 \newcommand\scdefault{sc}
109 \newcommand\updefault{up} % overwritten below (for rollback)

```

(End of definition for \itdefault and others.)

```

110 </text>
111 <*text | latexrelease>
112 <latexrelease>\IncludeInRelease{2020/02/02}%
113 <latexrelease>          {\updefault}{font defaults change}%
114 \renewcommand\updefault{up}

```

We append \@empty to the series value so that we can detect if it got changed via \def or \renewcommand later.

```

115 \renewcommand\bfdefault{b\@empty}
116 \renewcommand\mddefault{m\@empty}

```

```

117 \let\bfdefault@previous\bfdefault
118 \let\mddefault@previous\mddefault
119 </text | latexrelease>
120 <latexrelease>\EndIncludeInRelease
121 <latexrelease>\IncludeInRelease{0000/00/00}%
122 <latexrelease>                {\updefault}{font defaults change}%
123 <latexrelease>
124 <latexrelease>\renewcommand\updefault{n}
125 <latexrelease>\renewcommand\bfdefault{bx}
126 <latexrelease>
127 <latexrelease>\let\bfdefault@previous\undefined
128 <latexrelease>\let\mddefault@previous\undefined
129 <latexrelease>\EndIncludeInRelease
130 <*text>

```

`\familydefault` Finally we have the hooks that describe the behaviour of the `\normalfont` command.
`\seriesdefault` To stay portable, the definition of `\encodingdefault` should *not* be changed and should
`\shapedefault` match the setting above for `\fontencoding`. All other values can be set according to
your taste.

```

131 \newcommand\familydefault{\rmdefault}
132 \newcommand\seriesdefault{\mddefault}

```

In previous releases `\shapedefault` pointed to `\updefault` which resolved to `n`, but these days that is no longer the case (and `up` is wrong when you want to do a reset. So we now use `n` explicitly.

```

133 \newcommand\shapedefault{n}

```

(End of definition for `\familydefault`, `\seriesdefault`, and `\shapedefault`.)

This finishes the low-level setup in `fonttext.ltx`.

```

134 </text>

```

6 The fontmath.ltx file

The identification is done earlier on with a `\ProvidesFile` declaration.

```

135 <*math>
136 \typeout{=== Don't modify this file, use a .cfg file instead ===^^J}

```

6.1 The font encodings used

```

137 \DeclareFontEncoding{OML}{}{}
138 \DeclareFontEncoding{OMS}{}{}
139 \DeclareFontEncoding{OMX}{}{}

```

Finally a declaration for U encoding which serves for all fonts that do not fit standard encodings. For math this sets up `\noaccents@` providing for AMS- \LaTeX . This macro is used therein to handle accented characters if they are not supported by the font. In other words, if fonts with U encoding are used in math, all accents (like from `\breve`) are obtained from some other font that has them.

```

140 \DeclareFontEncoding{U}{}{\noaccents@}

```

The encodings for math are next:

```

141 \DeclareFontSubstitution{OML}{cmm}{m}{it}
142 \DeclareFontSubstitution{OMS}{cmsy}{m}{n}

```

```

143 \DeclareFontSubstitution{OMX}{cmex}{m}{n}
144 \DeclareFontSubstitution{U}{cmr}{m}{n}
145 \begingroup
146 \nfss@catcodes
147 \input {omlcmm.fd}
148 \input {omscmsy.fd}
149 \input {omxcmx.fd}
150 \input {ucmr.fd}
151 \endgroup

```

6.1.1 Symbolfont and Alphabet declarations

We now define the basic symbol fonts used by L^AT_EX. These four symbol fonts must be defined by this file.

It is possible to make the symbol fonts point to other external fonts without losing the ability to process documents written at other sites, as long as one defines the same symbol font names with the same encodings, e.g. `operators` with `OT1` etc. If other encodings are used documents become non-portable. Such a change should therefore be done in a package file.

```

152 \DeclareSymbolFont{operators} {OT1}{cmr} {m}{n}
153 \DeclareSymbolFont{letters} {OML}{cmm} {m}{it}
154 \DeclareSymbolFont{symbols} {OMS}{cmsy}{m}{n}
155 \DeclareSymbolFont{largesymbols}{OMX}{cmex}{m}{n}
156 \SetSymbolFont{operators}{bold}{OT1}{cmr} {bx}{n}
157 \SetSymbolFont{letters} {bold}{OML}{cmm} {b}{it}
158 \SetSymbolFont{symbols} {bold}{OMS}{cmsy}{b}{n}

```

Below are the seven math alphabets which are defined by NFSS. Again they must be defined by this file. However, as before you can change the fonts used without losing portability, but you should be careful when changing the encoding since that may make documents come out wrong.

```

159 \DeclareSymbolFontAlphabet{\mathrm} {operators}
160 \DeclareSymbolFontAlphabet{\mathnormal}{letters}
161 \DeclareSymbolFontAlphabet{\mathcal} {symbols}
162 \DeclareMathAlphabet {\mathbf}{OT1}{cmr}{bx}{n}
163 \DeclareMathAlphabet {\mathsf}{OT1}{cmss}{m}{n}
164 \DeclareMathAlphabet {\mathit}{OT1}{cmr}{m}{it}
165 \DeclareMathAlphabet {\mathtt}{OT1}{cmtt}{m}{n}

```

Given the currently available fonts we cannot bold-en `\mathbf` and `\mathtt` but in principle one could use ‘ultra bold’ or something. The alphabets defined via `\DeclareSymbolFontAlphabet` will change automatically in a new math version if the corresponding symbol font changes.

```

166 \SetMathAlphabet\mathsf{bold}{OT1}{cmss}{bx}{n}
167 \SetMathAlphabet\mathit{bold}{OT1}{cmr}{bx}{it}

```

6.2 Math font sizes

The declarations below declare the text, script and scriptscript size to be used for each text font size.

All occurrences of sizes longer than a single character are replaced with the macro name that holds them, saving a number of tokens (but losing a bit of speed, so this may not stay this way).

```

168 \DeclareMathSizes{5}{5}{5}{5}
169 \DeclareMathSizes{6}{6}{5}{5}
170 \DeclareMathSizes{7}{7}{5}{5}
171 \DeclareMathSizes{8}{8}{6}{5}
172 \DeclareMathSizes{9}{9}{6}{5}
173 \DeclareMathSizes{\@xpt}{\@xpt}{7}{5}
174 \DeclareMathSizes{\@xipt}{\@xipt}{8}{6}
175 \DeclareMathSizes{\@xiipt}{\@xiipt}{8}{6}
176 \DeclareMathSizes{\@xivpt}{\@xivpt}{\@xpt}{7}
177 \DeclareMathSizes{\@xvipt}{\@xvipt}{\@xiipt}{\@xpt}
178 \DeclareMathSizes{\@xxpt}{\@xxpt}{\@xivpt}{\@xiipt}
179 \DeclareMathSizes{\@xxvpt}{\@xxvpt}{\@xxpt}{\@xvipt}

```

6.3 The math symbol assignments

We start by setting up math codes for most of the characters typed in directly from the keyboard. Most of them are normally already setup up in the same way by `IniTeX`. However, we repeat them here to have a complete setup which can be exchanged with another if desired.

6.3.1 The letters

```

180 \DeclareMathSymbol{a}{\mathalpha}{letters}{'a}
181 \DeclareMathSymbol{b}{\mathalpha}{letters}{'b}
182 \DeclareMathSymbol{c}{\mathalpha}{letters}{'c}
183 \DeclareMathSymbol{d}{\mathalpha}{letters}{'d}
184 \DeclareMathSymbol{e}{\mathalpha}{letters}{'e}
185 \DeclareMathSymbol{f}{\mathalpha}{letters}{'f}
186 \DeclareMathSymbol{g}{\mathalpha}{letters}{'g}
187 \DeclareMathSymbol{h}{\mathalpha}{letters}{'h}
188 \DeclareMathSymbol{i}{\mathalpha}{letters}{'i}
189 \DeclareMathSymbol{j}{\mathalpha}{letters}{'j}
190 \DeclareMathSymbol{k}{\mathalpha}{letters}{'k}
191 \DeclareMathSymbol{l}{\mathalpha}{letters}{'l}
192 \DeclareMathSymbol{m}{\mathalpha}{letters}{'m}
193 \DeclareMathSymbol{n}{\mathalpha}{letters}{'n}
194 \DeclareMathSymbol{o}{\mathalpha}{letters}{'o}
195 \DeclareMathSymbol{p}{\mathalpha}{letters}{'p}
196 \DeclareMathSymbol{q}{\mathalpha}{letters}{'q}
197 \DeclareMathSymbol{r}{\mathalpha}{letters}{'r}
198 \DeclareMathSymbol{s}{\mathalpha}{letters}{'s}
199 \DeclareMathSymbol{t}{\mathalpha}{letters}{'t}
200 \DeclareMathSymbol{u}{\mathalpha}{letters}{'u}
201 \DeclareMathSymbol{v}{\mathalpha}{letters}{'v}
202 \DeclareMathSymbol{w}{\mathalpha}{letters}{'w}
203 \DeclareMathSymbol{x}{\mathalpha}{letters}{'x}
204 \DeclareMathSymbol{y}{\mathalpha}{letters}{'y}
205 \DeclareMathSymbol{z}{\mathalpha}{letters}{'z}

206 \DeclareMathSymbol{A}{\mathalpha}{letters}{'A}
207 \DeclareMathSymbol{B}{\mathalpha}{letters}{'B}
208 \DeclareMathSymbol{C}{\mathalpha}{letters}{'C}
209 \DeclareMathSymbol{D}{\mathalpha}{letters}{'D}
210 \DeclareMathSymbol{E}{\mathalpha}{letters}{'E}

```

```

211 \DeclareMathSymbol{F}{\mathalpha}{letters}{'F}
212 \DeclareMathSymbol{G}{\mathalpha}{letters}{'G}
213 \DeclareMathSymbol{H}{\mathalpha}{letters}{'H}
214 \DeclareMathSymbol{I}{\mathalpha}{letters}{'I}
215 \DeclareMathSymbol{J}{\mathalpha}{letters}{'J}
216 \DeclareMathSymbol{K}{\mathalpha}{letters}{'K}
217 \DeclareMathSymbol{L}{\mathalpha}{letters}{'L}
218 \DeclareMathSymbol{M}{\mathalpha}{letters}{'M}
219 \DeclareMathSymbol{N}{\mathalpha}{letters}{'N}
220 \DeclareMathSymbol{O}{\mathalpha}{letters}{'O}
221 \DeclareMathSymbol{P}{\mathalpha}{letters}{'P}
222 \DeclareMathSymbol{Q}{\mathalpha}{letters}{'Q}
223 \DeclareMathSymbol{R}{\mathalpha}{letters}{'R}
224 \DeclareMathSymbol{S}{\mathalpha}{letters}{'S}
225 \DeclareMathSymbol{T}{\mathalpha}{letters}{'T}
226 \DeclareMathSymbol{U}{\mathalpha}{letters}{'U}
227 \DeclareMathSymbol{V}{\mathalpha}{letters}{'V}
228 \DeclareMathSymbol{W}{\mathalpha}{letters}{'W}
229 \DeclareMathSymbol{X}{\mathalpha}{letters}{'X}
230 \DeclareMathSymbol{Y}{\mathalpha}{letters}{'Y}
231 \DeclareMathSymbol{Z}{\mathalpha}{letters}{'Z}

```

6.3.2 The digits

```

232 \DeclareMathSymbol{0}{\mathalpha}{operators}{'0}
233 \DeclareMathSymbol{1}{\mathalpha}{operators}{'1}
234 \DeclareMathSymbol{2}{\mathalpha}{operators}{'2}
235 \DeclareMathSymbol{3}{\mathalpha}{operators}{'3}
236 \DeclareMathSymbol{4}{\mathalpha}{operators}{'4}
237 \DeclareMathSymbol{5}{\mathalpha}{operators}{'5}
238 \DeclareMathSymbol{6}{\mathalpha}{operators}{'6}
239 \DeclareMathSymbol{7}{\mathalpha}{operators}{'7}
240 \DeclareMathSymbol{8}{\mathalpha}{operators}{'8}
241 \DeclareMathSymbol{9}{\mathalpha}{operators}{'9}

```

6.3.3 Punctuation, brace, etc. keys

```

242 \DeclareMathSymbol{!}{\mathclose}{operators}{"21}
243 \DeclareMathSymbol{*}{\mathbin}{symbols}{"03} % \ast
244 \DeclareMathSymbol{+}{\mathbin}{operators}{"2B}
245 \DeclareMathSymbol{,}{\mathpunct}{letters}{"3B}
246 \DeclareMathSymbol{-}{\mathbin}{symbols}{"00}
247 \DeclareMathSymbol{.}{\mathord}{letters}{"3A}
248 \DeclareMathSymbol{:}{\mathrel}{operators}{"3A}
249 \DeclareMathSymbol{;}{\mathpunct}{operators}{"3B}
250 \DeclareMathSymbol{=}{\mathrel}{operators}{"3D}
251 \DeclareMathSymbol{?}{\mathclose}{operators}{"3F}

```

The following symbols are defined as delimiters below which automatically defines them as math symbols.

```

252 %\DeclareMathSymbol{(}{\mathopen}{operators}{"28}
253 %\DeclareMathSymbol{)}{\mathclose}{operators}{"29}
254 %\DeclareMathSymbol{/}{\mathord}{letters}{"3D}
255 %\DeclareMathSymbol{[}{\mathopen}{operators}{"5B}
256 %\DeclareMathSymbol{]}{\mathclose}{operators}{"5D}
257 %\DeclareMathSymbol{|}{\mathord}{symbols}{"6A}

```

```

258 %\DeclareMathSymbol{<}{\mathrel}{letters}{"3C}
259 %\DeclareMathSymbol{>}{\mathrel}{letters}{"3E}

```

Should all of the following being activated by default? Probably not.

```

260 %\DeclareMathSymbol{\{}{\mathopen}{symbols}{"66}
261 %\DeclareMathSymbol{\}}{\mathclose}{symbols}{"67}
262 %\DeclareMathSymbol{\}\}{\mathord}{symbols}{"6E} % \backslash
263 \mathcode'\ =8000 % \space
264 \mathcode\' =8000 % ^\prime
265 \mathcode\'_ =8000 % \_

```

6.3.4 Delimitercodes for characters

[to be completed]

Finally, iniTeX sets all $\backslash\delcode$ values to -1, except $\backslash\delcode' = 0$

```

266 \DeclareMathDelimiter{()}{\mathopen}{operators}{"28}{largesymbols}{"00}
267 \DeclareMathDelimiter{)}{\mathclose}{operators}{"29}{largesymbols}{"01}
268 \DeclareMathDelimiter{[]}{\mathopen}{operators}{"5B}{largesymbols}{"02}
269 \DeclareMathDelimiter{[]}{\mathclose}{operators}{"5D}{largesymbols}{"03}

```

The next two are considered to be relations when not used in the context of a delimiter! And worse, they do even represent different glyphs when being used as delimiter and not as delimiter. This is a user level syntax inherited from plain TeX . Therefore we explicitly redefine the math symbol definitions for these symbols afterwards.

```

270 \DeclareMathDelimiter{<}{\mathopen}{symbols}{"68}{largesymbols}{"0A}
271 \DeclareMathDelimiter{>}{\mathclose}{symbols}{"69}{largesymbols}{"0B}
272 \DeclareMathSymbol{<}{\mathrel}{letters}{"3C}
273 \DeclareMathSymbol{>}{\mathrel}{letters}{"3E}

```

And here is another case where the non-delimiter version produces a glyph different from the delimiter version.

```

274 \DeclareMathDelimiter{/}{\mathord}{operators}{"2F}{largesymbols}{"0E}
275 \DeclareMathSymbol{/}{\mathord}{letters}{"3D}

276 \DeclareMathDelimiter{|}{\mathord}{symbols}{"6A}{largesymbols}{"0C}
277 \xexpandafter\DeclareMathDelimiter\@backslashchar
278 \mathord}{symbols}{"6E}{largesymbols}{"0F}

```

N.B. { and } should NOT get delcodes; otherwise parameter grouping fails!

6.4 Symbols accessed via control sequences

6.4.1 Greek letters

```

279 \DeclareMathSymbol{\alpha}{\mathord}{letters}{"0B}
280 \DeclareMathSymbol{\beta}{\mathord}{letters}{"0C}
281 \DeclareMathSymbol{\gamma}{\mathord}{letters}{"0D}
282 \DeclareMathSymbol{\delta}{\mathord}{letters}{"0E}
283 \DeclareMathSymbol{\epsilon}{\mathord}{letters}{"0F}
284 \DeclareMathSymbol{\zeta}{\mathord}{letters}{"10}
285 \DeclareMathSymbol{\eta}{\mathord}{letters}{"11}
286 \DeclareMathSymbol{\theta}{\mathord}{letters}{"12}
287 \DeclareMathSymbol{\iota}{\mathord}{letters}{"13}
288 \DeclareMathSymbol{\kappa}{\mathord}{letters}{"14}
289 \DeclareMathSymbol{\lambda}{\mathord}{letters}{"15}
290 \DeclareMathSymbol{\mu}{\mathord}{letters}{"16}
291 \DeclareMathSymbol{\nu}{\mathord}{letters}{"17}

```

```

292 \DeclareMathSymbol{\xi}{\mathord}{letters}{"18}
293 \DeclareMathSymbol{\pi}{\mathord}{letters}{"19}
294 \DeclareMathSymbol{\rho}{\mathord}{letters}{"1A}
295 \DeclareMathSymbol{\sigma}{\mathord}{letters}{"1B}
296 \DeclareMathSymbol{\tau}{\mathord}{letters}{"1C}
297 \DeclareMathSymbol{\upsilon}{\mathord}{letters}{"1D}
298 \DeclareMathSymbol{\phi}{\mathord}{letters}{"1E}
299 \DeclareMathSymbol{\chi}{\mathord}{letters}{"1F}
300 \DeclareMathSymbol{\psi}{\mathord}{letters}{"20}
301 \DeclareMathSymbol{\omega}{\mathord}{letters}{"21}
302 \DeclareMathSymbol{\varepsilon}{\mathord}{letters}{"22}
303 \DeclareMathSymbol{\vartheta}{\mathord}{letters}{"23}
304 \DeclareMathSymbol{\varpi}{\mathord}{letters}{"24}
305 \DeclareMathSymbol{\varrho}{\mathord}{letters}{"25}
306 \DeclareMathSymbol{\varsigma}{\mathord}{letters}{"26}
307 \DeclareMathSymbol{\varphi}{\mathord}{letters}{"27}
308 \DeclareMathSymbol{\Gamma}{\mathalpha}{operators}{"00}
309 \DeclareMathSymbol{\Delta}{\mathalpha}{operators}{"01}
310 \DeclareMathSymbol{\Theta}{\mathalpha}{operators}{"02}
311 \DeclareMathSymbol{\Lambda}{\mathalpha}{operators}{"03}
312 \DeclareMathSymbol{\Xi}{\mathalpha}{operators}{"04}
313 \DeclareMathSymbol{\Pi}{\mathalpha}{operators}{"05}
314 \DeclareMathSymbol{\Sigma}{\mathalpha}{operators}{"06}
315 \DeclareMathSymbol{\Upsilon}{\mathalpha}{operators}{"07}
316 \DeclareMathSymbol{\Phi}{\mathalpha}{operators}{"08}
317 \DeclareMathSymbol{\Psi}{\mathalpha}{operators}{"09}
318 \DeclareMathSymbol{\Omega}{\mathalpha}{operators}{"0A}

```

6.4.2 Ordinary symbols

```

319 \DeclareMathSymbol{\aleph}{\mathord}{symbols}{"40}
320 \DeclareMathSymbol{\imath}{\mathord}{letters}{"7B}
321 \DeclareMathSymbol{\jmath}{\mathord}{letters}{"7C}
322 \DeclareMathSymbol{\ell}{\mathord}{letters}{"60}
323 \DeclareMathSymbol{\wp}{\mathord}{letters}{"7D}
324 \DeclareMathSymbol{\Re}{\mathord}{symbols}{"3C}
325 \DeclareMathSymbol{\Im}{\mathord}{symbols}{"3D}
326 \DeclareMathSymbol{\partial}{\mathord}{letters}{"40}
327 \DeclareMathSymbol{\infty}{\mathord}{symbols}{"31}
328 \DeclareMathSymbol{\prime}{\mathord}{symbols}{"30}
329 \DeclareMathSymbol{\emptyset}{\mathord}{symbols}{"3B}
330 \DeclareMathSymbol{\nabla}{\mathord}{symbols}{"72}
331 \DeclareMathSymbol{\top}{\mathord}{symbols}{"3E}
332 \DeclareMathSymbol{\bot}{\mathord}{symbols}{"3F}
333 \DeclareMathSymbol{\triangle}{\mathord}{symbols}{"34}
334 \DeclareMathSymbol{\forall}{\mathord}{symbols}{"38}
335 \DeclareMathSymbol{\exists}{\mathord}{symbols}{"39}
336 \DeclareMathSymbol{\neg}{\mathord}{symbols}{"3A}

```

Alias:

```

337 % \let\not=\neg
338 \DeclareMathSymbol{\not}{\mathord}{symbols}{"3A}
339 \DeclareMathSymbol{\flat}{\mathord}{letters}{"5B}
340 \DeclareMathSymbol{\natural}{\mathord}{letters}{"5C}
341 \DeclareMathSymbol{\sharp}{\mathord}{letters}{"5D}

```



```

342 \DeclareMathSymbol{\clubsuit}{\mathord}{symbols}{"7C}
343 \DeclareMathSymbol{\diamondsuit}{\mathord}{symbols}{"7D}
344 \DeclareMathSymbol{\heartsuit}{\mathord}{symbols}{"7E}
345 \DeclareMathSymbol{\spadesuit}{\mathord}{symbols}{"7F}

346 \DeclareRobustCommand\hbar{{\mathchar'26\mkern-9muh}}
347 \DeclareRobustCommand\surd{{\mathchar"1270}}
348 \DeclareRobustCommand\angle{{\vbox{\ialign{$\m@th\scriptstyle##$\crrc
349     \not\mathrel{\mkern14mu}\crrc
350     \noalign{\nointerlineskip}
351     \mkern2.5mu\leaders\hrule \@height.34pt\hfill\mkern2.5mu\crrc}}}}

```

6.4.3 Large Operators

```

352 \DeclareMathSymbol{\coprod}{\mathop}{largesymbols}{"60}
353 \DeclareMathSymbol{\bigvee}{\mathop}{largesymbols}{"57}
354 \DeclareMathSymbol{\bigwedge}{\mathop}{largesymbols}{"56}
355 \DeclareMathSymbol{\biguplus}{\mathop}{largesymbols}{"55}
356 \DeclareMathSymbol{\bigcap}{\mathop}{largesymbols}{"54}
357 \DeclareMathSymbol{\bigcup}{\mathop}{largesymbols}{"53}
358 \DeclareMathSymbol{\intop}{\mathop}{largesymbols}{"52}
359     \DeclareRobustCommand\int{\intop\nolimits}
360 \DeclareMathSymbol{\prod}{\mathop}{largesymbols}{"51}
361 \DeclareMathSymbol{\sum}{\mathop}{largesymbols}{"50}
362 \DeclareMathSymbol{\bigotimes}{\mathop}{largesymbols}{"4E}
363 \DeclareMathSymbol{\bigoplus}{\mathop}{largesymbols}{"4C}
364 \DeclareMathSymbol{\bigodot}{\mathop}{largesymbols}{"4A}
365 \DeclareMathSymbol{\ointop}{\mathop}{largesymbols}{"48}
366     \DeclareRobustCommand\oint{\ointop\nolimits}
367 \DeclareMathSymbol{\bigsqcup}{\mathop}{largesymbols}{"46}
368 \DeclareMathSymbol{\smallint}{\mathop}{symbols}{"73}

```

6.4.4 Binary symbols

```

369 \DeclareMathSymbol{\triangleleft}{\mathbin}{letters}{"2F}
370 \DeclareMathSymbol{\triangleright}{\mathbin}{letters}{"2E}
371 \DeclareMathSymbol{\bigtriangleup}{\mathbin}{symbols}{"34}
372 \DeclareMathSymbol{\bigtriangledown}{\mathbin}{symbols}{"35}

```

Alias:

```

373 % \let \varbigtriangledown \bigtriangledown
374 % \let \varbigtriangleup \bigtriangleup
375 \DeclareMathSymbol{\varbigtriangleup}{\mathbin}{symbols}{"34}
376 \DeclareMathSymbol{\varbigtriangledown}{\mathbin}{symbols}{"35}

```

These last two synonyms are needed because the `stmaryrd` package redefines them as Operators.

```

377 \DeclareMathSymbol{\wedge}{\mathbin}{symbols}{"5E}
378 \DeclareMathSymbol{\vee}{\mathbin}{symbols}{"5F}

```

Alias:

```

379 % \let \land = \wedge
380 % \let \lor = \vee
381 \DeclareMathSymbol{\land}{\mathbin}{symbols}{"5E}
382 \DeclareMathSymbol{\lor}{\mathbin}{symbols}{"5F}
383 \DeclareMathSymbol{\cap}{\mathbin}{symbols}{"5C}
384 \DeclareMathSymbol{\cup}{\mathbin}{symbols}{"5B}
385 \DeclareMathSymbol{\ddagger}{\mathbin}{symbols}{"7A}

```

```

386 \DeclareMathSymbol{\dagger}{\mathbin}{symbols}{"79}
387 \DeclareMathSymbol{\sqcap}{\mathbin}{symbols}{"75}
388 \DeclareMathSymbol{\sqcup}{\mathbin}{symbols}{"74}
389 \DeclareMathSymbol{\uplus}{\mathbin}{symbols}{"5D}
390 \DeclareMathSymbol{\amalg}{\mathbin}{symbols}{"71}
391 \DeclareMathSymbol{\diamond}{\mathbin}{symbols}{"05}
392 \DeclareMathSymbol{\bullet}{\mathbin}{symbols}{"0F}
393 \DeclareMathSymbol{\wr}{\mathbin}{symbols}{"6F}
394 \DeclareMathSymbol{\div}{\mathbin}{symbols}{"04}
395 \DeclareMathSymbol{\odot}{\mathbin}{symbols}{"0C}
396 \DeclareMathSymbol{\oslash}{\mathbin}{symbols}{"0B}
397 \DeclareMathSymbol{\otimes}{\mathbin}{symbols}{"0A}
398 \DeclareMathSymbol{\ominus}{\mathbin}{symbols}{"09}
399 \DeclareMathSymbol{\oplus}{\mathbin}{symbols}{"08}
400 \DeclareMathSymbol{\mp}{\mathbin}{symbols}{"07}
401 \DeclareMathSymbol{\pm}{\mathbin}{symbols}{"06}
402 \DeclareMathSymbol{\circ}{\mathbin}{symbols}{"0E}
403 \DeclareMathSymbol{\bigcirc}{\mathbin}{symbols}{"0D}
404 \DeclareMathSymbol{\setminus}{\mathbin}{symbols}{"6E}
405 \DeclareMathSymbol{\cdot}{\mathbin}{symbols}{"01}
406 \DeclareMathSymbol{\ast}{\mathbin}{symbols}{"03}
407 \DeclareMathSymbol{\times}{\mathbin}{symbols}{"02}
408 \DeclareMathSymbol{\star}{\mathbin}{letters}{"3F}

```

6.4.5 Relations

```

409 \DeclareMathSymbol{\propto}{\mathrel}{symbols}{"2F}
410 \DeclareMathSymbol{\sqsubseteq}{\mathrel}{symbols}{"76}
411 \DeclareMathSymbol{\sqsupseteq}{\mathrel}{symbols}{"77}
412 \DeclareMathSymbol{\parallel}{\mathrel}{symbols}{"6B}
413 \DeclareMathSymbol{\mid}{\mathrel}{symbols}{"6A}
414 \DeclareMathSymbol{\dashv}{\mathrel}{symbols}{"61}
415 \DeclareMathSymbol{\vdash}{\mathrel}{symbols}{"60}
416 \DeclareMathSymbol{\nearrow}{\mathrel}{symbols}{"25}
417 \DeclareMathSymbol{\searrow}{\mathrel}{symbols}{"26}
418 \DeclareMathSymbol{\nrightarrow}{\mathrel}{symbols}{"2D}
419 \DeclareMathSymbol{\swarrow}{\mathrel}{symbols}{"2E}
420 \DeclareMathSymbol{\Leftrightarrow}{\mathrel}{symbols}{"2C}
421 \DeclareMathSymbol{\Leftarrow}{\mathrel}{symbols}{"28}
422 \DeclareMathSymbol{\Rightarrow}{\mathrel}{symbols}{"29}
423 \DeclareRobustCommand\neq{\not=}

```

As `\neq` is robust we should not use `\let` to define `\ne` as then it would change if `\neq` changes.

```

424 \DeclareRobustCommand\ne{\not=}

```

It would ok to use `\let` for those declared by `\DeclareMathSymbol` but for a cleaner interface we avoid it always (just in case the internals change).

```

425 \DeclareMathSymbol{\leq}{\mathrel}{symbols}{"14}
426 \DeclareMathSymbol{\geq}{\mathrel}{symbols}{"15}

```

Alias:

```

427 % \let\le=\leq
428 % \let\ge=\geq
429 \DeclareMathSymbol{\le}{\mathrel}{symbols}{"14}
430 \DeclareMathSymbol{\ge}{\mathrel}{symbols}{"15}

```

```

431 \DeclareMathSymbol{\succ}{\mathrel}{symbols}{"1F}
432 \DeclareMathSymbol{\prec}{\mathrel}{symbols}{"1E}
433 \DeclareMathSymbol{\approx}{\mathrel}{symbols}{"19}
434 \DeclareMathSymbol{\succeq}{\mathrel}{symbols}{"17}
435 \DeclareMathSymbol{\preceq}{\mathrel}{symbols}{"16}
436 \DeclareMathSymbol{\supset}{\mathrel}{symbols}{"1B}
437 \DeclareMathSymbol{\subset}{\mathrel}{symbols}{"1A}
438 \DeclareMathSymbol{\supseteq}{\mathrel}{symbols}{"13}
439 \DeclareMathSymbol{\subseteq}{\mathrel}{symbols}{"12}
440 \DeclareMathSymbol{\in}{\mathrel}{symbols}{"32}
441 \DeclareMathSymbol{\ni}{\mathrel}{symbols}{"33}

```

Alias:

```

442 % \let\owns=\ni
443 \DeclareMathSymbol{\owns}{\mathrel}{symbols}{"33}
444 \DeclareMathSymbol{\gg}{\mathrel}{symbols}{"1D}
445 \DeclareMathSymbol{\ll}{\mathrel}{symbols}{"1C}
446 \DeclareMathSymbol{\not}{\mathrel}{symbols}{"36}
447 \DeclareMathSymbol{\leftrightarrow}{\mathrel}{symbols}{"24}
448 \DeclareMathSymbol{\leftarrow}{\mathrel}{symbols}{"20}
449 \DeclareMathSymbol{\rightarrow}{\mathrel}{symbols}{"21}

```

Alias:

```

450 % \let\gets=\leftarrow
451 % \let\to=\rightarrow
452 \DeclareMathSymbol{\gets}{\mathrel}{symbols}{"20}
453 \DeclareMathSymbol{\to}{\mathrel}{symbols}{"21}
454 \DeclareMathSymbol{\mapstochar}{\mathrel}{symbols}{"37}
455 \DeclareRobustCommand\mapsto{\mapstochar\rightarrow}
456 \DeclareMathSymbol{\sim}{\mathrel}{symbols}{"18}
457 \DeclareMathSymbol{\simeq}{\mathrel}{symbols}{"27}
458 \DeclareMathSymbol{\perp}{\mathrel}{symbols}{"3F}
459 \DeclareMathSymbol{\equiv}{\mathrel}{symbols}{"11}
460 \DeclareMathSymbol{\asymp}{\mathrel}{symbols}{"10}
461 \DeclareMathSymbol{\smile}{\mathrel}{letters}{"5E}
462 \DeclareMathSymbol{\frown}{\mathrel}{letters}{"5F}
463 \DeclareMathSymbol{\leftharpoonup}{\mathrel}{letters}{"28}
464 \DeclareMathSymbol{\leftharpoondown}{\mathrel}{letters}{"29}
465 \DeclareMathSymbol{\rightharpoonup}{\mathrel}{letters}{"2A}
466 \DeclareMathSymbol{\rightharpoondown}{\mathrel}{letters}{"2B}

```

Here cometh much profligate robustification of math constructs. Warning: some of these commands may become non-robust if an AMS package is loaded.

Further potential problems: some math font packages may make unfortunate assumptions about some of these definitions that are not true of the robust versions we need.

```

467 \DeclareRobustCommand
468 \cong{\mathrel{\mathpalette\@vereq\sim}} % congruence sign
469 \def\@vereq#1#2{\lower.5p@\vbox{\lineskiplimit\maxdimen\lineskip-.5p@
470 \ialign{${\m@th#1\hfil##\hfil$\crcr#2\crcr=\crcr}}}}
471 \DeclareRobustCommand
472 \notin{\mathrel{\m@th\mathpalette\c@ncel\in}}
473 \def\c@ncel#1#2{\m@th\oalign{${\hfil#1\mkern1mu/\hfil$\crcr$#1#2$}}
474 \DeclareRobustCommand
475 \rightleftarpoons{\mathrel{\mathpalette\righth@{}}}

```

```

476 \def\rhlh@#1{\vcenter{\m@th\hbox{\ooalign{\raise2pt
477 \hbox{#1\rightarpoonup$}\crr
478 #1\leftarpoondown$}}}}
479 \DeclareRobustCommand
480 \doteq{\buildrel\textstyle.\over=}

```

6.4.6 Arrows

```

481 \DeclareRobustCommand
482 \joinrel{\mathrel{\mkern-3mu}}
483 \DeclareRobustCommand
484 \relbar{\mathrel{\smash-}} % \smash, because -
485 % has the same height as +

```

In contrast to `plain.tex` `\Relbar` got braces around the equal sign to guard against it being “math active” expanding to `\futurelet...`. This might be the case when packages are implementing shorthands for math, e.g. `=>` meaning `\Rightarrow` etc. It would actually be better not to use `=` in such definitions but instead define something like `\mathequalsign` and use this. However we can’t do this now as it would break other math layouts where characters are in different places (since those wouldn’t know about the need for a new command name).

```

486 \DeclareRobustCommand
487 \Relbar{\mathrel{=}}
488 \DeclareMathSymbol{\lhook}{\mathrel}{letters}{"2C}
489 \DeclareRobustCommand\hookrightarrow{\lhook\joinrel\rightarrow}
490 \DeclareMathSymbol{\rhook}{\mathrel}{letters}{"2D}
491 \DeclareRobustCommand\hookleftarrow{\leftarrow\joinrel\rhook}
492 \DeclareRobustCommand
493 \bowtie{\mathrel\triangleright\joinrel\mathrel\triangleleft}
494 \DeclareRobustCommand
495 \models{\mathrel{|}\joinrel\Relbar}
496 \DeclareRobustCommand
497 \Longrightarrow{\Relbar\joinrel\rightarrow}

```

LaTeX Change: `\longrightarrow` and `\longleftarrow` redefined to make them robust.

```

498 \DeclareRobustCommand\longrightarrow
499 {\relbar\joinrel\rightarrow}
500 \DeclareRobustCommand\longleftarrow
501 {\leftarrow\joinrel\relbar}
502 \DeclareRobustCommand
503 \Longleftarrow{\Leftarrow\joinrel\Relbar}
504 \DeclareRobustCommand
505 \longmapsto{\mapstochar\longrightarrow}
506 \DeclareRobustCommand
507 \longlefttrightarrow{\leftarrow\joinrel\rightarrow}
508 \DeclareRobustCommand
509 \Longlefttrightarrow{\Leftarrow\joinrel\rightarrow}
510 \DeclareRobustCommand
511 \iff{\;\Longlefttrightarrow\;}

```

6.4.7 Punctuation symbols

```

512 \DeclareMathSymbol{\ldotp}{\mathpunct}{letters}{"3A}
513 \DeclareMathSymbol{\cdotp}{\mathpunct}{symbols}{"01}
514 \DeclareMathSymbol{\colon}{\mathpunct}{operators}{"3A}

```

This is commented out, since `\ldots` is now defined in `ltoutenc.dtx`.

```

515 %\def\@ldots{\mathinner{\ldotp\ldotp\ldotp}}
516 %\DeclareRobustCommand\ldots
517 %      {\relax\ifmmode\@ldots\else\mbox{$\m@th\@ldots\,$}\fi}
518 \DeclareRobustCommand
519   \cdots{\mathinner{\cdotp\cdotp\cdotp}}
520 \DeclareRobustCommand
521   \vdots{\vbox{\baselineskip4\p@ \lineskiplimit\z@
522     \kern6\p@\hbox{.}\hbox{.}\hbox{.}}}
523 \DeclareRobustCommand
524   \ddots{\mathinner{\mkern1mu\raise7\p@
525     \vbox{\kern7\p@\hbox{.}}\mkern2mu
526     \raise4\p@\hbox{.}\mkern2mu\raise\p@\hbox{.}\mkern1mu}}

```

6.4.8 Math accents

```

527 \DeclareMathAccent{\acute}{\mathalpha}{operators}{"13}
528 \DeclareMathAccent{\grave}{\mathalpha}{operators}{"12}
529 \DeclareMathAccent{\ddot}{\mathalpha}{operators}{"7F}
530 \DeclareMathAccent{\tilde}{\mathalpha}{operators}{"7E}
531 \DeclareMathAccent{\bar}{\mathalpha}{operators}{"16}
532 \DeclareMathAccent{\breve}{\mathalpha}{operators}{"15}
533 \DeclareMathAccent{\check}{\mathalpha}{operators}{"14}
534 \DeclareMathAccent{\hat}{\mathalpha}{operators}{"5E}
535 \DeclareMathAccent{\vec}{\mathord}{letters}{"7E}
536 \DeclareMathAccent{\dot}{\mathalpha}{operators}{"5F}
537 \DeclareMathAccent{\widetilde}{\mathord}{largesymbols}{"65}
538 \DeclareMathAccent{\widehat}{\mathord}{largesymbols}{"62}

```

For some reason plain T_EX never bothered to provide a ring accent in math (although it is available in the fonts), but since we got a request for it here we go:

```

539 \DeclareMathAccent{\mathring}{\mathalpha}{operators}{"17}

```

6.4.9 Radicals

```

540 \DeclareMathRadical{\sqrtsign}{symbols}{"70}{largesymbols}{"70}

```

6.4.10 Over and under something, etc

```

541 \DeclareRobustCommand\overrightarrow[1]{\vbox{\m@th\ialign{##\crrc
542   \rightarrowfill\crrc\noalign{\kern-\p@\nointerlineskip}
543   $\hfil\displaystyle{#1}\hfil$\crrc}}}
544 \DeclareRobustCommand\overleftarrow[1]{\vbox{\m@th\ialign{##\crrc
545   \leftarrowfill\crrc\noalign{\kern-\p@\nointerlineskip}%
546   $\hfil\displaystyle{#1}\hfil$\crrc}}}
547 \DeclareRobustCommand\overbrace[1]
548   {\mathop{\vbox{\m@th\ialign{##\crrc\noalign{\kern3\p@}%
549     \downbracefill\crrc\noalign{\kern3\p@\nointerlineskip}%
550     $\hfil\displaystyle{#1}\hfil$\crrc}}}\limits}
551 \DeclareRobustCommand\underbrace[1]{\mathop{\vtop{\m@th\ialign{##\crrc
552   $\hfil\displaystyle{#1}\hfil$\crrc
553   \noalign{\kern3\p@\nointerlineskip}%
554   \upbracefill\crrc\noalign{\kern3\p@}}}\limits}

```

(quite a waste of tokens, IMHO — Frank)

```

555 \DeclareRobustCommand\skew[3]
556   {\muskip\z@#1mu\divide\muskip\z@#2\mkern\muskip\z@
557   #2{\mkern-\muskip\z@#3}\mkern\muskip\z@}\mkern-\muskip\z@}

```

```

558 \DeclareRobustCommand\rightarrowfill{\m@th\smash-\mkern-7mu%
559 \cleaders\hbox{\mkern-2mu\smash-\mkern-2mu$\}\hfill
560 \mkern-7mu\mathord\rightarrow$}
561 \DeclareRobustCommand\leftarrowfill{\m@th\mathord\leftarrow\mkern-7mu%
562 \cleaders\hbox{\mkern-2mu\smash-\mkern-2mu$\}\hfill
563 \mkern-7mu\smash-$}
564 \DeclareMathSymbol{\braceld}{\mathord}{largesymbols}{"7A}
565 \DeclareMathSymbol{\bracerd}{\mathord}{largesymbols}{"7B}
566 \DeclareMathSymbol{\bracelu}{\mathord}{largesymbols}{"7C}
567 \DeclareMathSymbol{\braceru}{\mathord}{largesymbols}{"7D}
568 \DeclareRobustCommand\downbracefill{\m@th \setbox\z@\hbox{\braceld$}%
569 \braceld\leaders\vrule \@height\ht\z@ \@depth\z@\hfill\braceru
570 \bracelu\leaders\vrule \@height\ht\z@ \@depth\z@\hfill\bracerd$}
571 \DeclareRobustCommand\upbracefill{\m@th \setbox\z@\hbox{\braceld$}%
572 \bracelu\leaders\vrule \@height\ht\z@ \@depth\z@\hfill\bracerd
573 \braceld\leaders\vrule \@height\ht\z@ \@depth\z@\hfill\braceru$}

```

6.4.11 Delimiters

```

574 \DeclareMathDelimiter{\lmoustache} % top from (, bottom from )
575 {\mathopen}{largesymbols}{"7A}{largesymbols}{"40}
576 \DeclareMathDelimiter{\rmoustache} % top from ), bottom from (
577 {\mathclose}{largesymbols}{"7B}{largesymbols}{"41}
578 \DeclareMathDelimiter{\arrowvert} % arrow without arrowheads
579 {\mathord}{symbols}{"6A}{largesymbols}{"3C}
580 \DeclareMathDelimiter{\Arrowvert} % double arrow without arrowheads
581 {\mathord}{symbols}{"6B}{largesymbols}{"3D}
582 \DeclareMathDelimiter{\Vert}
583 {\mathord}{symbols}{"6B}{largesymbols}{"0D}

```

`\DeclareMathDelimiter` produces a command that is robust (with an internal macro containing the payload) so we should not use `\let` for making an alias

```

584 %\let\|= \Vert
585 \DeclareMathDelimiter{\|}
586 {\mathord}{symbols}{"6B}{largesymbols}{"0D}
587 \DeclareMathDelimiter{\vert}
588 {\mathord}{symbols}{"6A}{largesymbols}{"0C}
589 \DeclareMathDelimiter{\uparrow}
590 {\mathrel}{symbols}{"22}{largesymbols}{"78}
591 \DeclareMathDelimiter{\downarrow}
592 {\mathrel}{symbols}{"23}{largesymbols}{"79}
593 \DeclareMathDelimiter{\updownarrow}
594 {\mathrel}{symbols}{"6C}{largesymbols}{"3F}
595 \DeclareMathDelimiter{\Uparrow}
596 {\mathrel}{symbols}{"2A}{largesymbols}{"7E}
597 \DeclareMathDelimiter{\Downarrow}
598 {\mathrel}{symbols}{"2B}{largesymbols}{"7F}
599 \DeclareMathDelimiter{\Updownarrow}
600 {\mathrel}{symbols}{"6D}{largesymbols}{"77}
601 \DeclareMathDelimiter{\backslash} % for double coset G\backslash H
602 {\mathord}{symbols}{"6E}{largesymbols}{"0F}
603 \DeclareMathDelimiter{\rangle}
604 {\mathclose}{symbols}{"69}{largesymbols}{"0B}
605 \DeclareMathDelimiter{\langle}
606 {\mathopen}{symbols}{"68}{largesymbols}{"0A}

```

```

607 \DeclareMathDelimiter{\rbrace
608   {\mathclose}{symbols}{"67}{largesymbols}{"09}
609 \DeclareMathDelimiter{\lbrace
610   {\mathopen}{symbols}{"66}{largesymbols}{"08}
611 \DeclareMathDelimiter{\rceil}
612   {\mathclose}{symbols}{"65}{largesymbols}{"07}
613 \DeclareMathDelimiter{\lceil}
614   {\mathopen}{symbols}{"64}{largesymbols}{"06}
615 \DeclareMathDelimiter{\rfloor}
616   {\mathclose}{symbols}{"63}{largesymbols}{"05}
617 \DeclareMathDelimiter{\lfloor}
618   {\mathopen}{symbols}{"62}{largesymbols}{"04}

```

`\lgroup` `\rgroup` `\bracevert` There are three plain \TeX delimiters which are not fully supported by NFSS, since they partly point into a bold `cmr` font. Allocating a full symbol font, just to have three delimiters seems a bit too much given the limited space available. For this reason only the extensible sizes are supported. If this is not desired one can use, without losing portability, define `\mathbf` and `\mathtt` as font symbol alphabet (setting up `cmr/bx/n` and `cmtt/m/n` as symbol fonts first) and modify the delimiter declarations to point with their small variant to those symbol fonts. (This is done in `oldlfont.dtx` so look there for examples.)

```

619 \DeclareMathDelimiter{\lgroup} % extensible ( with sharper tips
620   {\mathopen}{largesymbols}{"3A}{largesymbols}{"3A}
621 \DeclareMathDelimiter{\rgroup} % extensible ) with sharper tips
622   {\mathclose}{largesymbols}{"3B}{largesymbols}{"3B}
623 \DeclareMathDelimiter{\bracevert} % the vertical bar that extends braces
624   {\mathord}{largesymbols}{"3E}{largesymbols}{"3E}

```

(End of definition for `\lgroup`, `\rgroup`, and `\bracevert`.)

6.5 Math versions of text commands

The `\mathunderscore` here is really a text definition, so it has been put back into `ltoutenc.dtx` (by Chris, 30/04/97) and should be removed from here.

These symbols are the math versions of text commands such as `\P`, `\$`, etc.

```

\mathparagraph These math symbols are not in plain  $\TeX$ .
\mathsection 625 \DeclareMathSymbol{\mathparagraph}{\mathord}{symbols}{"7B}
\mathdollar 626 \DeclareMathSymbol{\mathsection}{\mathord}{symbols}{"78}
\mathsterling 627 \DeclareMathSymbol{\mathdollar}{\mathord}{operators}{"24}
\mathunderscore 628 \DeclareRobustCommand\mathsterling{\mathit{\mathchar"7024}}
629 \DeclareRobustCommand\mathunderscore{\kern.06em\vbox{\hrule\@width.3em}}

```

(End of definition for `\mathparagraph` and others.)

`\mathellipsis` This is plain \TeX 's `\ldots`.

```

630 \DeclareRobustCommand\mathellipsis{\mathinner{\ldotp\ldotp\ldotp}}%

```

(End of definition for `\mathellipsis`.)

6.6 Other special functions and parameters

6.6.1 Biggggg

```
631 </math>
632 <*math | latexrelease>
633 <latexrelease>\IncludeInRelease{2018/12/01}%
634 <latexrelease>          {\Big}{Start LR-mode}%
635 \DeclareRobustCommand\big[1]{\leavevmode@ifvmode
636   {\hbox{\left#1\vbox to8.5\p@{\right.\n@space$}}}
637 \DeclareRobustCommand\Big[1]{\leavevmode@ifvmode
638   {\hbox{\left#1\vbox to11.5\p@{\right.\n@space$}}}
639 \DeclareRobustCommand\bigg[1]{\leavevmode@ifvmode
640   {\hbox{\left#1\vbox to14.5\p@{\right.\n@space$}}}
641 \DeclareRobustCommand\Bigg[1]{\leavevmode@ifvmode
642   {\hbox{\left#1\vbox to17.5\p@{\right.\n@space$}}}
643 </math | latexrelease>
644 <latexrelease>\EndIncludeInRelease
645 <latexrelease>\IncludeInRelease{0000/00/00}%
646 <latexrelease>          {\Big}{Start LR-mode}%
647 <latexrelease>\def\big#1{{\hbox{\left#1\vbox to8.5\p@{\right.\n@space$}}}
648 <latexrelease>\def\Big#1{{\hbox{\left#1\vbox to11.5\p@{\right.\n@space$}}}
649 <latexrelease>\def\bigg#1{{\hbox{\left#1\vbox to14.5\p@{\right.\n@space$}}}
650 <latexrelease>\def\Bigg#1{{\hbox{\left#1\vbox to17.5\p@{\right.\n@space$}}}
651 <latexrelease>\EndIncludeInRelease
652 <*math>

653 \def\n@space{\null\delimiterspace\z@ \m@th}
```

6.6.2 The log-like functions

`\operator@font` The `\operator@font` determines the symbol font used for log-like functions.

```
654 \def\operator@font{\mathgroup\symoperators}
```

(End of definition for \operator@font.)

6.6.3 Parameters

```
655 \thinmuskip=3mu
656 \medmuskip=4mu plus 2mu minus 4mu
657 \thickmuskip=5mu plus 5mu
```

This finishes the low-level setup in `fontmath.ltx`.

```
658 </math>
```

7 Default cfg files

We provide default `cfg` files here to ensure that on installations that search large file trees we do not pick up some strange customisation files from somewhere.

```
659 <*cfgtext | cfgmath | cfgprel>
660 %%
661 %%
662 %%
663 %% Load the standard setup:
664 %%
665 <+cfgtext>\input{fonttext.ltx}
```



```

666 <+cfgmath>\input{fontmath.ltx}
667 <+cfgprel>\input{preload.ltx}
668 %%
669 %% Small changes could go here; see documentation in cffguide.tex for
670 %% allowed modifications.
671 %%
672 %% In particular it is not allowed to misuse this configuration file
673 %% to modify internal LaTeX commands!
674 %%
675 %% If you use this file as the basis for configuration please change
676 %% the \ProvidesFile lines to clearly identify your modification, e.g.,
677 %%
678 <+cfgtext>%% \ProvidesFile{fonttext.cfg}[2001/06/01
679 <+cfgmath>%% \ProvidesFile{fonttext.cfg}[2001/06/01
680 <+cfgprel>%% \ProvidesFile{preload.cfg}[2001/06/01
681 %% Customised local font setup]
682 %%
683 %%
684 </cfgtext | cfgmath | cfgprel>

```

File 31

preload.dtx

1 Overview

This file contains an number of possible settings for preloading fonts during installation of NFSS2 (which is used by L^AT_EX 2_ε). It will be used to generate the following files:

preload.min	minimal subset of fonts necessary to run NFSS2
preload.ori	preload of CM fonts similar to the old <code>lfonts.tex</code>
preload.ltx	The standard selection of preloads
cmpreloa.xpt	preload of CM fonts for 10pt document size
cmpreloa.xip	preload of CM fonts for 11pt document size
cmpreloa.xii	preload of CM fonts for 12pt document size
dcpreloa.xpt	preload of DC fonts for 10pt size
dcpreloa.xip	preload of DC fonts for 11pt size
dcpreloa.xii	preload of DC fonts for 12pt size

These files are for installations that make use of Computer Modern fonts either old encoding (OT1) or Cork encoding (T1). The Computer Modern fonts with Cork encoding are known as DC-fonts.

Most important is `preload.ltx` which is used during format generation. You are *not* allowed to change this file.

2 Customization

You can customize the preloaded fonts in your L^AT_EX 2_ε system by installing a file with the name `preload.cfg`. If this file exists it will be used in place of the system file `preload.ltx`. You can, for example, copy one of the files mentioned above (that can be generated from this source) to `preload.cfg`.

Or you can define completely other preloads. In that case start from `preload.min` since that contains the fonts that have to be preloaded by **all** L^AT_EX 2_ε systems.

Avoid using `preload.ori`, it will load so many fonts that on most installations it is nearly impossible to load other font families afterwards. This file is only generated to show what fonts have been preloaded by L^AT_EX 2.09.

If you normally use other fonts than Computer Modern `preload.min` might be best.

Warning: If you preload fonts with encodings other than the normally supported encodings you have to declare that encoding in a `fontdef.cfg` configuration file (see the documentation in the file `fontdef.dtx`). Adding an extra encoding to the format might produce non-portable documents, thus this should be avoided if possible.

3 Module switches for the DOCSTRIP program

The DOCSTRIP will generate the above file from this source using the following module directives:

driver	produce a documentation driver file
preload	produce a preload... file
cm	for OT1 encoded Computer Modern
dc	for T1 encoded Computer Modern
min	produce minimal subset
xpt	produce 10pt preloads
xipt	produce 11pt preloads
xipt	produce 12pt preloads
ori	produce preloads similar to old <code>lfonts.tex</code>
tex	produce <code>preload.ltx</code>

A typical DOCSTRIP command file would then have entries like:

```
generateFile{preload.min}{t}{\from{preload.dtx}{preload,min}}
```

for generating preload files.

4 A driver for this document

The next bit of code contains the documentation driver file for \TeX , i.e., the file that will produce the documentation you are currently reading. It will be extracted from this file by the DOCSTRIP program.

```

1 <*driver>
2 \documentclass{ltxdoc}
3 %\OnlyDescription % comment out for implementation details
4 \begin{document}
5   \DocInput{preload.dtx}
6 \end{document}
7 </driver>
```

5 The code

We begin by loading the math extension font (`cmex10`) and the \LaTeX line and circle fonts. It is necessary to do this explicitly since these are used by the \LaTeX format. Since the internal font name contains / characters and digits we construct the name via `\csname`. These are the only fonts (!) that must be loaded in this file.

All `\DeclarePreloadSizes` can be removed or others can be added, they only influence the processing speed.

```

8 \expandafter\font\csname OMX/cmex/m/n/10\endcsname=cmex10\relax
9 \font\tenln =line10 \font\tenlnw =linew10\relax
10 \font\tenirc=lcircle10 \font\tenircw=lcirclew10\relax
```

The above fonts should not be touched but anything below this point here in the preload suggestions can be modified without any problems.

```

11 <-tex>%*****
12 <-tex>% Start any modification below this point **
13 <-tex>%*****
14 <-tex>
15 %%
16 %% Computer Modern Roman:
17 %%-----
```

```

18 <*ori>
19 \DeclarePreloadSizes{OT1}{cmr}{m}{n}
20     {5,6,7,8,9,10,10.95,12,14.4,17.28,20.74,24.88}
21 \DeclarePreloadSizes{OT1}{cmr}{bx}{n}{9,10,10.95,12,14.4,17.28}
22 \DeclarePreloadSizes{OT1}{cmr}{m}{sl}{10,10.95,12}
23 \DeclarePreloadSizes{OT1}{cmr}{m}{it}{7,8,9,10,10.95,12}
24 </ori>
25 <+xpt & cm> \DeclarePreloadSizes{OT1}{cmr}{m}{n}{5,7,10}
26 <+xpt & dc> \DeclarePreloadSizes{T1}{cmr}{m}{n}{5,7,10}
27 <+xipt & cm> \DeclarePreloadSizes{OT1}{cmr}{m}{n}{6,8,10.95}
28 <+xipt & dc> \DeclarePreloadSizes{T1}{cmr}{m}{n}{6,8,10.95}
29 <+xipt & cm> \DeclarePreloadSizes{OT1}{cmr}{m}{n}{6,8,12}
30 <+xipt & dc> \DeclarePreloadSizes{T1}{cmr}{m}{n}{6,8,12}
31 %%
32 %% Computer Modern Sans:
33 %%-----
34 <+ori> \DeclarePreloadSizes{OT1}{cmss}{m}{n}{10,10.95,12}
35 %%
36 %% Computer Modern Typewriter:
37 %%-----
38 <+ori> \DeclarePreloadSizes{OT1}{cmtt}{m}{n}{9,10,10.95,12}
39 %%
40 %% Computer Modern Math:
41 %%-----
42 <*ori>
43 \DeclarePreloadSizes{OML}{cmm}{m}{it}
44     {5,6,7,8,9,10,10.95,12,14.4,17.28,20.74}
45 \DeclarePreloadSizes{OMS}{cmsy}{m}{n}
46     {5,6,7,8,9,10,10.95,12,14.4,17.28,20.74}
47 </ori>
48
49 The math fonts are the same for both DC and CM fonts. So far there isn't an agreed
50 on standard.
51
52 <*xpt>
53 \DeclarePreloadSizes{OML}{cmm}{m}{it}{5,7,10}
54 \DeclarePreloadSizes{OMS}{cmsy}{m}{n}{5,7,10}
55 </xpt>
56 <*xipt>
57 \DeclarePreloadSizes{OML}{cmm}{m}{it}{6,8,10.95}
58 \DeclarePreloadSizes{OMS}{cmsy}{m}{n}{6,8,10.95}
59 </xipt>
60 <*xipt>
61 \DeclarePreloadSizes{OML}{cmm}{m}{it}{6,8,12}
62 \DeclarePreloadSizes{OMS}{cmsy}{m}{n}{6,8,12}
63 </xipt>
64 %%
65 %% LaTeX symbol fonts:
66 %%-----
67 <*ori>
68 \DeclarePreloadSizes{U}{lasy}{m}{n}
69     {5,6,7,8,9,10,10.95,12,14.4,17.28,20.74}
70 </ori>
71 </preload>

```

File 32

lftntcmd.dtx

Abstract

The commands defined in this file `lftntcmd` are part of the kernel code for $\text{\LaTeX} 2_{\epsilon}/\text{NFSS2}$.

It is also meant to serve as documentation for package writers since it demonstrates how to define high-level font changing commands using a small number of creator functions.

1 Introduction

Font changes such as `\bfseries`, `\sffamily`, etc. are declarations; this means that their scope is delimited by the grouping structure, either by the next `\end` of some environment or by explicitly using a group, e.g., writing something like `{\bfseries...}` in the source. If you make the mistake of writing `\bfseries{...}` (thinking of `\bfseries` as a command with one argument) then the result is rather striking.

Font declarations are an artifact of the \TeX system and for several reasons it is better to avoid them on the user level whenever possible. In $\text{\LaTeX}3$ they will probably all be replaced by environments and by font commands taking one argument.

This file defines a creator function for such declarative font switches. This function creates commands which can be used in both math and text.

This file also defines a number of high-level commands (all starting with `\text..`) that have one argument and typeset this argument in the requested way. Thus these commands are for typesetting short pieces of text in a specific family, series or shape. These are all produced as examples of the use of a creator function which is itself also defined in this file.

Table 3 shows all these high-level commands in action. A further advantage of using these commands is that they automatically take care of any necessary italic correction on either side of their argument.

Thus, when using such commands, one does not have to worry about forgetting the italic correction when changing fonts. Only in very few situations is this additional space wrong but, for example, most typographers recommend omitting the italic correction if a small punctuation character, like a comma, directly follows the font change. Since the amount of correction required is partly a matter of taste, you can define in what situations the italic correction should be suppressed. This is done by putting the characters that should cancel a preceding italic correction in the list `\nocorrlist`.³⁹ The default definition for this list is produced by the following.

```
\newcommand \nocorrlist {,.}
```

It is best to declare the most often used characters first, because this will make the processing slightly faster. For example,

```
\emph{When using the \NFSS{} high-level commands,  
the \emph{proper} use of italic corrections is  
automatically taken care of}. Only
```

³⁹Any package that changes the `\catcode` of a character inside `\nocorrlist` must then explicitly reset the list. Otherwise the changed character will no longer be recognized by the suppression algorithm.

<i>Command</i>	<i>Corresponds to</i>	<i>Action</i>
<code>\textnormal{..}</code>	<code>\normalfont</code>	Typeset argument in normal family
<code>\textrm{..}</code>	<code>\rmfamily</code>	Typeset argument in roman family
<code>\textsf{..}</code>	<code>\sffamily</code>	Typeset argument in sans serif family
<code>\texttt{..}</code>	<code>\ttfamily</code>	Typeset argument in typewriter family
<code>\textmd{..}</code>	<code>\mdseries</code>	Typeset argument in medium series
<code>\textbf{..}</code>	<code>\bfseries</code>	Typeset argument in bold series
<code>\textup{..}</code>	<code>\upshape</code>	Typeset argument in normal shape
<code>\textit{..}</code>	<code>\itshape</code>	Typeset argument in <i>italic</i> shape
<code>\textsl{..}</code>	<code>\slshape</code>	Typeset argument in <i>slanted</i> shape
<code>\textsc{..}</code>	<code>\scshape</code>	Typeset argument in SMALL CAPS shape
<code>\emph{..}</code>	<code>\em</code>	Typeset argument <i>emphasized</i>

Table 3: Font-change commands with arguments

The font change commands provided here all start with `\text..` to emphasize that they are for use in normal text and to be easily memorable. They automatically take care of any necessary italic correction on either side of the argument.

`\emph{sometimes}` one has to help `\LaTeX{}` by adding a `\verb=\nocorr=` command.

which results in:

When using the NFSS high-level commands, the proper use of italic corrections is automatically taken care of. Only sometimes one has to help L^AT_EX by adding a `\nocorr` command.

In contrast, the use of the declaration forms is often more appropriate when you define your own commands or environments.

```
\newenvironment{bfitemize}{\begin{itemize}\normalfont\bfseries}
                        {\end{itemize}}
\begin{bfitemize}
\item This environment produces boldface items.
\item It is defined in terms of \LaTeX's
      \texttt{itemize} environment and NFSS
      declarations.
\end{bfitemize}
```

This gives:

- **This environment produces boldface items.**
- **It is defined in terms of L^AT_EX's `itemize` environment and NFSS declarations.**

In addition to global customization of when to insert the italic correction, it is of course sometimes necessary to explicitly insert one with `\/`.

It is also possible to suppress the italic correction in individual instances. For this, the command `\nocorr` is provided.

The `\nocorr` must appear as the first or last token inside the braces of the argument of the `\text...` commands, at that end of the text where you wish to suppress the italic correction.

It is worth pointing out here that inserting a `\/` in places where it can have no function (i.e. anywhere except immediately after a slanted letter) is not an error—it will just be silently ignored. Unfortunately this is not true if the redefinition of `\/` in `amstex.sty` is used as this version can cause space to be removed immediately before the `\/`.

2 The implementation

`\DeclareTextFontCommand` This is the creator function for `\text..` commands. It gives a warning if `\foo` or `\fragfoo` is already defined.

In math mode it simply puts the font declaration and text into a box (possibly an automagically sized one).

Otherwise it first scans the text to see where `\nocorr` occurs within it. This sets the `\check@ic` commands to do what is necessary concerning the italic correction at both ends.

The algorithm for deciding whether to put in an italic correction is not very subtle: one is added whenever the newly current font is not itself positively sloped, unless the next token is a character in the ‘nocorr’ list. At the end of the text this is done after closing the group so as to check the ‘outer font’. Note that this will often result in adding an italic correction token after a character in an unsloped font; we believe (in early 2003) that this is perhaps inefficient but not dangerous.

It also now checks for empty contents of the text command and optimizes this case. Some care is also taken to check that doing dangerous things in vertical mode is avoided.

The italic correction token is added to the horizontal list before (in the list) an immediately preceding non-zero glob of glue (skip) and any non-zero penalty preceding that since, in the typical case, this puts it immediately after the last character in the preceding word.

Note that it is necessary to put in the `\aftergroup\maybe@ic` at the end of the group so that it comes after any other aftergroup tokens and immediately before the following tokens. It is also necessary to remove the `\fi` from the token list before the group ends; this is done by adding an `\expandafter` just before the closing brace.

```

1  \langle*2kernel\rangle
2  \def \DeclareTextFontCommand #1#2{%
3    \DeclareRobustCommand#1[1]{%
4      \ifmmode
5        \nfss@text{#2##1}%
6      \else
7        \hmode\bgroup
8        \text@command{##1}%
9        #2\check@icl ##1\check@icr
10       \expandafter
11       \egroup
12       \fi
13           }%
14 }

```

(End of definition for `\DeclareTextFontCommand`.)

```
\textrm Now we define the \text<family> commands in terms of the above; \texttt does not
\textsf look very nice!
\texttt 15 \DeclareTextFontCommand{\textrm}{\rmfamily}
\textnormal 16 \DeclareTextFontCommand{\textsf}{\sffamily}
17 \DeclareTextFontCommand{\texttt}{\ttfamily}
18 \DeclareTextFontCommand{\textnormal}{\normalfont}
```

(End of definition for `\textrm` and others.)

```
\textbf For the series attribute:
\textmd 19 \DeclareTextFontCommand{\textbf}{\bfseries}
20 \DeclareTextFontCommand{\textmd}{\mdseries}
```

(End of definition for `\textbf` and `\textmd`.)

```
\textit And for the shapes:
\textsl 21 \DeclareTextFontCommand{\textit}{\itshape}
\textsc 22 \DeclareTextFontCommand{\textsl}{\slshape}
\textup 23 \DeclareTextFontCommand{\textsc}{\scshape}
24 \DeclareTextFontCommand{\textup}{\upshape}
```

(End of definition for `\textit` and others.)

```
\textulc
\textsw 25 </2ekernel>
\textssc 26 <*2ekernel | latexrelease>
27 <latexrelease>\IncludeInRelease{2020/02/02}%
28 <latexrelease> \textulc}{Additional text commands}%
29 \DeclareTextFontCommand{\textulc}{\ulcshape}
30 \DeclareTextFontCommand{\textsw}{\swshape}
31 \DeclareTextFontCommand{\textssc}{\sscshape}
32 </2ekernel | latexrelease>
33 <latexrelease>\EndIncludeInRelease
34 <latexrelease>\IncludeInRelease{0000/00/00}%
35 <latexrelease> \textulc}{Additional text commands}%
36 <latexrelease>
37 <latexrelease>\let\textulc\@undefined
38 <latexrelease>\let\textsw\@undefined
39 <latexrelease>\let\textssc\@undefined
40 <latexrelease>\EndIncludeInRelease
41 <*2ekernel>
```

(End of definition for `\textulc`, `\textsw`, and `\textssc`.)

`\emph` Finally we have the `\em` font change declaration of L^AT_EX. The corresponding definition with argument is

```
42 \DeclareTextFontCommand{\emph}{\em}
```

(End of definition for `\emph`.)

`\nocorr` This is just a label, so it does nothing; it should also be unexpandable.

```
43 \let \nocorr \relax
```


(End of definition for `\nocorr`.)

`\check@ic1` We define these defaults in case some error causes them to be expanded at the wrong
`\check@icr` time.

```
44 \let \check@ic1 \@empty
45 \let \check@icr \@empty
```

(End of definition for `\check@ic1` and `\check@icr`.)

`\text@command` This checks for a `\nocorr` as the first token in its argument and also for one in any other
`\check@nocorr@` position not protected within braces (the latter is treated as if it were at the end of the argument).

Is this the correct action in the ‘empty’ case? It is efficient but typographically it is, strictly, incorrect!

```
46 \def \text@command #1{%
47   \edef \reserved@a {\unexpanded{#1}}%
48   \ifx \reserved@a \@empty
49     \let \check@ic1 \@empty
50     \let \check@icr \@empty
51   \else
```

`\space` is a reserved word in L^AT_EX or actually already in plain T_EX. If somebody really redefines it so many things will break that I don’t see any reason to make this routine here slower than necessary.

```
52 %   \def \reserved@b { }%
53 %   \ifx \reserved@a \reserved@b
54   \ifx \reserved@a \space
55     \let \check@ic1 \@empty
56     \let \check@icr \@empty
57   \else
58     \check@nocorr@ #1\nocorr\@nil
59   \fi
60 \fi
61 }
62 \def \check@nocorr@ #1#2\nocorr#3\@nil {%
```

The two checks are initialised here to their values in the normal case.

```
63 \let \check@ic1 \maybe@ic
64 \def \check@icr {\ifvmode \else \aftergroup \maybe@ic \fi}%
65 \def \reserved@a {\nocorr}%
66 \def \reserved@b {#1}%
67 \def \reserved@c {#3}%
68 \ifx \reserved@a \reserved@b
69   \ifx \reserved@c \@empty
```

In this case there is a `\nocorr` at the start but not at the end, so `\check@ic1` should be empty.

```
70   \let \check@ic1 \@empty
71   \else
```

Otherwise there is a `\nocorr` both at the start and elsewhere, so no italic corrections should be added.

```
72     \let \check@ic1 \@empty
73     \let \check@icr \@empty
74   \fi
```

```

75  \else
76  \ifx \reserved@c \@empty

```

In this case there is no `\nocorr` anywhere, so we need to check for an italic correction at both the beginning and the end. This has been set up as the default so no code is needed here.

```

77  \else

```

In this case there is no `\nocorr` at the start but there is one elsewhere, so no `\aftergroup` is needed.

```

78      \let \check@icr \@empty
79      \fi
80  \fi
81 }

```

(End of definition for `\text@command` and `\check@nocorr@`.)

`\ifmaybe@ic` Switch used solely within `\maybe@ic` not interfering with other switches.

```

82 \newif\ifmaybe@ic

```

(End of definition for `\ifmaybe@ic`.)

`\maybe@ic` These macros implement the italic correction.

```

\maybe@ic@
83 \def \maybe@ic {\futurelet\@let@token\maybe@ic@}
84 \def \maybe@ic@ {%

```

We first check to see if the current font is positively sloped. (But do not forget the message Rainer sent about an upright font with non-zero slope! Or is this an urban myth?) It has been suggested that this should test against a small positive value, but what?

```

85  \ifdim \fontdimen\@ne\font>\z@
86  \else
87  \maybe@ictrue

```

It would be possible, but probably not worthwhile, to continue the forward scan beyond any closing braces.

```

88      \expandafter\@tfor\expandafter\reserved@a\expandafter:\expandafter=%
89      \nocorrlist

```

We have to hide the `\@let@token` in the macro `\t@st@ic` rather than testing it directly in the loop since it might be `\let` to a `\fi` or `\else`, which would result in chaos.

```

90  \do \t@st@ic

```

Frank thinks that the next bit is inefficient if done after the second change. Chris thinks that most all of this is inefficient for the commonest cases: but that is the price of a cleverer algorithm. It is certainly needed to deal with the use of `\nolinebreak`.

```

91      \ifmaybe@ic \sw@slant \fi
92  \fi
93 }

```

(End of definition for `\maybe@ic` and `\maybe@ic@`.)

`\t@st@ic` The next token in the input stream is stored in `\@let@token` via a `\let`, the current token from `\nocorrlist` is stored via `\def` in `\reserved@a`. To compare them we have to fiddle around a bit.

If the only things to check were characters then this could be done via an `\if` thus their catcodes would not matter; but this will not work whilst `\futurelet` is used above.

```

94 \def \t@st@ic {%
95   \expandafter\let\expandafter\reserved@b\expandafter=\reserved@a\relax
96   \ifx\reserved@b\@let@token

```

If they are the same we record the fact and jump out of the loop.

```

97     \maybe@icfalse
98     \@break@tfor
99   \fi
100 }

```

(End of definition for \t@st@ic.)

`\sw@slant` The definition of the mysterious `\sw@slant` command is as follows.
`\fix@penalty`

```

101 \def \sw@slant {%

```

It is surely correct to put in an italic correction when there is no skip. If the last thing on the list is actually a zero skip (including things whose dimension part is zero, such as `\hfill`), or anything other than a character, then the italic correction will have no effect.

In order to work correctly with unbreakable spaces from `~` (and other common forms of line-breaking control) we also move back across a penalty before the glue.

```

102   \ifdim \lastskip=\z@
103     \fix@penalty
104   \else
105     \skip@ \lastskip
106     \unskip
107     \fix@penalty
108     \hskip \skip@
109   \fi
110 }

```

The above code means: “If there is a non-zero space just before the current position (`\ifdim...`) save the amount of that space (`\skip@\lastskip`), remove it (`\unskip`), then do a similar thing if there is a penalty just before the skip, and finally put the space back in.”

Since zero glue cannot be distinguished in this context from no glue, we dare not put in an `\hskip` in this case as this may produce an unwanted breakpoint. This is not satisfactory.

The penalty before the glue is handled similarly, with the same caveats concerning the zero case. Is this the first recorded use of `\unpenalty` in standard L^AT_EX code?

```

111 \def \fix@penalty {%
112   \ifnum \lastpenalty=\z@
113     \@@italiccorr
114   \else
115     \count@ \lastpenalty
116     \unpenalty
117     \@@italiccorr

```

```

118     \penalty \count@
119     \fi
120 }

```

(End of definition for `\sw@slant` and `\fix@penalty`.)

`\nocorrlist` This holds the list of characters that should prevent italic correction. They should be ordered by decreasing frequency of use. If any such character is made active later on one needs to redefine the list so that the active character becomes part of it.

```

121 \def \nocorrlist {,.}

```

(End of definition for `\nocorrlist`.)

`\nfss@text` This command will by default behave like a L^AT_EX `\mbox` but may be redefined by packages such as `amstext.sty` to be a bit cleverer.

```

122 \ifx \nfss@text\undefined
123   \def \nfss@text {\leavevmode\hbox}
124 \fi

```

(End of definition for `\nfss@text`.)

`\DeclareOldFontCommand` This is the function used to create declarative font-changing commands that can also be used to change alphabets in math-mode.

Usage: `\DeclareOldFontCommand \fn{⟨font-change decls⟩} ⟨math-alphabet⟩`

Here `\fn` is the font-declaration command being defined, `⟨font-change decls⟩` is the declaration it will expand to in text-mode, and `⟨math-alphabet⟩` is the (single) math alphabet specifier which is to be used in math-mode.

It does not care whether the command being defined already exists but it does give a warning if it redefines anything.

Here are some typical examples of its use in conjunction with more basic NFSS2 font commands.

```

\DeclareOldFontCommand{\rm}{\normalfont\rmfamily}{\mathrm}
\DeclareOldFontCommand{\sf}{\normalfont\sffamily}{\mathsf}
\DeclareOldFontCommand{\tt}{\normalfont\ttfamily}{\mathtt}

```

```

125 \def \DeclareOldFontCommand #1#2#3{%
126   \DeclareRobustCommand #1{\@fontswitch {#2}{#3}}%
127 }

```

(End of definition for `\DeclareOldFontCommand`.)

`\@fontswitch` `\@math@egroup` These two commands actually do the necessary tests and declarative font- or alphabet-changing.

```

\@math@egroup
\@math@egroup
128 \def \@fontswitch #1#2{%
129   \ifmmode
130     \let \math@bgroup \relax
131     \def \math@egroup {\let \math@bgroup \@math@bgroup
132                       \let \math@egroup \@math@egroup}%

```

We need to have a `\relax` in the following line in case the `#2` is something like `\mathsf` grabbing the next token as an argument. For this reason the code also uses explicit arguments again (see pr/1275).

```

133     #2\relax
134   \else
135     #1%
136   \fi
137 }
138 \let \@@math@bgroup \math@bgroup
139 \let \@@math@egroup \math@egroup

```

(End of definition for `\fontswitch`, `\@math@egroup`, and `\@math@egroup`.)

These commands are available only in the preamble.

```

140 \@onlypreamble \DeclareTextFontCommand
141 \@onlypreamble \DeclareOldFontCommand

```

3 Initialization

`\normalsize` This is defined to produce an error.

```

142 \def\normalsize{%
143   \@latex@error {The font size command \protect\normalsize\space
144                 is not defined:\MessageBreak
145                 there is probably something wrong with
146                 the class file}\@eha
147 }
148 </2ekernel>

```

(End of definition for `\normalsize`.)

File 33

ltxtextcomp.dtx

This file contains the implementation for accessing the glyphs provided by the TS1 encoding (Text Companion Encoding). This is now offered as part of the kernel and so the `textcomp` package which used to provide the definitions is now mainly needed for compatibility reasons (and doesn't do much any more).

```
1 <*2ekernel | latexrelease>
2 <latexrelease>\NewModuleRelease{2020/02/02}{ltxtextcomp}
3 <latexrelease>                {Text Companion symbols}
```

`\oldstylenums` Preserve the old definition of `\oldstylenums` under a different name.

`\legacyoldstylenums` This macro implements old style numerals but only works if we assume that the standard math fonts are used. Thus it needs changing in case other math encodings are used.

```
4 \DeclareRobustCommand\legacyoldstylenums[1]{%
5   \begingroup
```

Provide spacing using the interword space of the current font.

```
6   \spaceskip\fontdimen\tw@\font
```

Then switch to the math italic font. We don't change the current value of `\f@series` which means that you can use bold numerals if `\bfseries` is in force. As family we use `\rmdefault` which means that this only works if there exist an OML encoded version of that font or rather a corresponding `.fd` file (which is the case for standard L^AT_EX fonts even though they only contain substitutions).

```
7   \usefont{OML}{\rmdefault}{\f@series}{it}%
8   \mathgroup\symbols #1%
9   \endgroup
10 }
```

And here is the improved one that adjusts depending on surroundings.

```
11 \DeclareRobustCommand\oldstylenums[1]{%
12   \begingroup
13   \ifmode
14     \mathgroup\symbols #1%
15   \else
```

The `\CheckEncodingSubset` is discussed below.

```
16   \CheckEncodingSubset\@use@text@encoding{TS1}\tc@oldstylesubst2{{#1}}%
17   \fi
18   \endgroup
19 }
```

The helper to select the substitution if needed.

```
20 \def\tc@oldstylesubst#1{%
21   \tc@errorwarn
22     {Oldstyle digits unavailable for
23     family \f@family.\MessageBreak
24     Default oldstyle digits used instead}\@eha
25   \bgroup
26     \expand@font@defaults
```

The substitution defaults are provided in the file `fonttext.ltx`.

```

27     \ifx\f@family\rmdef@ult
28         \fontfamily\rmsubstdefault
29     \else\ifx\f@family\sfdef@ult
30         \fontfamily\sfsbstdefault
31     \else\ifx\f@family\ttdef@ult
32         \fontfamily\ttsubstdefault
33     \else
34         \fontfamily\textcompsubstdefault
35     \fi\fi\fi
36     \fontencoding{TS1}\selectfont#1%
37 \egroup
38 }

```

(End of definition for \oldstylenums and \legacyoldstylenums.)

`\textcompsubstdefault` Here is the default for the “unknown” case:

```

39 \def\textcompsubstdefault{\rmsubstdefault}

```

(End of definition for \textcompsubstdefault.)

To set up the glyphs for the subsets we need a number helpers.

`\tc@errorwarn` To we produce errors, warnings, or only info in the transcripts if glyphs require substitutions? By default it is “info” only. With the `textcomp` package that can be changed.

```

40 \def\tc@errorwarn#1#2{\@latex@info{#1}}

```

(End of definition for \tc@errorwarn.)

`\tc@subst`

```

41 \def\tc@subst#1{%
42     \tc@errorwarn
43     {Symbol \string#1 not provided by\MessageBreak
44     font family \f@family\space
45     in TS1 encoding.\MessageBreak Default family used instead}\@eha
46     \bgroup
47     \expand@font@defaults
48     \ifx\f@family\rmdef@ult
49         \fontfamily\rmsubstdefault
50     \else\ifx\f@family\sfdef@ult
51         \fontfamily\sfsbstdefault
52     \else\ifx\f@family\ttdef@ult
53         \fontfamily\ttsubstdefault
54     \else
55         \fontfamily\textcompsubstdefault
56     \fi\fi\fi

```

Whatever default was chosen, we claim now (locally hopefully) that it can handle all slots (even if not true) to avoid looping in certain situations, e.g., when something was set up incorrectly.

```

57     \@namedef{TS1:\f@family}{0}%
58     \selectfont#1%
59 \egroup
60 }

```

(End of definition for \tc@subst.)

`\tc@fake@euro` `\tc@fake@euro` is an example of a “fake” definition to use in arg #3 of the command `\CheckEncodingSubset` when a symbol is not available in a certain font family. Here we produce a poor man’s Euro symbol by combining a “C” with a “=”.

```

61 \def\tc@fake@euro#1{%
62   \leavevmode
63   \@font@info{Faking \noexpand#1for font family
64                 \f@family\MessageBreak in TS1 encoding}%
65   \valign{##\cr
66     \vfil\hbox to 0.07em{\dimen@f@size\p@
67                       \math@fontsfalse
68                       \fontsize{.7\dimen@}\z@\selectfont=\hss}%
69     \vfil\cr%
70     \hbox{C}\crrr
71   }%
72 }

```

(End of definition for `\tc@fake@euro`.)

`\tc@check@symbol` These are two abbreviations that we use below to check symbols and accents in TS1.
`\tc@check@accent` Only there to save some space, e.g., we can then write

```
DeclareTextCommandDefault{\textcurrency}{\tc@check@symbol3\textcurrency}
```

to ensure that `\textcurrency` is only typeset if the current font has a TS1 subset id of less than 3. Otherwise `\tc@error` is called telling the user that for this font family `\textcurrency` is not available.

```
73 \def\tc@check@symbol{\CheckEncodingSubset\UseTextSymbol{TS1}\tc@subst}
```

Accents have been made an error in the `textcomp` package when not available. Now that we provide the functionality in the kernel we avoid the error by swapping in a T1 accent if the TS1 accent is not available.

```

74 %\def\tc@check@accent{\CheckEncodingSubset\UseTextAccent{TS1}\tc@error}
75 \def\tc@check@accent#1{\CheckEncodingSubset\UseTextAccent
76                       {TS1}{\tc@swap@accent#1}}
77 \def\tc@swap@accent#1#2{\UseTextAccent{T1}#1}

```

(End of definition for `\tc@check@symbol` and `\tc@check@accent`.)

1 Sub-encodings

Here are the default definitions for the TS1 symbols. First those that we assume are always available if a font implements TS1.

```

78 \DeclareTextSymbolDefault{\textdollar}{TS1}
79 \UndeclareTextCommand{\textdollar}{OT1} % don't use the OT1 def any longer
80 \DeclareTextSymbolDefault{\textsterling}{TS1}
81 \UndeclareTextCommand{\textsterling}{OT1}% don't use the OT1 def any longer
82 \DeclareTextSymbolDefault{\textperthousand}{TS1}
83 \UndeclareTextCommand{\textperthousand}{T1} % don't use the T1 def

```


Using `\UndeclareTextCommand` above is enough only if the encoding definition files are not reloaded afterwards. In the past that happened if `fontenc` was used in the document preamble (not any longer). So in some sense it is better to fully remove them from the encoding files, but for rollbacks it is easier to keep them in for now.

These are the standard `itemize` and footnote symbols originally taken from OMS and now from TS1:

```

84 \DeclareTextSymbolDefault{\textasteriskcentered}{TS1}
85 \DeclareTextSymbolDefault{\textbullet}{TS1}
86 \DeclareTextSymbolDefault{\textdaggerdbl}{TS1}
87 \DeclareTextSymbolDefault{\textdagger}{TS1}
88 \DeclareTextSymbolDefault{\textparagraph}{TS1}
89 \DeclareTextSymbolDefault{\textperiodcentered}{TS1}
90 \DeclareTextSymbolDefault{\textsection}{TS1}

```

And here are the other TS1 glyphs that are implemented by every font (or nearly every)—a few are commented out and moved to sub-encoding 9, because they aren't around in some fonts.

```

91 %%\DeclareTextSymbolDefault{\textbardbl}{TS1} % subst in sub-enc 9 above
92 \DeclareTextSymbolDefault{\textbrokenbar}{TS1}
93 %%\DeclareTextSymbolDefault{\textcelsius}{TS1} % subst in sub-enc 9 above
94 \DeclareTextSymbolDefault{\textcent}{TS1}
95 \DeclareTextSymbolDefault{\textcopyright}{TS1}
96 \DeclareTextSymbolDefault{\textdegree}{TS1}
97 \DeclareTextSymbolDefault{\textdiv}{TS1}
98 \DeclareTextSymbolDefault{\textlnot}{TS1}
99 \DeclareTextSymbolDefault{\textonehalf}{TS1}
100 \DeclareTextSymbolDefault{\textonequarter}{TS1}
101 %%\DeclareTextSymbolDefault{\textonesuperior}{TS1} % subst in sub-enc 9 above
102 \DeclareTextSymbolDefault{\textordfeminine}{TS1}
103 \DeclareTextSymbolDefault{\textordmasculine}{TS1}
104 \DeclareTextSymbolDefault{\textpm}{TS1}
105 \DeclareTextSymbolDefault{\textquotesingle}{TS1}
106 \DeclareTextSymbolDefault{\textquotestraightbase}{TS1}
107 \DeclareTextSymbolDefault{\textquotestraightdblbase}{TS1}
108 \DeclareTextSymbolDefault{\textregistered}{TS1}
109 %%\DeclareTextSymbolDefault{\textthreequartersemdash}{TS1} % subst in sub-enc 9 above
110 \DeclareTextSymbolDefault{\textthreequarters}{TS1}
111 %%\DeclareTextSymbolDefault{\textthreesuperior}{TS1} % subst in sub-enc 9 above
112 \DeclareTextSymbolDefault{\texttimes}{TS1}
113 \DeclareTextSymbolDefault{\texttrademark}{TS1}
114 %%\DeclareTextSymbolDefault{\texttwelveudash}{TS1} % subst in sub-enc 9 above
115 %%\DeclareTextSymbolDefault{\texttwosuperior}{TS1} % subst in sub-enc 9 above
116 \DeclareTextSymbolDefault{\textyen}{TS1}

117 \DeclareTextSymbolDefault{\textcapitalcompwordmark}{TS1}
118 \DeclareTextSymbolDefault{\textascendercompwordmark}{TS1}

```

In the following sections the remaining default definitions are ordered by sub-encoding in which they are become **unavailable**, i.e., they are not provided in the sub-encoding with that number and all sub-encodings with higher numbers.

Thus the symbols that are available in sub-encoding x are the symbols above (always available) and the symbols listed as becoming unavailable in sub-encodings $x + 1$ and higher.

1.1 Unavailable in sub-encoding 1 and higher (drop symbols not working in Latin Modern)

The `\textcircled` is available but the glyph is simply too small so we keep using the OMS glyph.

```
119 \DeclareTextCommandDefault{\textcircled}
120   {\CheckEncodingSubset\UseTextAccent{TS1}{\UseTextAccent{OMS}}1\textcircled}
```

1.2 Unavailable in sub-encoding 2 (majority of new OTF fonts via autoinst) and higher

```
121 \DeclareTextCommandDefault{\t}
122   {\CheckEncodingSubset\UseTextAccent{TS1}{\UseTextAccent{OML}}2\t}
```

Capital accents are really only very seldom implemented, so from sub-encoding 2 onwards we use the normal T1 accents if they are asked for in the document.

In Unicode engines we don't implement them at all but always use the basic accents instead. whether that works or not really depends on the font, something like `\"X` usually comes out wrong in Unicode engines.

```
123 \ifx\Umathcode\@undefined
124   \DeclareTextCommandDefault{\capitalacute}
125     {\tc@check@accent{\'}2\capitalacute}
126   \DeclareTextCommandDefault{\capitalbreve}
127     {\tc@check@accent{\u}2\capitalbreve}
128   \DeclareTextCommandDefault{\capitalcaron}
129     {\tc@check@accent{\v}2\capitalcaron}
130   \DeclareTextCommandDefault{\capitalcedilla}
131     {\tc@check@accent{\c}2\capitalcedilla}
132   \DeclareTextCommandDefault{\capitalcircumflex}
133     {\tc@check@accent{\^}2\capitalcircumflex}
134   \DeclareTextCommandDefault{\capitaldieresis}
135     {\tc@check@accent{\"}2\capitaldieresis}
136   \DeclareTextCommandDefault{\capitaldotaccent}
137     {\tc@check@accent{\.}2\capitaldotaccent}
138   \DeclareTextCommandDefault{\capitalgrave}
139     {\tc@check@accent{\`}2\capitalgrave}
140   \DeclareTextCommandDefault{\capitalhungarumlaut}
141     {\tc@check@accent{\H}2\capitalhungarumlaut}
142   \DeclareTextCommandDefault{\capitalmacron}
143     {\tc@check@accent{\=}2\capitalmacron}
144   \DeclareTextCommandDefault{\capitalogonek}
145     {\tc@check@accent{\k}2\capitalogonek}
146   \DeclareTextCommandDefault{\capitalring}
147     {\tc@check@accent{\r}2\capitalring}
148   \DeclareTextCommandDefault{\capitaltie}
149     {\tc@check@accent{\t}2\capitaltie}
150   \DeclareTextCommandDefault{\capitaltilde}
151     {\tc@check@accent{\~}2\capitaltilde}
```

For `\newtie` and `\capitalnewtie` this is actually wrong, they should pick up the accent from the substitution font (not done yet).

```
152 \DeclareTextCommandDefault{\newtie}
153     {\tc@check@accent{\t}2\newtie}
154 \DeclareTextCommandDefault{\capitalnewtie}
```

```
155 \tc@check@accent{\t}2\capitalnewtie}
```

In Unicode engines we just execute the simple accents:

```
156 \else
157 \DeclareTextCommandDefault\capitalacute{\@tabacckludge'}
158 \DeclareTextCommandDefault\capitalbreve{\u}
159 \DeclareTextCommandDefault\capitalcaron{\v}
160 \DeclareTextCommandDefault\capitalcedilla{c}
161 \DeclareTextCommandDefault\capitalcircumflex{\^}
162 \DeclareTextCommandDefault\capitaldieresis{"}
163 \DeclareTextCommandDefault\capitaldotaccent{.}
164 \DeclareTextCommandDefault\capitalgrave{\@tabacckludge'}
165 \DeclareTextCommandDefault\capitalhungarumlaut{H}
166 \DeclareTextCommandDefault\capitalmacron{\@tabacckludge=}
167 \DeclareTextCommandDefault\capitalnewtie{\t}
168 \DeclareTextCommandDefault\capitalogonek{k}
169 \DeclareTextCommandDefault\capitalring{r}
170 \DeclareTextCommandDefault\capitaltie{\t}
171 \DeclareTextCommandDefault\capitaltilde{\~}
172 \DeclareTextCommandDefault\newtie{\t}
173 \fi
```

The next two symbols exist in some fonts (faked?), but we ignore that to keep the subsets reasonable compact and most important linear.

```
174 \DeclareTextCommandDefault{\textlbrackdbl}
175 \tc@check@symbol2\textlbrackdbl}
176 \DeclareTextCommandDefault{\textrbrackdbl}
177 \tc@check@symbol2\textrbrackdbl}
```

Old style numerals are again in some fonts but using -OsF, etc. is the better approach to get them, so we claim they aren't in sub-encoding 2 as that's true for most fonts.

```
178 \DeclareTextCommandDefault{\texteightoldstyle}
179 \tc@check@symbol2\texteightoldstyle}
180 \DeclareTextCommandDefault{\textfiveoldstyle}
181 \tc@check@symbol2\textfiveoldstyle}
182 \DeclareTextCommandDefault{\textfouroldstyle}
183 \tc@check@symbol2\textfouroldstyle}
184 \DeclareTextCommandDefault{\textnineoldstyle}
185 \tc@check@symbol2\textnineoldstyle}
186 \DeclareTextCommandDefault{\textoneoldstyle}
187 \tc@check@symbol2\textoneoldstyle}
188 \DeclareTextCommandDefault{\textsevenoldstyle}
189 \tc@check@symbol2\textsevenoldstyle}
190 \DeclareTextCommandDefault{\textsixoldstyle}
191 \tc@check@symbol2\textsixoldstyle}
192 \DeclareTextCommandDefault{\textthreeoldstyle}
193 \tc@check@symbol2\textthreeoldstyle}
194 \DeclareTextCommandDefault{\texttwooldstyle}
195 \tc@check@symbol2\texttwooldstyle}
196 \DeclareTextCommandDefault{\textzerooldstyle}
197 \tc@check@symbol2\textzerooldstyle}
```

The next set of glyphs is special to T_EX fonts (and available with a few older PS fonts supported through virtual fonts), but not any longer in the majority of fonts provided through autinst, so we pretend there aren't available in sub-encoding 2 and below.

```
198 \DeclareTextCommandDefault{\textacutedbl}
```

```

199             {\tc@check@symbol2\textacutedbl}
200 \DeclareTextCommandDefault{\textasciiacute}
201             {\tc@check@symbol2\textasciiacute}
202 \DeclareTextCommandDefault{\textasciibreve}
203             {\tc@check@symbol2\textasciibreve}
204 \DeclareTextCommandDefault{\textasciicaron}
205             {\tc@check@symbol2\textasciicaron}
206 \DeclareTextCommandDefault{\textasciidieresis}
207             {\tc@check@symbol2\textasciidieresis}
208 \DeclareTextCommandDefault{\textasciigrave}
209             {\tc@check@symbol2\textasciigrave}
210 \DeclareTextCommandDefault{\textasciimacron}
211             {\tc@check@symbol2\textasciimacron}
212 \DeclareTextCommandDefault{\textgravedbl}
213             {\tc@check@symbol2\textgravedbl}
214 \DeclareTextCommandDefault{\texttildelow}
215             {\tc@check@symbol2\texttildelow}

```

Finally those below are only available in CM-based fonts but in no font that has its origin outside of the T_EX world.

```

216 \DeclareTextCommandDefault{\textbaht}
217             {\tc@check@symbol2\textbaht}
218 \DeclareTextCommandDefault{\textbigcircle}
219             {\tc@check@symbol2\textbigcircle}
220 \DeclareTextCommandDefault{\textborn}
221             {\tc@check@symbol2\textborn}
222 \DeclareTextCommandDefault{\textcentoldstyle}
223             {\tc@check@symbol2\textcentoldstyle}
224 \DeclareTextCommandDefault{\textcircledP}
225             {\tc@check@symbol2\textcircledP}
226 \DeclareTextCommandDefault{\textcopyleft}
227             {\tc@check@symbol2\textcopyleft}
228 \DeclareTextCommandDefault{\textdblhyphenchar}
229             {\tc@check@symbol2\textdblhyphenchar}
230 \DeclareTextCommandDefault{\textdblhyphen}
231             {\tc@check@symbol2\textdblhyphen}
232 \DeclareTextCommandDefault{\textdied}
233             {\tc@check@symbol2\textdied}
234 \DeclareTextCommandDefault{\textdiscount}
235             {\tc@check@symbol2\textdiscount}
236 \DeclareTextCommandDefault{\textdivorced}
237             {\tc@check@symbol2\textdivorced}
238 \DeclareTextCommandDefault{\textdollaroldstyle}
239             {\tc@check@symbol2\textdollaroldstyle}
240 \DeclareTextCommandDefault{\textguarani}
241             {\tc@check@symbol2\textguarani}
242 \DeclareTextCommandDefault{\textleaf}
243             {\tc@check@symbol2\textleaf}
244 \DeclareTextCommandDefault{\textlquill}
245             {\tc@check@symbol2\textlquill}
246 \DeclareTextCommandDefault{\textmarried}
247             {\tc@check@symbol2\textmarried}
248 \DeclareTextCommandDefault{\textmho}
249             {\tc@check@symbol2\textmho}
250 \DeclareTextCommandDefault{\textmusicalnote}

```

```

251             {\tc@check@symbol2\textmusicalnote}
252 \DeclareTextCommandDefault{\textnaira}
253             {\tc@check@symbol2\textnaira}
254 \DeclareTextCommandDefault{\textopenbullet}
255             {\tc@check@symbol2\textopenbullet}
256 \DeclareTextCommandDefault{\textpeso}
257             {\tc@check@symbol2\textpeso}
258 \DeclareTextCommandDefault{\textpilcrow}
259             {\tc@check@symbol2\textpilcrow}
260 \DeclareTextCommandDefault{\textrecipe}
261             {\tc@check@symbol2\textrecipe}
262 \DeclareTextCommandDefault{\textreferencemark}
263             {\tc@check@symbol2\textreferencemark}
264 \DeclareTextCommandDefault{\textrquill}
265             {\tc@check@symbol2\textrquill}
266 \DeclareTextCommandDefault{\textservicemark}
267             {\tc@check@symbol2\textservicemark}
268 \DeclareTextCommandDefault{\textsurd}
269             {\tc@check@symbol2\textsurd}

```

The `\textpertenthousand` also belongs in this group but here we have a choice: in T1 there is a definition for `\textpertenthousand` making the symbol up from % and `\char 24` (twice) but in many fonts that char doesn't exist and the slot is reused for random ligatures. So better not use it because often it is wrong. But pointing to TS1 is also not great as only a few fonts have it as a real symbol, so we get a substitution to CM or LM.

Alternatively we could just state that the symbol is unavailable in those fonts. For now I substitute.

```

270 \DeclareTextCommandDefault{\textpertenthousand}
271             {\tc@check@symbol2\textpertenthousand}
272 \UndeclareTextCommand{\textpertenthousand}{T1}

```

1.3 Unavailable in sub-encoding 3 and higher

Sub-encoding 2 is the one where we loose many symbols. In the higher-numbered sub-encodings we see only a few dropped additionally.

```

273 \DeclareTextCommandDefault{\textlangle}
274             {\tc@check@symbol3\textlangle}
275 \DeclareTextCommandDefault{\textrangle}
276             {\tc@check@symbol3\textrangle}

```

1.4 Unavailable in sub-encoding 4 and higher

```

277 \DeclareTextCommandDefault{\textcolonmonetary}
278             {\tc@check@symbol4\textcolonmonetary}
279 \DeclareTextCommandDefault{\textdong}
280             {\tc@check@symbol4\textdong}
281 \DeclareTextCommandDefault{\textdownarrow}
282             {\tc@check@symbol4\textdownarrow}
283 \DeclareTextCommandDefault{\textleftarrow}
284             {\tc@check@symbol4\textleftarrow}
285 \DeclareTextCommandDefault{\textlira}
286             {\tc@check@symbol4\textlira}
287 \DeclareTextCommandDefault{\textrightarrow}

```

```

288             {\tc@check@symbol4\textrightarrow}
289 \DeclareTextCommandDefault{\textuparrow}
290             {\tc@check@symbol4\textuparrow}
291 \DeclareTextCommandDefault{\textwon}
292             {\tc@check@symbol4\textwon}

```

1.5 Unavailable in sub-encoding 5 (most older PS fonts) and higher

Most older PS fonts (supported in T_EX since the early nineties when virtual fonts became available) are sorted under this sub-encoding. But in reality, many of them don't have all glyphs that should be available in sub-encoding 5. Instead they show little squares, i.e., they produce “tofu” if you are unlucky.

But the coverage is so random that it is impossible to sort them properly and if we tried to ensure that they only typeset those glyphs that are really always available, we would have to put them all into sub-encoding 9; so putting them into 5 is really a compromise.

Modern fonts usually don't typeset a tofu character if a glyph is missing. They are therefore only classified as sub-encoding 5 if they really support its glyph set completely.

```

293 \DeclareTextCommandDefault{\textestimated}
294             {\tc@check@symbol5\textestimated}
295 \DeclareTextCommandDefault{\textnumero}
296             {\tc@check@symbol5\textnumero}

```

1.6 Unavailable in sub-encoding 6 and higher

```

297 \DeclareTextCommandDefault{\textflorin}
298             {\tc@check@symbol6\textflorin}
299 \DeclareTextCommandDefault{\textcurrency}
300             {\tc@check@symbol6\textcurrency}

```

1.7 Unavailable in sub-encoding 7 and higher

```

301 \DeclareTextCommandDefault{\textfractionsolidus}
302             {\tc@check@symbol7\textfractionsolidus}
303 \DeclareTextCommandDefault{\textohm}
304             {\tc@check@symbol7\textohm}
305 \DeclareTextCommandDefault{\textmu}
306             {\tc@check@symbol7\textmu}
307 \DeclareTextCommandDefault{\textminus}
308             {\tc@check@symbol7\textminus}

```

1.8 Unavailable in sub-encoding 8 and higher

```

309 \DeclareTextCommandDefault{\textblank}
310             {\tc@check@symbol{8}\textblank}
311 \DeclareTextCommandDefault{\textinterrobangdown}
312             {\tc@check@symbol{8}\textinterrobangdown}
313 \DeclareTextCommandDefault{\textinterrobang}
314             {\tc@check@symbol{8}\textinterrobang}

```

Fonts with this sub-encoding don't have a Euro symbol, but instead of substituting we fake it.

```

315 \DeclareTextCommandDefault{\texteuro}
316     {\CheckEncodingSubset\UseTextSymbol{TS1}\tc@fake@euro{8}\texteuro}

```

1.9 Unavailable in Sub-encoding 9 (most missing)

```
317 \DeclareTextCommandDefault{\textcelsius}
318         {\tc@check@symbol{9}\textcelsius}
319 \DeclareTextCommandDefault{\textonesuperior}
320         {\tc@check@symbol{9}\textonesuperior}
321 \DeclareTextCommandDefault{\textthreequartersemdash}
322         {\tc@check@symbol{9}\textthreequartersemdash}
323 \DeclareTextCommandDefault{\textthreesuperior}
324         {\tc@check@symbol{9}\textthreesuperior}
325 \DeclareTextCommandDefault{\texttwelveudash}
326         {\tc@check@symbol{9}\texttwelveudash}
327 \DeclareTextCommandDefault{\texttwosuperior}
328         {\tc@check@symbol{9}\texttwosuperior}
329 \DeclareTextCommandDefault{\textbardbl}
330         {\tc@check@symbol{9}\textbardbl}
```

2 Unicode engine specials

If we are using a unicode engine we handle some glyphs differently, so this here are the definitions for the Unicode encoding (overwriting the defaults above).

```
331 \ifx \Umathcode\@undefined \else
```

This set should be taken from TS1 encoding even if it means you get it from the default font for that encoding.

```
332 %\DeclareTextSymbol{\textcopyleft}{TS1}{171}
333 %\DeclareTextSymbol{\textdblhyphen}{TS1}{45}
334 %\DeclareTextSymbol{\textdblhyphenchar}{TS1}{127}
335 %\DeclareTextSymbol{\textquotestraightbase}{TS1}{13}
336 %\DeclareTextSymbol{\textquotestraightdblbase}{TS1}{18}
337 %\DeclareTextSymbol{\textleaf}{TS1}{108}
338 %\DeclareTextSymbol{\texttwelveudash}{TS1}{21}
339 %\DeclareTextSymbol{\textthreequartersemdash}{TS1}{22}
```

If oldstyle numerals are asked for we just use `\oldstylenums`.

```
340 \DeclareTextCommand{\textzerooldstyle} \UnicodeEncodingName{\oldstylenums{0}}
341 \DeclareTextCommand{\textoneoldstyle} \UnicodeEncodingName{\oldstylenums{1}}
342 \DeclareTextCommand{\texttwooldstyle} \UnicodeEncodingName{\oldstylenums{2}}
343 \DeclareTextCommand{\textthreeoldstyle} \UnicodeEncodingName{\oldstylenums{3}}
344 \DeclareTextCommand{\textfouroldstyle} \UnicodeEncodingName{\oldstylenums{4}}
345 \DeclareTextCommand{\textfiveoldstyle} \UnicodeEncodingName{\oldstylenums{5}}
346 \DeclareTextCommand{\textsixoldstyle} \UnicodeEncodingName{\oldstylenums{6}}
347 \DeclareTextCommand{\textsevenoldstyle} \UnicodeEncodingName{\oldstylenums{7}}
348 \DeclareTextCommand{\texteightoldstyle} \UnicodeEncodingName{\oldstylenums{8}}
349 \DeclareTextCommand{\textnineoldstyle} \UnicodeEncodingName{\oldstylenums{9}}
```

These have Unicode slots so this should be integrated into TU explicitly

```
350 \DeclareTextSymbol{\textpilcrow} \UnicodeEncodingName{"00B6}
351 \DeclareTextSymbol{\textborn} \UnicodeEncodingName{"002A}
352 \DeclareTextSymbol{\textdied} \UnicodeEncodingName{"2020}
353 \DeclareTextSymbol{\textlbrackdbl} \UnicodeEncodingName{"27E6}
354 \DeclareTextSymbol{\textrbrackdbl} \UnicodeEncodingName{"27E7}
355 \DeclareTextSymbol{\textguarani} \UnicodeEncodingName{"20B2}
```

We could make `\textcentoldstyle` and `\textdollaroldstyle` point to dollar and cent in the Unicode encoding

```
356 %\DeclareTextSymbol{\textcentoldstyle} \UnicodeEncodingName{"00A2}
357 %\DeclareTextSymbol{\textdollaroldstyle}\UnicodeEncodingName{"0024}
```

but I think it is better to pick them up from TS1 even if that usually means LMR fonts

```
358 \DeclareTextSymbol{\textdollaroldstyle}{TS1}{138}
359 \DeclareTextSymbol{\textcentoldstyle} {TS1}{139}
360 \fi % --- END of Unicode engines specials
```

3 Font family sub-encodings setup

We declare the subsets for a good number of fonts in the kernel ...

But first the default for anything that is not declared. We use 9 which is most likely much too conservative, but with the advantage that we aren't getting missing glyphs (or at least that this is very unlikely). For nearly all font in the T_EX Live distribution of 2019 "correct" classifications are given below, so that this default is only used for new font families, and over time the right classifications can be added here too.

```
361 \DeclareEncodingSubset{TS1}{?}{9}
```

This first block contains the fonts that have been already supported by the `textcomp` package way back, i.e., the font families that have T_EX support since the mid-nineties.

```
362 \DeclareEncodingSubset{TS1}{ccr} {0}
363 \DeclareEncodingSubset{TS1}{cmbr} {0}
```

The following 4 declarations are now part of the corresponding `.fd` file, hopefully other will follow so that this list of declarations can eventually be removed from the kernel (where it doesn't belong).

```
364 %\DeclareEncodingSubset{TS1}{cmr} {0}
365 %\DeclareEncodingSubset{TS1}{cmss} {0}
366 %\DeclareEncodingSubset{TS1}{cmtt} {0}
367 %\DeclareEncodingSubset{TS1}{cmvtt} {0}

368 \DeclareEncodingSubset{TS1}{cmtl} {0}
369 \DeclareEncodingSubset{TS1}{pxr} {0}
370 \DeclareEncodingSubset{TS1}{pxss} {0}
371 \DeclareEncodingSubset{TS1}{pxtt} {0}
372 \DeclareEncodingSubset{TS1}{qag} {0}
373 \DeclareEncodingSubset{TS1}{qbk} {0}
374 \DeclareEncodingSubset{TS1}{qcr} {0}
375 \DeclareEncodingSubset{TS1}{qcs} {0}
376 \DeclareEncodingSubset{TS1}{qhvc} {0}
377 \DeclareEncodingSubset{TS1}{qhv} {0}
378 \DeclareEncodingSubset{TS1}{qpl} {0}
379 \DeclareEncodingSubset{TS1}{qtm} {0}
380 \DeclareEncodingSubset{TS1}{qzc} {0}
381 \DeclareEncodingSubset{TS1}{txr} {0}
382 \DeclareEncodingSubset{TS1}{txss} {0}
383 \DeclareEncodingSubset{TS1}{txtt} {0}

384 \DeclareEncodingSubset{TS1}{lmr} {1}
385 \DeclareEncodingSubset{TS1}{lmdh} {1}
386 \DeclareEncodingSubset{TS1}{lmss} {1}
387 \DeclareEncodingSubset{TS1}{lmssq} {1}
```



```
388 \DeclareEncodingSubset{TS1}{lmvtt} {1}
```

The lmtt family is missing TM, SM, and perthousand for some reason, so the first safe sub-encoding would be 2, but that is then missing out a huge number of glyphs that are available, so we claim it is sub-encoding 1 even if this can lead to missing glyphs.

```
389 \DeclareEncodingSubset{TS1}{lmtt} {1} % missing TM, SM and pertenthousand
```

The next three families have been removed from TeX Live, but we keep the definitions

```
390 \DeclareEncodingSubset{TS1}{ptmx} {2}
```

```
391 \DeclareEncodingSubset{TS1}{ptmj} {2}
```

```
392 \DeclareEncodingSubset{TS1}{ul8} {2}
```

The next set are the early PostScript font implementations, these days there are better alternatives, but . . . Note that, their virtual fonts contain a lot of “tofu” in form of black squares, thus they don’t even give a missing character warning if you select such a glyph. This is why they are set as sub-encoding 5.

```
393 \DeclareEncodingSubset{TS1}{bch} {5} % tofu for blank, ohm
```

```
394 \DeclareEncodingSubset{TS1}{futj}{5} % tofu for blank, interrobang/down, ohm
```

```
395 \DeclareEncodingSubset{TS1}{futs}{5} % tofu for blank, ohm
```

```
396 \DeclareEncodingSubset{TS1}{futx}{5} % probably (currently broken distrib)
```

```
397 \DeclareEncodingSubset{TS1}{pag} {5} % tofu for blank, interrobang/down, ohm
```

```
398 \DeclareEncodingSubset{TS1}{pbk} {5} % tofu for blank, interrobang/down, ohm
```

```
399 \DeclareEncodingSubset{TS1}{pcr} {5} % tofu for blank, interrobang/down, ohm
```

```
400 \DeclareEncodingSubset{TS1}{phv} {5} % tofu for blank, interrobang/down, ohm
```

```
401 \DeclareEncodingSubset{TS1}{pnc} {5} % tofu for blank, interrobang/down, ohm
```

```
402 \DeclareEncodingSubset{TS1}{pplj}{5} % tofu for blank
```

```
403 \DeclareEncodingSubset{TS1}{pplx}{5} % tofu for blank
```

```
404 \DeclareEncodingSubset{TS1}{ppl} {5} % tofu for blank interrobang/down
```

```
405 \DeclareEncodingSubset{TS1}{ptm} {5} % tofu for blank, interrobang/down, ohm
```

```
406 \DeclareEncodingSubset{TS1}{pzc} {5} % tofu for blank, interrobang/down, ohm
```

```
407 \DeclareEncodingSubset{TS1}{ul9} {5} % tofu for blank, interrobang/down, ohm
```

The next set suffers from the same problem and they contain even fewer real glyphs.

```
408 \DeclareEncodingSubset{TS1}{dayroms}{6} % tofu for blank, interrobang/down, ohm
```

```
409 \DeclareEncodingSubset{TS1}{dayrom} {6} % tofu for blank, interrobang/down, ohm
```

```
410 \DeclareEncodingSubset{TS1}{augie}{8} % really only missing euro
```

```
411 \DeclareEncodingSubset{TS1}{put} {8}
```

```
412 \DeclareEncodingSubset{TS1}{uag} {8} % probably (currently broken distrib)
```

```
413 \DeclareEncodingSubset{TS1}{ugq} {8}
```

```
414 \DeclareEncodingSubset{TS1}{zi4} {9}
```

LucidaBright (sold through TUG) probably not quite correct, I guess as I have the older fonts . . .

```
415 \DeclareEncodingSubset{TS1}{hls} {5}
```

```
416 \DeclareEncodingSubset{TS1}{hlst} {5}
```

```
417 \DeclareEncodingSubset{TS1}{hlct} {5}
```

```
418 \DeclareEncodingSubset{TS1}{hlh} {5}
```

```
419 \DeclareEncodingSubset{TS1}{hlx} {8}
```

```
420 \DeclareEncodingSubset{TS1}{hlce} {8}
```

```
421 \DeclareEncodingSubset{TS1}{hlcn} {8}
```

```
422 \DeclareEncodingSubset{TS1}{hlcw} {8}
```

```
423 \DeclareEncodingSubset{TS1}{hlcf} {8}
```

Below are the newer fonts that have support files for L^AT_EX. With very few exceptions the classifications are done so that all characters are correctly produced (either being available in the font or substituted).

There are a few fonts that contain “tofu” squares in places (instead of a real glyph) and in a few cases some really seldom needed chars are unavailable, i.e., produce missing glyphs (to avoid that a large number of available chars are unnecessarily substituted).

Encoding declarations for these font families shouldn’t really be in the kernel, but part of the .fd files for the family. When we introduced the concept in 2021 we had some hope that this would happen over time and that we could take the declarations out—after all it is nearly impossible to maintain it correctly in the kernel, given that fonts may get new glyphs added (happened for several of them in the recent year) which is something we wouldn’t notice. However, so far this hasn’t happened, so in 2024, I went through the current set and adjusted the declarations in several places.

Next four are wrong and still need adjustment:

```

424 \DeclareEncodingSubset{TS1}{lato-*}      {0} % with a bunch of tofu inside
425 \DeclareEncodingSubset{TS1}{opensans-*} {0} % with a bunch of tofu inside
426 \DeclareEncodingSubset{TS1}{cantarell-*} {0} % with a bunch of tofu inside
427 \DeclareEncodingSubset{TS1}{tli}        {1} % with lots of tofu inside

428 \DeclareEncodingSubset{TS1}{fbb-*}      {2} % missing centoldstyle

429 \DeclareEncodingSubset{TS1}{Alegreya-*}      {2}
430 \DeclareEncodingSubset{TS1}{AlegreyaSans-*}  {2}
431 \DeclareEncodingSubset{TS1}{BaskervilleF-*}  {2}
432 \DeclareEncodingSubset{TS1}{DejaVuSans-TLF}  {2}
433 \DeclareEncodingSubset{TS1}{DejaVuSansCondensed-TLF} {2}

```

Next one is missing \textfractionsolidus but is otherwise completely sub-encoding 2 so we use that sub-encoding.

```

434 \DeclareEncodingSubset{TS1}{DejaVuSansMono-TLF}      {2}

435 \DeclareEncodingSubset{TS1}{EBGaramond-*}            {2}
436 \DeclareEncodingSubset{TS1}{Merriwthr-OsF}           {2}
437 \DeclareEncodingSubset{TS1}{MerriwthrSans-OsF}       {2}
438 \DeclareEncodingSubset{TS1}{Montserrat-*}            {2}
439 \DeclareEncodingSubset{TS1}{MontserratAlternates-*}  {2}
440 \DeclareEncodingSubset{TS1}{NotoSansMono-TLF}        {2}
441 \DeclareEncodingSubset{TS1}{NotoSansMono-TOsF}       {2}
442 \DeclareEncodingSubset{TS1}{Tempora-TLF}             {2}
443 \DeclareEncodingSubset{TS1}{Tempora-TOsF}            {2}
444 \DeclareEncodingSubset{TS1}{XCharter-TLF}            {2}
445 \DeclareEncodingSubset{TS1}{XCharter-TOsF}           {2}
446 \DeclareEncodingSubset{TS1}{erewhon-*}              {2}

447 \DeclareEncodingSubset{TS1}{Arimo-TLF}              {3}
448 \DeclareEncodingSubset{TS1}{Crlt-*}                 {3}
449 \DeclareEncodingSubset{TS1}{IBMPlexMono-TLF}        {3}
450 \DeclareEncodingSubset{TS1}{IBMPlexSans-TLF}        {3}
451 \DeclareEncodingSubset{TS1}{IBMPlexSerif-TLF}       {3}
452 \DeclareEncodingSubset{TS1}{SourceCodePro-TLF}      {3}
453 \DeclareEncodingSubset{TS1}{SourceCodePro-TOsF}     {3}
454 \DeclareEncodingSubset{TS1}{SourceSansPro-*}        {3}
455 \DeclareEncodingSubset{TS1}{SourceSerifPro-*}       {3}
456 \DeclareEncodingSubset{TS1}{Tinos-TLF}              {3}

```

```

457 \DeclareEncodingSubset{TS1}{AccanthisADFStdNoThree-LF}{4}
458 \DeclareEncodingSubset{TS1}{Cabin-TLF} {4}
459 \DeclareEncodingSubset{TS1}{Caladea-TLF} {4}
460 \DeclareEncodingSubset{TS1}{Chivo-*} {4}
461 \DeclareEncodingSubset{TS1}{ClearSans-TLF} {4}
462 \DeclareEncodingSubset{TS1}{Coelacanth-LF} {4}
463 \DeclareEncodingSubset{TS1}{CrimsonPro-*} {4}
464 \DeclareEncodingSubset{TS1}{FiraMono-TLF} {4}
465 \DeclareEncodingSubset{TS1}{FiraMono-TOfS} {4}
466 \DeclareEncodingSubset{TS1}{FiraSans-*} {4}
467 \DeclareEncodingSubset{TS1}{Go-TLF} {4}
468 \DeclareEncodingSubset{TS1}{GoMono-TLF} {4}
469 \DeclareEncodingSubset{TS1}{InriaSans-*} {4}
470 \DeclareEncodingSubset{TS1}{InriaSerif-*} {4}
471 \DeclareEncodingSubset{TS1}{LibertinusSans-*} {4}
472 \DeclareEncodingSubset{TS1}{LibertinusSerif-*} {4}
473 \DeclareEncodingSubset{TS1}{LibreBodoni-TLF} {4}
474 \DeclareEncodingSubset{TS1}{LibreFranklin-TLF} {4}
475 \DeclareEncodingSubset{TS1}{LinguisticsPro-LF} {4}
476 \DeclareEncodingSubset{TS1}{LinguisticsPro-OfS} {4}
477 \DeclareEncodingSubset{TS1}{LinuxBiolinumT-*} {4}
478 \DeclareEncodingSubset{TS1}{LinuxLibertineT-*} {4}
479 \DeclareEncodingSubset{TS1}{MintSpirit-*} {4}
480 \DeclareEncodingSubset{TS1}{MintSpiritNoTwo-*} {4}
481 \DeclareEncodingSubset{TS1}{PTMono-TLF} {4}
482 \DeclareEncodingSubset{TS1}{PTSans-TLF} {4}
483 \DeclareEncodingSubset{TS1}{PTSansCaption-TLF} {4}
484 \DeclareEncodingSubset{TS1}{PTSansNarrow-TLF} {4}
485 \DeclareEncodingSubset{TS1}{PTSerif-TLF} {4}
486 \DeclareEncodingSubset{TS1}{PTSerifCaption-TLF} {4}
487 \DeclareEncodingSubset{TS1}{Raleway-TLF} {4}
488 \DeclareEncodingSubset{TS1}{Raleway-TOfS} {4}
489 \DeclareEncodingSubset{TS1}{Roboto-*} {4}
490 \DeclareEncodingSubset{TS1}{RobotoMono-TLF} {4}
491 \DeclareEncodingSubset{TS1}{RobotoSlab-TLF} {4}
492 \DeclareEncodingSubset{TS1}{Rosario-*} {4}
493 \DeclareEncodingSubset{TS1}{SticksTooText-*} {4}
494 \DeclareEncodingSubset{TS1}{UniversalisADFStd-LF} {4}

495 \DeclareEncodingSubset{TS1}{Almndr-OfS} {5}
496 \DeclareEncodingSubset{TS1}{Baskervaldx-*} {5}
497 \DeclareEncodingSubset{TS1}{Bttr-TLF} {5}
498 \DeclareEncodingSubset{TS1}{Cinzel-LF} {5}
499 \DeclareEncodingSubset{TS1}{CinzelDecorative-LF} {5}
500 \DeclareEncodingSubset{TS1}{Cochineal-*} {5}
501 \DeclareEncodingSubset{TS1}{DejaVuSerif-TLF} {5}
502 \DeclareEncodingSubset{TS1}{DejaVuSerifCondensed-TLF} {5}
503 \DeclareEncodingSubset{TS1}{GilliusADF-LF} {5}
504 \DeclareEncodingSubset{TS1}{GilliusADFCond-LF} {5}
505 \DeclareEncodingSubset{TS1}{GilliusADFNoTwo-LF} {5}
506 \DeclareEncodingSubset{TS1}{GilliusADFNoTwoCond-LF} {5}
507 \DeclareEncodingSubset{TS1}{OldStandard-TLF} {5}
508 \DeclareEncodingSubset{TS1}{PlyfrDisplay-TLF} {5}
509 \DeclareEncodingSubset{TS1}{PlyfrDisplay-TOfS} {5}
510 \DeclareEncodingSubset{TS1}{TheanoDidot-TLF} {5}

```

```

511 \DeclareEncodingSubset{TS1}{TheanoDidot-TOfF} {5}
512 \DeclareEncodingSubset{TS1}{TheanoModern-TLF} {5}
513 \DeclareEncodingSubset{TS1}{TheanoModern-TOfF} {5}
514 \DeclareEncodingSubset{TS1}{TheanoOldStyle-TLF} {5}
515 \DeclareEncodingSubset{TS1}{TheanoOldStyle-TOfF} {5}
516 \DeclareEncodingSubset{TS1}{charssil-TLF} {5}

517 \DeclareEncodingSubset{TS1}{Crimson-TLF} {6}
518 \DeclareEncodingSubset{TS1}{LibertinusSerifDisplay-LF} {6}
519 \DeclareEncodingSubset{TS1}{LinuxLibertineDisplayT-*} {6}
520 \DeclareEncodingSubset{TS1}{LinuxLibertineMonoT-LF} {6}
521 \DeclareEncodingSubset{TS1}{LinuxLibertineMonoT-TLF} {6}
522 \DeclareEncodingSubset{TS1}{Ovrlck-LF} {6}

523 \DeclareEncodingSubset{TS1}{ComicNeue-TLF} {7}
524 \DeclareEncodingSubset{TS1}{ComicNeueAngular-TLF} {7}
525 \DeclareEncodingSubset{TS1}{CormorantGaramond-*} {7}
526 \DeclareEncodingSubset{TS1}{Heuristica-TLF} {7}
527 \DeclareEncodingSubset{TS1}{Heuristica-TOfF} {7}
528 \DeclareEncodingSubset{TS1}{IMFELLEnglish-TLF} {7}
529 \DeclareEncodingSubset{TS1}{LibreBskrv1-TLF} {7}
530 \DeclareEncodingSubset{TS1}{LibreCsln-*} {7}
531 \DeclareEncodingSubset{TS1}{Lbstr-LF} {7}
532 \DeclareEncodingSubset{TS1}{Mrcls-LF} {7}

```

Strangely enough NotoSerif and NotoSans are sub-encoding 7 as they are missing `\textminus` and several other glyphs. In contrast, the NotoSansMono is far more complete.

```

533 \DeclareEncodingSubset{TS1}{NotoSans-*} {7}
534 \DeclareEncodingSubset{TS1}{NotoSerif-*} {7}
535 \DeclareEncodingSubset{TS1}{Quattro-LF} {7}
536 \DeclareEncodingSubset{TS1}{QuattroSans-LF} {7}
537 \DeclareEncodingSubset{TS1}{Frm-LF} {7} % the superiors are missing

538 \DeclareEncodingSubset{TS1}{LibertinusMono-TLF} {8}
539 \DeclareEncodingSubset{TS1}{AlgolRevived-TLF} {9}

```

4 Legacy symbol support for lists and footnote symbols

`\UseLegacyTextSymbols`

```

540 \def\UseLegacyTextSymbols{%
541   \DeclareTextSymbolDefault{\textasteriskcentered}{OMS}%
542   \DeclareTextSymbolDefault{\textbardbl}{OMS}%
543   \DeclareTextSymbolDefault{\textbullet}{OMS}%
544   \DeclareTextSymbolDefault{\textdaggerdbl}{OMS}%
545   \DeclareTextSymbolDefault{\textdagger}{OMS}%
546   \DeclareTextSymbolDefault{\textparagraph}{OMS}%
547   \DeclareTextSymbolDefault{\textperiodcentered}{OMS}%
548   \DeclareTextSymbolDefault{\textsection}{OMS}%
549   \UndeclareTextCommand{\textsection}{T1}%
550   \expandafter\let\csname oldstylenums \expandafter\endcsname
551     \csname legacyoldstylenums \endcsname
552 }

```

(End of definition for `\UseLegacyTextSymbols`.)

`\textlegacyasteriskcentered` Here are new names for the legacy symbols that L^AT_EX used to pick up from the OMS
`\textlegacybardbl` encoded fonts (and used for itemize lists or footnote symbols).

We go the roundabout way via separate OMS declarations so that

```
\renewcommand\textbullet{\textlegacybullet}
```

doesn't produce an endless loop.

```
\textlegacybullet
\textlegacydaggerdbl
\textlegacydagger
\textlegacyparagraph
\textlegacyperiodcentered
\textlegacysection
553 \DeclareTextSymbol{\textlegacyasteriskcentered}{OMS}{3} % "03
554 \DeclareTextSymbol{\textlegacybardbl}{OMS}{107} % "6B
555 \DeclareTextSymbol{\textlegacybullet}{OMS}{15} % "0F
556 \DeclareTextSymbol{\textlegacydaggerdbl}{OMS}{122} % "7A
557 \DeclareTextSymbol{\textlegacydagger}{OMS}{121} % "79
558 \DeclareTextSymbol{\textlegacyparagraph}{OMS}{123} % "7B
559 \DeclareTextSymbol{\textlegacyperiodcentered}{OMS}{1} % "01
560 \DeclareTextSymbol{\textlegacysection}{OMS}{120} % "78

561 \DeclareTextSymbolDefault{\textlegacyasteriskcentered}{OMS}
562 \DeclareTextSymbolDefault{\textlegacybardbl}{OMS}
563 \DeclareTextSymbolDefault{\textlegacybullet}{OMS}
564 \DeclareTextSymbolDefault{\textlegacydaggerdbl}{OMS}
565 \DeclareTextSymbolDefault{\textlegacydagger}{OMS}
566 \DeclareTextSymbolDefault{\textlegacyparagraph}{OMS}
567 \DeclareTextSymbolDefault{\textlegacyperiodcentered}{OMS}
568 \DeclareTextSymbolDefault{\textlegacysection}{OMS}
```

(End of definition for `\textlegacyasteriskcentered` and others.)

Supporting rollback ...

```
569 </2ekernel | latexrelease>
570 <latexrelease>
571 <latexrelease>\IncludeInRelease{0000/00/00}%
572 <latexrelease> {ltxtextcomp}{\Undefine text companion symbols}%
573 <latexrelease>
574 <latexrelease>\DeclareRobustCommand\oldstylenums[1]{%
575 <latexrelease> \begingroup
576 <latexrelease> \spaceskip\fontdimen\tw@font
577 <latexrelease> \usefont{OML}{\rmdefault}{\f@series}{it}%
578 <latexrelease> \mathgroup\symletters #1%
579 <latexrelease> \endgroup
580 <latexrelease>}
581 <latexrelease>\let\legacyoldstylenums\@undefined
582 <latexrelease>\def\textcompsubstdefault{cmr}
583 <latexrelease>
584 <latexrelease>\let\DeclareEncodingSubset\@undefined
585 <latexrelease>\let\CheckEncodingSubset\@undefined
586 <latexrelease>
587 <latexrelease>\DeclareTextSymbolDefault{\textdollar}{OT1}
588 <latexrelease>\DeclareTextSymbolDefault{\textsterling}{OT1}
589 <latexrelease>\DeclareTextCommand{\textdollar}{OT1}{\hmode@bgroup
590 <latexrelease> \ifdim \fontdimen\@ne\font >\z@
591 <latexrelease> \slshape
592 <latexrelease> \else
593 <latexrelease> \upshape
594 <latexrelease> \fi
```

```

595 <latexrelease> \char'{$\egroup}
596 <latexrelease>\DeclareTextCommand{\textsterling}{OT1}{\hmode@bgroup
597 <latexrelease> \ifdim \fontdimen\@ne\font >\z@
598 <latexrelease> \itshape
599 <latexrelease> \else
600 <latexrelease> \fontshape{ui}\selectfont
601 <latexrelease> \fi
602 <latexrelease> \char'{$\egroup}
603 <latexrelease>\DeclareTextCommand{\textperthousand}{T1}
604 <latexrelease> {\%\char 24 }
605 <latexrelease>
606 <latexrelease>\DeclareTextSymbolDefault{\textasteriskcentered}{OMS}
607 <latexrelease>\DeclareTextSymbolDefault{\textbullet}{OMS}
608 <latexrelease>\DeclareTextSymbolDefault{\textdaggerdbl}{OMS}
609 <latexrelease>\DeclareTextSymbolDefault{\textdagger}{OMS}
610 <latexrelease>\DeclareTextSymbolDefault{\textparagraph}{OMS}
611 <latexrelease>\DeclareTextSymbolDefault{\textperiodcentered}{OMS}
612 <latexrelease>\DeclareTextSymbolDefault{\textsection}{OMS}
613 <latexrelease>
614 <latexrelease>\DeclareTextSymbolDefault{\textbardbl}{OMS}
615 <latexrelease>\let\textbrokenbar\@undefined
616 <latexrelease>\let\textcelsius\@undefined
617 <latexrelease>\let\textcent\@undefined
618 <latexrelease>\DeclareTextCommandDefault{\textcopyright}
619 <latexrelease> {\textcircled{c}}
620 <latexrelease>\let\textdegree\@undefined
621 <latexrelease>\let\textdiv\@undefined
622 <latexrelease>\let\textlnot\@undefined
623 <latexrelease>\let\textonehalf\@undefined
624 <latexrelease>\let\textonequarter\@undefined
625 <latexrelease>\let\textonesuperior\@undefined
626 <latexrelease>\DeclareTextCommandDefault{\textordfeminine}
627 <latexrelease> {\textsuperscript{a}}
628 <latexrelease>\DeclareTextCommandDefault{\textordmasculine}
629 <latexrelease> {\textsuperscript{o}}
630 <latexrelease>\let\textpm\@undefined
631 <latexrelease>\let\textquotesingle\@undefined
632 <latexrelease>\let\textquotestraightbase\@undefined
633 <latexrelease>\let\textquotestraightdblbase\@undefined
634 <latexrelease>\DeclareTextCommandDefault{\textregistered}
635 <latexrelease> {\textcircled{%
636 <latexrelease> \check@mathfonts\fontsize\sf@size\z@
637 <latexrelease> \math@fontsfalse\selectfont R}}
638 <latexrelease>\let\textthreequartersemdash\@undefined
639 <latexrelease>\let\textthreequarters\@undefined
640 <latexrelease>\let\textthreesuperior\@undefined
641 <latexrelease>\let\texttimes\@undefined
642 <latexrelease>\DeclareTextCommandDefault{\texttrademark}
643 <latexrelease> {\textsuperscript{TM}}
644 <latexrelease>\let\texttwelveudash\@undefined
645 <latexrelease>\let\texttwosuperior\@undefined
646 <latexrelease>\let\textyen\@undefined
647 <latexrelease>
648 <latexrelease>\let\textcapitalcompwordmark\@undefined

```

```

649 <latexrelease>\let\textascendercompwordmark\@undefined
650 <latexrelease>
651 <latexrelease>\DeclareTextAccentDefault{\textcircled}{OMS}
652 <latexrelease>\DeclareTextAccentDefault{\t}{OML}
653 <latexrelease>
654 <latexrelease>\let\capitalacute\@undefined
655 <latexrelease>\let\capitalbreve\@undefined
656 <latexrelease>\let\capitalcaron\@undefined
657 <latexrelease>\let\capitalcedilla\@undefined
658 <latexrelease>\let\capitalcircumflex\@undefined
659 <latexrelease>\let\capitaldieresisis\@undefined
660 <latexrelease>\let\capitaldotaccent\@undefined
661 <latexrelease>\let\capitalgrave\@undefined
662 <latexrelease>\let\capitalhungarumlaut\@undefined
663 <latexrelease>\let\capitalmacron\@undefined
664 <latexrelease>\let\capitalnewtie\@undefined
665 <latexrelease>\let\capitalogonek\@undefined
666 <latexrelease>\let\capitalring\@undefined
667 <latexrelease>\let\capitaltie\@undefined
668 <latexrelease>\let\capitaltilde\@undefined
669 <latexrelease>\let\newtie\@undefined
670 <latexrelease>
671 <latexrelease>\let\textlbrackdbl\@undefined
672 <latexrelease>\let\textrbrackdbl\@undefined
673 <latexrelease>
674 <latexrelease>\let\texteightoldstyle\@undefined
675 <latexrelease>\let\textfiveoldstyle\@undefined
676 <latexrelease>\let\textfouroldstyle\@undefined
677 <latexrelease>\let\textnineoldstyle\@undefined
678 <latexrelease>\let\textoneoldstyle\@undefined
679 <latexrelease>\let\textsevenoldstyle\@undefined
680 <latexrelease>\let\textsixoldstyle\@undefined
681 <latexrelease>\let\textthreeoldstyle\@undefined
682 <latexrelease>\let\texttwooldstyle\@undefined
683 <latexrelease>\let\textzerooldstyle\@undefined
684 <latexrelease>
685 <latexrelease>\let\textacutedbl\@undefined
686 <latexrelease>\let\textasciiacute\@undefined
687 <latexrelease>\let\textasciibreve\@undefined
688 <latexrelease>\let\textasciicaron\@undefined
689 <latexrelease>\let\textasciidieresis\@undefined
690 <latexrelease>\let\textasciigrave\@undefined
691 <latexrelease>\let\textasciimacron\@undefined
692 <latexrelease>\let\textgravedbl\@undefined
693 <latexrelease>\let\texttildelow\@undefined
694 <latexrelease>
695 <latexrelease>\let\textbaht\@undefined
696 <latexrelease>\let\textbigcircle\@undefined
697 <latexrelease>\let\textborn\@undefined
698 <latexrelease>\let\textcentoldstyle\@undefined
699 <latexrelease>\let\textcircledP\@undefined
700 <latexrelease>\let\textcopyleft\@undefined
701 <latexrelease>\let\textdblhyphenchar\@undefined
702 <latexrelease>\let\textdblhyphen\@undefined

```

```

703 <latexrelease>\let\textdied\@undefined
704 <latexrelease>\let\textdiscount\@undefined
705 <latexrelease>\let\textdivorced\@undefined
706 <latexrelease>\let\textdollaroldstyle\@undefined
707 <latexrelease>\let\textguarani\@undefined
708 <latexrelease>\let\textleaf\@undefined
709 <latexrelease>\let\textlquill\@undefined
710 <latexrelease>\let\textmarried\@undefined
711 <latexrelease>\let\textmho\@undefined
712 <latexrelease>\let\textmusicalnote\@undefined
713 <latexrelease>\let\textnaira\@undefined
714 <latexrelease>\let\textopenbullet\@undefined
715 <latexrelease>\let\textpeso\@undefined
716 <latexrelease>\let\textpilcrow\@undefined
717 <latexrelease>\let\textrecipe\@undefined
718 <latexrelease>\let\textreferencemark\@undefined
719 <latexrelease>\let\textrquill\@undefined
720 <latexrelease>\let\textservicemark\@undefined
721 <latexrelease>\let\textsurd\@undefined
722 <latexrelease>
723 <latexrelease>\DeclareTextCommand{\textpertenthousand}{T1}
724 <latexrelease>          {\%\char 24\char 24 }
725 <latexrelease>
726 <latexrelease>\let\textlangle\@undefined
727 <latexrelease>\let\textrangle\@undefined
728 <latexrelease>
729 <latexrelease>\let\textcolonmonetary\@undefined
730 <latexrelease>\let\textdong\@undefined
731 <latexrelease>\let\textdownarrow\@undefined
732 <latexrelease>\let\textleftarrow\@undefined
733 <latexrelease>\let\textlira\@undefined
734 <latexrelease>\let\textrightarrow\@undefined
735 <latexrelease>\let\textuparrow\@undefined
736 <latexrelease>\let\textwon\@undefined
737 <latexrelease>
738 <latexrelease>\let\textestimated\@undefined
739 <latexrelease>\let\textnumero\@undefined
740 <latexrelease>
741 <latexrelease>\let\textflorin\@undefined
742 <latexrelease>\let\textcurrency\@undefined
743 <latexrelease>
744 <latexrelease>\let\textfractionsolidus\@undefined
745 <latexrelease>\let\textohm\@undefined
746 <latexrelease>\let\textmu\@undefined
747 <latexrelease>\let\textminus\@undefined
748 <latexrelease>
749 <latexrelease>\let\textblank\@undefined
750 <latexrelease>\let\textinterrobangdown\@undefined
751 <latexrelease>\let\textinterrobang\@undefined
752 <latexrelease>
753 <latexrelease>\let\texteuro\@undefined
754 <latexrelease>
755 <latexrelease>\let\textcelsius\@undefined
756 <latexrelease>\let\textonesuperior\@undefined

```



```

757 <latexrelease>\let\textthreequartersemdash\@undefined
758 <latexrelease>\let\textthreesuperior\@undefined
759 <latexrelease>\let\texttwelveudash\@undefined
760 <latexrelease>\let\texttwosuperior\@undefined
761 <latexrelease>\let\textbardbl\@undefined
762 <latexrelease>
763 <latexrelease>\let\UseLegacyTextSymbols\@undefined
764 <latexrelease>\let\textlegacystariskcentered\@undefined
765 <latexrelease>\let\textlegacybardbl\@undefined
766 <latexrelease>\let\textlegacybullet\@undefined
767 <latexrelease>\let\textlegacydaggerdbl\@undefined
768 <latexrelease>\let\textlegacydagger\@undefined
769 <latexrelease>\let\textlegacyparagraph\@undefined
770 <latexrelease>\let\textlegacyperiodcentered\@undefined
771 <latexrelease>\let\textlegacysection\@undefined
772 <latexrelease>
773 <latexrelease>\EndModuleRelease

```

5 The textcomp package

For any rollback request before 2018-08-11 we make an attempt by loading the 2018 version.

```

774 <*TS1sty>
775 \DeclareRelease{}{1997-12-01}{textcomp-2018-08-11.sty}
776 \DeclareRelease{}{2018-08-11}{textcomp-2018-08-11.sty}
777 \DeclareCurrentRelease{}{2020-02-02}
778
779 \ProvidesPackage{textcomp}
780 [2024/04/24 v2.1b Standard LaTeX package]
    A precaution in case this is used without rebuilding the format.
781 \NeedsTeXFormat{LaTeX2e}[2020/02/02]
    This is implemented by defining the default subset:
782 \DeclareOption{full}{\DeclareEncodingSubset{TS1}{?}{0}}
783 \DeclareOption{almostfull}{\DeclareEncodingSubset{TS1}{?}{1}}
784 \DeclareOption{euro}{\DeclareEncodingSubset{TS1}{?}{8}}
785 \DeclareOption{safe}{\DeclareEncodingSubset{TS1}{?}{9}}

```

The default is set up in the kernel is “safe” these days for unknown fonts but LaTeX has definitions for most families so it seldom applies.

If a different default is used then one needs to check the results to ensure that there aren’t “missing glyphs”.

The next set of options define the warning level (default in the kernel is info only). Using the package options you can change this behavior.

```

786 \DeclareOption{error}
787     {\gdef\tc@errorwarn{\PackageError{textcomp}}}
788 \DeclareOption{warn}
789     {\gdef\tc@errorwarn#1#2{\PackageWarning{textcomp}{#1}}}
790 \DeclareOption{info}
791     {\gdef\tc@errorwarn#1#2{\PackageInfo{textcomp}{#1}}}
792 \DeclareOption{quiet}{\gdef\tc@errorwarn#1#2{}}

```

The “force” option basically changes the sub-encoding to that of the default (which, unless changes, is 9 these days), i.e., it no longer depends on the font in use. This is mainly there because it might have been used in older documents, but not something that is recommended.

```

793 \DeclareOption{force}{%
794   \def\CheckEncodingSubset#1#2#3#4#5{%
795     \ifnum #4>%
796       0\csname #2:?\endcsname
797       \relax
798     \expandafter\@firstoftwo
799   \else
800     \expandafter\@secondoftwo
801   \fi
802   {#1{#2}}{#3}%
803   #5}%
804 }

805 \ProcessOptions\relax

```

There is not much else to do nowadays, because everything is already set up in the L^AT_EX kernel.

```

806 \InputIfFileExists{textcomp.cfg}
807   {\PackageInfo{textcomp}{Local configuration file used}}{}
808 </TS1sty>

```

5.1 The old textcomp package code

This section contains the old code for the textcomp package and its documentation. It is only used if we roll back prior to 2020. Thus all the rest is mainly for historians. Note that the old code categorized in the sub-encodings only into 6 classes not 10.

```

809 <*TS1oldsty>
810 \ProvidesPackage{textcomp}
811   [2018/08/11 v2.0j Standard LaTeX package]

```

This one is for the TS1 encoding which contains text symbols for use with the T1-encoded text fonts. It therefore first inputs the file `TS1enc.def` and then sets (or resets) the defaults for the symbols it contains. The result of this is that when one of these symbols is accessed and the current encoding does not provide it, the symbol will be supplied by a silent, local change to this encoding.

Since many PostScript fonts only implement a subset of TS1 many commands only produce black blobs of ink. To resolve the resulting problems a number of options have been introduced and some code has been developed to distinguish sub-encodings.

The sub-encodings have a numerical id and are defined as follows for TS1:

#5 those TS1 symbols that are also in the ISO-Adobe character set; without `textcurrency`, which is often misused for the Euro. Older Type1 fonts from the non-T_EX world provide only this subset.

#4 = #5 + `\texteuro`. Most newer fonts provide this.

#3 = #4 + `\textomega`. Can also be described as $TS1 \cap (ISO-Adobe \cup MacRoman)$. (Except for the missing "currency".)

#2 = **#3** + `\textestimated` + `\textcurrency`. Can also be described as `TS1` \cap `Adobe-Western-2`. This may be relevant for OpenType fonts, which usually show the `Adobe-Western-2` character set.

#1 = `TS1` without `\textcircled` and `\t`. These two glyphs are often not implemented and if their kernel defaults are changed commands like `\copyright` unnecessarily fail.

#0 = full `TS1`

And here a summary to go in the transcript file:

```

812 \PackageInfo{textcomp}{Sub-encoding information:\MessageBreak
813   \space\space 5 = only ISO-Adobe without
814                       \string\textcurrency\MessageBreak
815   \space\space 4 = 5 + \string\texteuro\MessageBreak
816   \space\space 3 = 4 + \string\textohm\MessageBreak
817   \space\space 2 = 3 + \noexpand\textestimated+
818                       \string\textcurrency\MessageBreak
819   \space\space 1 = TS1 - \noexpand\textcircled-
820                       \string\t\MessageBreak
821   \space\space 0 = TS1 (full)\MessageBreak
822   Font families with sub-encoding setting implement\MessageBreak
823   only a restricted character set as indicated.\MessageBreak
824   Family '?' is the default used for unknown fonts.\MessageBreak
825   See the documentation for details\@gobble}

```

`\DeclareEncodingSubset` An encoding subset to which a font family belongs is declared by the command `\DeclareEncodingSubset` that takes the major encoding as the first argument (e.g., `TS1`), the family name as the second argument (e.g., `cmr`), and the subset encoding id as a third, (e.g., `0` for `cmr`).

The default encoding subset to use when nothing is known about the current font family is named `?`.

```

826 \def\DeclareEncodingSubset#1#2#3{%
827   \@ifundefined{#1:#2}%
828     {\PackageInfo{textcomp}{Setting #2 sub-encoding to #1/#3}}%
829     {\PackageInfo{textcomp}{Changing #2 sub-encoding to #1/#3}}%
830   \@namedef{#1:#2}{#3}}

```

In the original code this was only allowed in the preamble but now `.fds` might contain it so that even in a rollback situation it is necessary to allow it everywhere in the document.

```
831 %\@onlypreamble\DeclareEncodingSubset
```

(End of definition for `\DeclareEncodingSubset`.)

The options for the package are the following:

safe for unknown font families enables only symbols that are also in the ISO-Adobe character set; without "currency", which is often misused for the Euro. Older Type1 fonts from the non-TeX world provide only this subset.

euro enables the "safe" symbols plus the `\texteuro` command. Most newer fonts provide this.

full enables all `TS1` commands; useful only with fonts like `EC` or `CM bright`.

almostfull same as “full”, except that `\textcircled` and `\t` are *not* redefined from their defaults to avoid that commands like `\copyright` suddenly no longer work.

force ignore all subset encoding definitions stored in the package itself or in the configuration file and always use the default subset as specified by one of the other options (seldom useful, only dangerous).

`\iftc@forced` Switch used to implement the force option

```
832 \newif\iftc@forced \tc@forcedfalse
```

(End of definition for `\iftc@forced`.)

This is implemented by defining the default subset:

```
833 \DeclareOption{full}{\DeclareEncodingSubset{TS1}{?}{0}}
```

```
834 \DeclareOption{almostfull}{\DeclareEncodingSubset{TS1}{?}{1}}
```

```
835 \DeclareOption{euro}{\DeclareEncodingSubset{TS1}{?}{4}}
```

```
836 \DeclareOption{safe}{\DeclareEncodingSubset{TS1}{?}{5}}
```

The default is “almostfull” which means that old documents will work except that `\textcircled` and `\t` will use the kernel defaults (with the advantage that this also works if the current font, as often the case, doesn’t implement these glyphs).

The “force” option simply sets the switch to true.

```
837 \DeclareOption{force}{\tc@forcedtrue}
```

The suggestions to user is to use the “safe” option always unless that balks in which case they could switch to “almostfull” but then better check their output manually.

```
838 \def\tc@errorwarn{\PackageError}
```

```
839 \DeclareOption{warn}{\gdef\tc@errorwarn#1#2#3{\PackageWarning{#1}{#2}}}
```

```
840 \DeclareOption{quiet}{\gdef\tc@errorwarn#1#2#3{}}
```

```
841 \ExecuteOptions{almostfull}
```

```
842 \ProcessOptions\relax
```

`\CheckEncodingSubset` The command `\CheckEncodingSubset` will check if the current font family has the right encoding subset to typeset a certain command. It takes five arguments as follows: first argument is either `\UseTextSymbol`, `\UseTextAccent` depending on whether or not the symbol is a text symbol or a text accent.

The second argument is the encoding from which this symbol should be fetched.

The third argument is either a fake accessor command or an error message. the code in that argument (if ever executed) receives two arguments: `#2` and `#5` of `\CheckEncodingSubset`.

Argument four is the subset encoding id to test against: if this value is higher than the subset id of the current font family then we typeset the symbol, i.e., execute `#1{#2}#5` otherwise it runs `#3#5`, e.g., to produce an error message or fake the glyph somehow.

Argument five is the symbol or accent command that is being checked.

For usage examples see definitions below.

```
843 \iftc@forced
```

If the “force” option was given we always use the default for testing against.

```
844 \def\CheckEncodingSubset#1#2#3#4#5{%
```

```
845   \ifnum #4>%
```

```
846     0\csname #2:\endcsname
```

```
847     \relax
```

```
848   \expandafter\@firstoftwo
```

```
849   \else
```

```

850 \expandafter\@secondoftwo
851 \fi
852 {#1{#2}}{#3}%
853 #5%
854 }

```

In normal circumstances the test is a bit more complicated: first check if there exists a macro `\(arg2):⟨current-family⟩` and if so use that value to test against, otherwise use the default to test against.

```

855 \else
856 \def\CheckEncodingSubset#1#2#3#4#5{%
857   \ifnum #4>%
858     \expandafter\ifx\csname #2:\f@family\endcsname\relax
859       0\csname #2:\endcsname
860     \else
861       \csname #2:\f@family\endcsname
862     \fi
863   \relax
864   \expandafter\@firstoftwo
865   \else
866     \expandafter\@secondoftwo
867   \fi
868   {#1{#2}}{#3}%
869   #5%
870 }
871 \fi

```

(End of definition for \CheckEncodingSubset.)

`\tc@subst`

```

872 \def\tc@subst#1{%
873   \tc@errorwarn{textcomp}%
874   {Symbol \string#1 not provided by\MessageBreak
875     font family \f@family\space
876     in TS1 encoding.\MessageBreak Default family used instead}\@eha
877   \bgroup\fontfamily\textcompsubstdefault\selectfont#1\egroup
878 }

```

(End of definition for \tc@subst.)

`\tc@error` `\tc@error` is going to be used in arg #3 of `\CheckEncodingSubset` when a symbol is not available in a certain font family. It gets pass the encoding it normally lives in (arg one) and the name of the symbol or accent that has a problem.

```

879 % error commands take argument:
880 % #1 symbol to be used
881 \def\tc@error#1{%
882   \PackageError{textcomp}% % should be latex error if general
883   {Accent \string#1 not provided by\MessageBreak
884     font family \f@family\space
885     in TS1 encoding}\@eha
886 }

```

(End of definition for \tc@error.)

`\tc@fake@euro` `\tc@fake@euro` is an example of a “fake” definition to use in arg #3 of `\CheckEncodingSubset` when a symbol is not available in a certain font family. Here we produce an Euro symbol by combining a “C” with a “=”.

```

887 \def\tc@fake@euro#1{%
888   \leavevmode
889   \PackageInfo{textcomp}{Faking \noexpand#1for font family
890                 \f@family\MessageBreak in TS1 encoding}%
891   \valign{##\cr
892     \vfil\hbox to 0.07em{\dimen@f@size\p@
893                       \math@fontsfalse
894                       \fontsize{.7\dimen@}\z@\selectfont=\hss}%
895     \vfil\cr%
896     \hbox{C}\crrc
897   }%
898 }

```

(End of definition for `\tc@fake@euro`.)

`\tc@check@symbol` These are two abbreviations that we use below to check symbols and accents in TS1.
`\tc@check@accent` Only there to save some space, e.g., we can then write

```

DeclareTextCommandDefault{\textcurrency}{\tc@check@symbol3\textcurrency}

```

to ensure that `\textcurrency` is only typeset if the current font has a TS1 subset id of less than 3. Otherwise `\tc@error` is called telling the user that for this font family `\textcurrency` is not available.

```

899 \def\tc@check@symbol{\CheckEncodingSubset\UseTextSymbol{TS1}\tc@subst}
900 \def\tc@check@accent{\CheckEncodingSubset\UseTextAccent{TS1}\tc@error}

```

(End of definition for `\tc@check@symbol` and `\tc@check@accent`.)

We start with the commands that are “safe” and which can be unconditionally set up, first the accents...

```

901 \DeclareTextAccentDefault{\capitalcedilla}{TS1}
902 \DeclareTextAccentDefault{\capitalogonek}{TS1}
903 \DeclareTextAccentDefault{\capitalgrave}{TS1}
904 \DeclareTextAccentDefault{\capitalacute}{TS1}
905 \DeclareTextAccentDefault{\capitalcircumflex}{TS1}
906 \DeclareTextAccentDefault{\capitaltilde}{TS1}
907 \DeclareTextAccentDefault{\capitaldieresis}{TS1}
908 \DeclareTextAccentDefault{\capitalhungarumlaut}{TS1}
909 \DeclareTextAccentDefault{\capitalring}{TS1}
910 \DeclareTextAccentDefault{\capitalcaron}{TS1}
911 \DeclareTextAccentDefault{\capitalbreve}{TS1}
912 \DeclareTextAccentDefault{\capitalmacron}{TS1}
913 \DeclareTextAccentDefault{\capitaldotaccent}{TS1}

```

... and then the other glyphs.

```

914 \DeclareTextSymbolDefault{\textcapitalcompwordmark}{TS1}
915 \DeclareTextSymbolDefault{\textascendercompwordmark}{TS1}
916 \DeclareTextSymbolDefault{\textquotestraightbase}{TS1}
917 \DeclareTextSymbolDefault{\textquotestraightdblbase}{TS1}
918 \DeclareTextSymbolDefault{\texttwelveudash}{TS1}
919 \DeclareTextSymbolDefault{\textthreequartersemdash}{TS1}
920 \DeclareTextSymbolDefault{\textdollar}{TS1}

```

```

921 \DeclareTextSymbolDefault{\textquotesingle}{TS1}
922 \DeclareTextSymbolDefault{\textasteriskcentered}{TS1}
923 \DeclareTextSymbolDefault{\textfractionsolidus}{TS1}
924 \DeclareTextSymbolDefault{\textminus}{TS1}
925 \DeclareTextSymbolDefault{\textlbrackdbl}{TS1}
926 \DeclareTextSymbolDefault{\textrbrackdbl}{TS1}
927 \DeclareTextSymbolDefault{\textasciigrave}{TS1}
928 \DeclareTextSymbolDefault{\texttildelow}{TS1}
929 \DeclareTextSymbolDefault{\textasciibreve}{TS1}
930 \DeclareTextSymbolDefault{\textasciicaron}{TS1}
931 \DeclareTextSymbolDefault{\textgravedbl}{TS1}
932 \DeclareTextSymbolDefault{\textacutedbl}{TS1}
933 \DeclareTextSymbolDefault{\textdagger}{TS1}
934 \DeclareTextSymbolDefault{\textdaggerdbl}{TS1}
935 \DeclareTextSymbolDefault{\textbardbl}{TS1}
936 \DeclareTextSymbolDefault{\textperthousand}{TS1}
937 \DeclareTextSymbolDefault{\textbullet}{TS1}
938 \DeclareTextSymbolDefault{\textcelsius}{TS1}
939 \DeclareTextSymbolDefault{\textflorin}{TS1}
940 \DeclareTextSymbolDefault{\texttrademark}{TS1}
941 \DeclareTextSymbolDefault{\textcent}{TS1}
942 \DeclareTextSymbolDefault{\textsterling}{TS1}
943 \DeclareTextSymbolDefault{\textyen}{TS1}
944 \DeclareTextSymbolDefault{\textbrokenbar}{TS1}
945 \DeclareTextSymbolDefault{\textsection}{TS1}
946 \DeclareTextSymbolDefault{\textasciidieresis}{TS1}
947 \DeclareTextSymbolDefault{\textcopyright}{TS1}
948 \DeclareTextSymbolDefault{\textordfeminine}{TS1}
949 \DeclareTextSymbolDefault{\textlnot}{TS1}
950 \DeclareTextSymbolDefault{\textregistered}{TS1}
951 \DeclareTextSymbolDefault{\textasciimacron}{TS1}
952 \DeclareTextSymbolDefault{\textdegree}{TS1}
953 \DeclareTextSymbolDefault{\textpm}{TS1}
954 \DeclareTextSymbolDefault{\texttwosuperior}{TS1}
955 \DeclareTextSymbolDefault{\textthreesuperior}{TS1}
956 \DeclareTextSymbolDefault{\textasciacute}{TS1}
957 \DeclareTextSymbolDefault{\textmu}{TS1}
958 \DeclareTextSymbolDefault{\textparagraph}{TS1}
959 \DeclareTextSymbolDefault{\textperiodcentered}{TS1}
960 \DeclareTextSymbolDefault{\textonesuperior}{TS1}
961 \DeclareTextSymbolDefault{\textordmasculine}{TS1}
962 \DeclareTextSymbolDefault{\textonequarter}{TS1}
963 \DeclareTextSymbolDefault{\textonehalf}{TS1}
964 \DeclareTextSymbolDefault{\textthreequarters}{TS1}
965 \DeclareTextSymbolDefault{\texttimes}{TS1}
966 \DeclareTextSymbolDefault{\textdiv}{TS1}

```

The `\texteuro` is only available for subsets with id 4 or less. Otherwise we fake the glyph using `\tc@fake@euro`

```

967 \DeclareTextCommandDefault{\texteuro}
968   {\CheckEncodingSubset\UseTextSymbol{TS1}\tc@fake@euro5\texteuro}

```

The `\textohm` is only available for subsets with id 3 or less. Otherwise we produce an error.

```

969 \DeclareTextCommandDefault{\textohm}{\tc@check@symbol4\textohm}

```

The `\textestimated` and `\textcurrency` are only provided for fonts with subset encoding with id 2 or less.

```
970 \DeclareTextCommandDefault{\textestimated}%  
971   {\tc@check@symbol3\textestimated}  
972 \DeclareTextCommandDefault{\textcurrency}%  
973   {\tc@check@symbol3\textcurrency}
```

Nearly all of the remaining glyphs are provided only with fonts with id 1 or 0, i.e., are essentially complete.

```
974 \DeclareTextCommandDefault{\capitaltie}%  
975   {\tc@check@accent2\capitaltie}  
976 \DeclareTextCommandDefault{\newtie}%  
977   {\tc@check@accent2\newtie}  
978 \DeclareTextCommandDefault{\capitalnewtie}%  
979   {\tc@check@accent2\capitalnewtie}  
980 \DeclareTextCommandDefault{\textleftarrow}%  
981   {\tc@check@symbol2\textleftarrow}  
982 \DeclareTextCommandDefault{\textrightarrow}%  
983   {\tc@check@symbol2\textrightarrow}  
984 \DeclareTextCommandDefault{\textblank}%  
985   {\tc@check@symbol2\textblank}  
986 \DeclareTextCommandDefault{\textdblhyphen}%  
987   {\tc@check@symbol2\textdblhyphen}  
988 \DeclareTextCommandDefault{\textzerooldstyle}%  
989   {\tc@check@symbol2\textzerooldstyle}  
990 \DeclareTextCommandDefault{\textoneoldstyle}%  
991   {\tc@check@symbol2\textoneoldstyle}  
992 \DeclareTextCommandDefault{\texttwooldstyle}%  
993   {\tc@check@symbol2\texttwooldstyle}  
994 \DeclareTextCommandDefault{\textthreeoldstyle}%  
995   {\tc@check@symbol2\textthreeoldstyle}  
996 \DeclareTextCommandDefault{\textfouroldstyle}%  
997   {\tc@check@symbol2\textfouroldstyle}  
998 \DeclareTextCommandDefault{\textfiveoldstyle}%  
999   {\tc@check@symbol2\textfiveoldstyle}  
1000 \DeclareTextCommandDefault{\textsixoldstyle}%  
1001   {\tc@check@symbol2\textsixoldstyle}  
1002 \DeclareTextCommandDefault{\textsevenoldstyle}%  
1003   {\tc@check@symbol2\textsevenoldstyle}  
1004 \DeclareTextCommandDefault{\texteightoldstyle}%  
1005   {\tc@check@symbol2\texteightoldstyle}  
1006 \DeclareTextCommandDefault{\textnineoldstyle}%  
1007   {\tc@check@symbol2\textnineoldstyle}  
1008 \DeclareTextCommandDefault{\textlangle}%  
1009   {\tc@check@symbol2\textlangle}  
1010 \DeclareTextCommandDefault{\textrightangle}%  
1011   {\tc@check@symbol2\textrightangle}  
1012 \DeclareTextCommandDefault{\textmho}%  
1013   {\tc@check@symbol2\textmho}  
1014 \DeclareTextCommandDefault{\textbigcircle}%  
1015   {\tc@check@symbol2\textbigcircle}  
1016 \DeclareTextCommandDefault{\textuparrow}%  
1017   {\tc@check@symbol2\textuparrow}  
1018 \DeclareTextCommandDefault{\textdownarrow}%
```



```

1019     {\tc@check@symbol2\textdownarrow}
1020 \DeclareTextCommandDefault{\textborn}%
1021     {\tc@check@symbol2\textborn}
1022 \DeclareTextCommandDefault{\textdivorced}%
1023     {\tc@check@symbol2\textdivorced}
1024 \DeclareTextCommandDefault{\textdied}%
1025     {\tc@check@symbol2\textdied}
1026 \DeclareTextCommandDefault{\textleaf}%
1027     {\tc@check@symbol2\textleaf}
1028 \DeclareTextCommandDefault{\textmarried}%
1029     {\tc@check@symbol2\textmarried}
1030 \DeclareTextCommandDefault{\textmusicalnote}%
1031     {\tc@check@symbol2\textmusicalnote}
1032 \DeclareTextCommandDefault{\textdblhyphenchar}%
1033     {\tc@check@symbol2\textdblhyphenchar}
1034 \DeclareTextCommandDefault{\textdollaroldstyle}%
1035     {\tc@check@symbol2\textdollaroldstyle}
1036 \DeclareTextCommandDefault{\textcentoldstyle}%
1037     {\tc@check@symbol2\textcentoldstyle}
1038 \DeclareTextCommandDefault{\textcolonmonetary}%
1039     {\tc@check@symbol2\textcolonmonetary}
1040 \DeclareTextCommandDefault{\textwon}%
1041     {\tc@check@symbol2\textwon}
1042 \DeclareTextCommandDefault{\textnaira}%
1043     {\tc@check@symbol2\textnaira}
1044 \DeclareTextCommandDefault{\textguarani}%
1045     {\tc@check@symbol2\textguarani}
1046 \DeclareTextCommandDefault{\textpeso}%
1047     {\tc@check@symbol2\textpeso}
1048 \DeclareTextCommandDefault{\textlira}%
1049     {\tc@check@symbol2\textlira}
1050 \DeclareTextCommandDefault{\textrecipe}%
1051     {\tc@check@symbol2\textrecipe}
1052 \DeclareTextCommandDefault{\textinterrobang}%
1053     {\tc@check@symbol2\textinterrobang}
1054 \DeclareTextCommandDefault{\textinterrobangdown}%
1055     {\tc@check@symbol2\textinterrobangdown}
1056 \DeclareTextCommandDefault{\textdong}%
1057     {\tc@check@symbol2\textdong}
1058 \DeclareTextCommandDefault{\textpertenthousand}%
1059     {\tc@check@symbol2\textpertenthousand}
1060 \DeclareTextCommandDefault{\textpilcrow}%
1061     {\tc@check@symbol2\textpilcrow}
1062 \DeclareTextCommandDefault{\textbaht}%
1063     {\tc@check@symbol2\textbaht}
1064 \DeclareTextCommandDefault{\textnumero}%
1065     {\tc@check@symbol2\textnumero}
1066 \DeclareTextCommandDefault{\textdiscount}%
1067     {\tc@check@symbol2\textdiscount}
1068 \DeclareTextCommandDefault{\textopenbullet}%
1069     {\tc@check@symbol2\textopenbullet}
1070 \DeclareTextCommandDefault{\textservicemark}%
1071     {\tc@check@symbol2\textservicemark}
1072 \DeclareTextCommandDefault{\textlquill}%

```

```

1073     {\tc@check@symbol2\textlquill}
1074 \DeclareTextCommandDefault{\textrquill}%
1075     {\tc@check@symbol2\textrquill}
1076 \DeclareTextCommandDefault{\textcopyleft}%
1077     {\tc@check@symbol2\textcopyleft}
1078 \DeclareTextCommandDefault{\textcircledP}%
1079     {\tc@check@symbol2\textcircledP}
1080 \DeclareTextCommandDefault{\textreferencemark}%
1081     {\tc@check@symbol2\textreferencemark}
1082 \DeclareTextCommandDefault{\textsurd}%
1083     {\tc@check@symbol2\textsurd}

```

The `\textcircled` and `\t` are handled specially, unless the current font has a subset id of 0 (i.e. full TS1) we pick the symbols up from the math font encodings, i.e., the third argument to `\CheckEncodingSubset` uses `\UseTextAccent` to get them from there.

```

1084 \DeclareTextCommandDefault{\textcircled}
1085     {\CheckEncodingSubset\UseTextAccent{TS1}%
1086     {\UseTextAccent{OMS}}1\textcircled}
1087 \DeclareTextCommandDefault{\t}
1088     {\CheckEncodingSubset\UseTextAccent{TS1}%
1089     {\UseTextAccent{OML}}1\t}

```

Finally input the encoding-specific definitions for TS1 thus making the top-level definitions optimized for this encoding (and not for the default encoding).

```

1090 \input{ts1enc.def}

```

Now having the new glyphs available we also want to make sure that they are used. For most cases this will automatically happen but for some glyphs there are inferior definitions already known to L^AT_EX which will prevent the usage of the TS1 versions. So we better get rid of them:

```

1091 \UndeclareTextCommand{\textsterling}{OT1}
1092 \UndeclareTextCommand{\textdollar} {OT1}

```

Similar declarations should probably be made for other encodings like OT4 if they are in use.

```

1093 %\UndeclareTextCommand{\textsterling}{OT4}
1094 %\UndeclareTextCommand{\textdollar} {OT4}

```

From the T1 encoding there are two candidates for removal: `%0` and `%00` since these are both constructed from `%` followed by a tiny ‘`o`’ rather than being a single glyph. The problem with this approach is that in PostScript fonts this small zero is usually not available resulting in `%■` rather than `%0` while the real glyph (at least for `\textperthousand`) is available in the PostScript version of TS1. So for the moment we compromise by removing the T1 declaration for `\textperthousand` but keeping the one for `\textpertenthousand`. This will have the effect that with Computer Modern fonts everything will come out (although `%0` and `%00` are not taken from the same physical font) and with PostScript fonts `%0` will come out correctly while `%00` will most likely look like `%■` — which is probably an improvement over just getting a single ‘`■`’ to indicate a completely missing glyph, which would happen if we also ‘undeclared’ `\textpertenthousand`.

```

1095 \UndeclareTextCommand{\textperthousand}{T1}
1096 %\UndeclareTextCommand{\textpertenthousand}{T1}

```

5.1.1 Supporting oldstyle digits

```
1097 \DeclareRobustCommand\oldstylenums[1]{%
1098 \begingroup
1099 \ifmmode
1100 \mathgroup\symletters #1%
1101 \else
1102 \CheckEncodingSubset\@use@text@encoding{TS1}%
1103 {\PackageWarning{textcomp}%
1104 {Oldstyle digits unavailable for
1105 family \f@family.\MessageBreak
1106 Lining digits used instead}}%
1107 \tw@{#1}%
1108 \fi
1109 \endgroup
1110 }
```

5.1.2 Subset encoding defaults

For many font families commonly used in the T_EX world we provide the subset encoding data here. Users can add additional font families in the file `textcomp.cfg` if they own other fonts.

However, if the option “forced” was given then all subset encoding specifications are ignored, so there is no point in setting any of them up:

```
1111 \iftc@forced \else
```

Computer modern based fonts (e.g., CM, CM-Bright, Concrete):

```
1112 \DeclareEncodingSubset{TS1}{cmr} {0}
1113 \DeclareEncodingSubset{TS1}{cmss} {0}
1114 \DeclareEncodingSubset{TS1}{cmtt} {0}
1115 \DeclareEncodingSubset{TS1}{cmvtt} {0}
1116 \DeclareEncodingSubset{TS1}{cmbr} {0}
1117 \DeclareEncodingSubset{TS1}{cmtl} {0}
1118 \DeclareEncodingSubset{TS1}{ccr} {0}
```

PSNFSS fonts:

```
1119 \DeclareEncodingSubset{TS1}{ptm} {4}
1120 \DeclareEncodingSubset{TS1}{pcr} {4}
1121 \DeclareEncodingSubset{TS1}{phv} {4}
1122 \DeclareEncodingSubset{TS1}{ppl} {3}
1123 \DeclareEncodingSubset{TS1}{pag} {4}
1124 \DeclareEncodingSubset{TS1}{pbk} {4}
1125 \DeclareEncodingSubset{TS1}{pnc} {4}
1126 \DeclareEncodingSubset{TS1}{pzc} {4}
1127 \DeclareEncodingSubset{TS1}{bch} {4}
1128 \DeclareEncodingSubset{TS1}{put} {5}
```

Other CTAN fonts (probably not complete):

```
1129 \DeclareEncodingSubset{TS1}{uag} {5}
1130 \DeclareEncodingSubset{TS1}{ugq} {5}
1131 \DeclareEncodingSubset{TS1}{ul8} {4}
1132 \DeclareEncodingSubset{TS1}{ul9} {4} % (LuxiSans, one day)
1133 \DeclareEncodingSubset{TS1}{augie} {5}
1134 \DeclareEncodingSubset{TS1}{dayrom} {3}
1135 \DeclareEncodingSubset{TS1}{dayroms} {3}
1136 \DeclareEncodingSubset{TS1}{pxr} {0}
```

```

1137 \DeclareEncodingSubset{TS1}{pxss}    {0}
1138 \DeclareEncodingSubset{TS1}{pxtt}    {0}
1139 \DeclareEncodingSubset{TS1}{txr}     {0}
1140 \DeclareEncodingSubset{TS1}{txss}    {0}
1141 \DeclareEncodingSubset{TS1}{txtt}    {0}

```

Latin Modern and TeX Gyre:

```

1142 \DeclareEncodingSubset{TS1}{lmr}     {0}
1143 \DeclareEncodingSubset{TS1}{lmdh}    {0}
1144 \DeclareEncodingSubset{TS1}{lmss}    {0}
1145 \DeclareEncodingSubset{TS1}{lmssq}   {0}
1146 \DeclareEncodingSubset{TS1}{lmvtt}   {0}
1147 \DeclareEncodingSubset{TS1}{lmtt}   {0}

1148 \DeclareEncodingSubset{TS1}{qhv}    {0}
1149 \DeclareEncodingSubset{TS1}{qag}    {0}
1150 \DeclareEncodingSubset{TS1}{qbk}    {0}
1151 \DeclareEncodingSubset{TS1}{qcr}    {0}
1152 \DeclareEncodingSubset{TS1}{qcs}    {0}
1153 \DeclareEncodingSubset{TS1}{qpl}    {0}
1154 \DeclareEncodingSubset{TS1}{qtm}    {0}
1155 \DeclareEncodingSubset{TS1}{qzc}    {0}
1156 \DeclareEncodingSubset{TS1}{qhvc}   {0}

```

Fourier-GUTenberg:

```

1157 \DeclareEncodingSubset{TS1}{futs}    {4}
1158 \DeclareEncodingSubset{TS1}{futex}   {4}
1159 \DeclareEncodingSubset{TS1}{futj}    {4}

```

Y&Y's Lucida Bright

```

1160 \DeclareEncodingSubset{TS1}{hlh}     {3}
1161 \DeclareEncodingSubset{TS1}{hls}     {3}
1162 \DeclareEncodingSubset{TS1}{hlst}    {3}

```

The remaining settings for Lucida are conservative: the following fonts contain the `\textohm` character but not the `\texteuro`, i.e., belong to neither subset 4 nor subset 3. If you want to use the `\textohm` with these fonts copy these definition to `textcomp.cfg` and change the subset to 3. However in that case make sure that you do not use the `\texteuro`.

```

1163 \DeclareEncodingSubset{TS1}{hlct}    {5}
1164 \DeclareEncodingSubset{TS1}{hlx}     {5}
1165 \DeclareEncodingSubset{TS1}{hlce}    {5}
1166 \DeclareEncodingSubset{TS1}{hlcn}    {5}
1167 \DeclareEncodingSubset{TS1}{hlcw}    {5}
1168 \DeclareEncodingSubset{TS1}{hlcf}    {5}

```

Other commercial families...

```

1169 \DeclareEncodingSubset{TS1}{pplx}    {3}
1170 \DeclareEncodingSubset{TS1}{pplj}    {3}
1171 \DeclareEncodingSubset{TS1}{ptmx}    {4}
1172 \DeclareEncodingSubset{TS1}{ptmj}    {4}

```

If the file `textcomp.cfg` exists it will be loaded at this point. This allows to define further subset encodings for font families not covered by default.

```

1173 \InputIfFileExists{textcomp.cfg}
1174 {\PackageInfo{textcomp}{Local configuration file used}}{}

```

```

1175 \fi
1176 </TS1oldsty>

```

6 The checkencodingsubset.tex file

This is a simple file that asks for a name of a font family and then displays information about the TS1 encoding for this family and recommends the right encoding subset (to be used with `\DeclareEncodingsubset`) for this family.

```

1177 <*TS1check>
1178 \ProvidesFile{checkencodingsubset.tex}
1179 [2024/10/18 v0.5b Figure out safe TS1 encoding subsets]
1180 \let\typeoutdetails\typeout
1181 %\def\typeoutdetails#1{} % alternative definition used below

```

For the purpose of this check a glyph exists if the font slot is occupied—too bad if that contains the wrong glyph or some tofu. If it “exists” we return 0 otherwise 1. This way we can call this macro several times in a row and obtain a number that is 0 if all glyphs are existing or greater than 0 if any of them is missing.

The second argument (holding the command name for a symbol) is not used during these tests.

```

1182 \def\doesglyphexist#1#2{\iffontchar\testFont #1 0\else 1\relax \fi}

```

This macro also tests and outputs some information about the symbol if it is missing. This time we make use of the second argument.

```

1183 \def\glyphmissingdetails#1#2{\iffontchar\testFont #1 \else
1184   \typeoutdetails{\space\space\space ==> \string#2 (#1) is missing}\fi}
1185 \newif\ifsafesubencodingfound
1186 \newif\ifcoremisses

```

Testing a group of symbols that belong to one sub-encoding. More precisely, the symbols that become unavailable if you change from sub-encoding x (#2) to $x + 1$ (#3). As far as the code is concerned, the symbols that are supposed to be always available (the core) become available if we test the group -1 and 0 .

The first argument contains the testing code and is supposed to return a single number greater or equal to zero.

```

1187 \def\testgroup#1#2#3{%
1188   \ifnum 0 = #1%
1189     \ifnum #2<0
1190       \typeoutdetails{All glyphs in core exist}%
1191     \else
1192       \typeoutdetails{All glyphs between sub-encoding #2 and #3 exist}%
1193     \fi
1194   \else
1195     \ifnum #2<0
1196       \typeoutdetails{*****}%
1197       \typeoutdetails{Some glyphs are missing from core:}%
1198       \coremissestrue
1199       \ifsafesubencodingfound \else
1200         \def\subencodingresult{#2}%
1201       \fi
1202     \else

```

```

1203     \typeoutdetails{Some glyphs are missing from sub-encoding #2:}%
1204     \ifsafesubencodingfound \else
1205         \def\subencodingresult{#3}%
1206     \fi
1207 \fi

```

If some glyphs are missing, we rerun the test code but this time using `\glyphmissingdetails`.

```

1208     {\let\doesglyphexist \glyphmissingdetails #1}%

```

And because we had misses we have definitely found the subset.

```

1209     \safesubencodingfoundtrue
1210 \fi
1211 }

```

The currently defined subset for the family is either stored in `\TS1:<family>` if it was declared, or it is the default subset which is stored in `\TS1:?`.

```

1212 \def\currsubencoding#1{\csname TS1:\ifcsname TS1:#1\endcsname #1\else ?\fi\endcsname}

```

If a font family is not found when declaring it with `\DeclareFixedFont` we end up with the following font. This can then be used as a simple test if we failed loading the TS1 font.

```

1213 \DeclareFixedFont\cmrFont{TS1}{cmr}{m}{n}{10pt}

```

Check for all glyphs in all encoding subsets ...

```

1214 \def\testallgroups#1{%
1215     \DeclareFixedFont\testFont{TS1}{#1}{m}{n}{10pt}%
1216     \ifx\testFont\cmrFont
1217         \typeout{***** Font family #1 not found ****}%
1218     \else

```

We haven't checked anything yet.

```

1219     \safesubencodingfoundfalse
1220     \coremissesfalse
1221     \typeoutdetails{^^J-----}%
1222     \typeoutdetails{Testing font family #1^^J(currently TS1-sub-encoding
1223         \currsubencoding{#1})}%
1224     \typeout{-----}%

```

Then we start testing the groups beginning with the glyphs between sub-encoding 8 and 9. If any of them is missing (checked with `\doesglyphexist`) then we already know that 9 is the correct answer.

```

1225     \testgroup{%
1226         \doesglyphexist{21}{\texttwelveudash}%
1227         \doesglyphexist{22}{\textthreequartersemdash}%
1228         \doesglyphexist{134}{\textbardbl}%
1229         \doesglyphexist{137}{\textcelsius}%
1230         \doesglyphexist{178}{\texttwosuperior}%
1231         \doesglyphexist{179}{\textthreesuperior}%
1232         \doesglyphexist{185}{\textonesuperior}%
1233     }{8}{9}%

```

Nevertheless we go on with further groups so that the output lists all missing glyphs.

```

1234     \testgroup{%
1235         \doesglyphexist{32}{\textblank}%
1236         \doesglyphexist{148}{\textinterrobang}%
1237         \doesglyphexist{149}{\textinterrobangdown}%
1238         \doesglyphexist{191}{\texteuro}%

```

```

1239 }{7}{8}%
1240 \testgroup{%
1241   \doesglyphexist{47}{\textfractionsolidus}%
1242   \doesglyphexist{61}{\textminus}%
1243   \doesglyphexist{87}{\textohm}%
1244   \doesglyphexist{181}{\textmu}%
1245 }{6}{7}%
1246 \testgroup{%
1247   \doesglyphexist{140}{\textflorin}%
1248   \doesglyphexist{164}{\textcurrency}%
1249 }{5}{6}%
1250 \testgroup{%
1251   \doesglyphexist{155}{\textnumero}%
1252   \doesglyphexist{157}{\textestimated}%
1253 }{4}{5}%
1254 \testgroup{%
1255   \doesglyphexist{24}{\textleftarrow}%
1256   \doesglyphexist{25}{\textrightarrow}%
1257   \doesglyphexist{94}{\textuparrow}%
1258   \doesglyphexist{95}{\textdownarrow}%
1259   \doesglyphexist{141}{\textcolonmonetary}%
1260   \doesglyphexist{142}{\textwon}%
1261   \doesglyphexist{146}{\textlira}%
1262   \doesglyphexist{150}{\textdong}%
1263 }{3}{4}%
1264 \testgroup{%
1265   \doesglyphexist{60}{\textlangle}%
1266   \doesglyphexist{62}{\textrangle}%
1267 }{2}{3}%
1268 \testgroup{%
1269   \doesglyphexist{0}{\capitalgrave}%
1270   \doesglyphexist{1}{\capitalacute}%
1271   \doesglyphexist{2}{\capitalcircumflex}%
1272   \doesglyphexist{3}{\capitaltilde}%
1273   \doesglyphexist{4}{\capitaldieresis}%
1274   \doesglyphexist{5}{\capitalhungarumlaut}%
1275   \doesglyphexist{6}{\capitalring}%
1276   \doesglyphexist{7}{\capitalcaron}%
1277   \doesglyphexist{8}{\capitalbreve}%
1278   \doesglyphexist{9}{\capitalmacron}%
1279   \doesglyphexist{10}{\capitaldotaccent}%
1280   \doesglyphexist{11}{\capitalcedilla}%
1281   \doesglyphexist{12}{\capitalogonek}%
1282   \doesglyphexist{26}{\t}%
1283   \doesglyphexist{27}{\capitaltie}%
1284   \doesglyphexist{28}{\newtie}%
1285   \doesglyphexist{29}{\capitalnewtie}%
1286   \doesglyphexist{45}{\textdblhyphen}%
1287   \doesglyphexist{48}{\textzerooldstyle}%
1288   \doesglyphexist{49}{\textoneoldstyle}%
1289   \doesglyphexist{50}{\texttwooldstyle}%
1290   \doesglyphexist{51}{\textthreeoldstyle}%
1291   \doesglyphexist{52}{\textfouroldstyle}%
1292   \doesglyphexist{53}{\textfiveoldstyle}%

```

```

1293 \doesglyphexist{54}{\textsixoldstyle}%
1294 \doesglyphexist{55}{\textsevenoldstyle}%
1295 \doesglyphexist{56}{\texteightoldstyle}%
1296 \doesglyphexist{57}{\textnineoldstyle}%
1297 \doesglyphexist{77}{\textmho}%
1298 \doesglyphexist{79}{\textbigcircle}%
1299 \doesglyphexist{91}{\textlbrackdbl}%
1300 \doesglyphexist{93}{\textrbrackdbl}%
1301 \doesglyphexist{96}{\textasciigrave}%
1302 \doesglyphexist{98}{\textborn}%
1303 \doesglyphexist{99}{\textdivorced}%
1304 \doesglyphexist{100}{\textdied}%
1305 \doesglyphexist{108}{\textleaf}%
1306 \doesglyphexist{109}{\textmarried}%
1307 \doesglyphexist{110}{\textmusicalnote}%
1308 \doesglyphexist{126}{\texttildelow}%
1309 \doesglyphexist{127}{\textdblhyphenchar}%
1310 \doesglyphexist{128}{\textasciibreve}%
1311 \doesglyphexist{129}{\textasciicaron}%
1312 \doesglyphexist{175}{\textasciimacron}%
1313 \doesglyphexist{130}{\textacutedbl}%
1314 \doesglyphexist{131}{\textgravedbl}%
1315 \doesglyphexist{138}{\textdollaroldstyle}%
1316 \doesglyphexist{139}{textcentoldstyle}%
1317 \doesglyphexist{143}{\textnaira}%
1318 \doesglyphexist{144}{\textguarani}%
1319 \doesglyphexist{145}{\textpeso}%
1320 \doesglyphexist{147}{\textrecipe}%
1321 \doesglyphexist{152}{\textpertenthousand}%
1322 \doesglyphexist{153}{\textpilcrow}%
1323 \doesglyphexist{154}{\textbaht}%
1324 \doesglyphexist{156}{\textdiscount}%
1325 \doesglyphexist{158}{\textopenbullet}%
1326 \doesglyphexist{159}{\textservicemark}%
1327 \doesglyphexist{160}{\textlquill}%
1328 \doesglyphexist{161}{\textrquill}%
1329 \doesglyphexist{168}{\textasciidieresis}%
1330 \doesglyphexist{171}{\textcopyleft}%
1331 \doesglyphexist{173}{\textcircledP}%
1332 \doesglyphexist{180}{\textasciiacute}%
1333 \doesglyphexist{184}{\textreferencemark}%
1334 \doesglyphexist{187}{\textsurd}%
1335 }{1}{2}%

```

All fonts (up to now) that belong to sub-encoding 1 do have the `\textcircled` glyph, but it is too small to be usable. So this test for this group currently doesn't do much good—but who knows maybe one day a font shows up in which this glyph is actually missing.

```

1336 \testgroup{%
1337 \doesglyphexist{79}{\textcircled}% this is not a proper test because the symbol is
1338 % usually available but not usable
1339 }{0}{1}%
1340 \testgroup{%
1341 \doesglyphexist{13}{\textquotestraightbase}%

```



```

1342     \doesglyphexist{18}{\textquotestraightdblbase}%
1343     \doesglyphexist{23}{\textcapitalcompwordmark}%
1344     \doesglyphexist{31}{\textascendercompwordmark}%
1345     \doesglyphexist{36}{\textdollar}%
1346     \doesglyphexist{39}{\textquotesingle}%
1347     \doesglyphexist{42}{\textasteriskcentered}%
1348     \doesglyphexist{132}{\textdagger}%
1349     \doesglyphexist{133}{\textdaggerdbl}%
1350     \doesglyphexist{135}{\textperthousand}%
1351     \doesglyphexist{136}{\textbullet}%
1352     \doesglyphexist{151}{\texttrademark}%
1353     \doesglyphexist{162}{\textcent}%
1354     \doesglyphexist{163}{\textsterling}%
1355     \doesglyphexist{165}{\textyen}%
1356     \doesglyphexist{166}{\textbrokenbar}%
1357     \doesglyphexist{167}{\textsection}%
1358     \doesglyphexist{169}{\textcopyright}%
1359     \doesglyphexist{170}{\textordfeminine}%
1360     \doesglyphexist{172}{\textlnot}%
1361     \doesglyphexist{174}{\textregistered}%
1362     \doesglyphexist{176}{\textdegree}%
1363     \doesglyphexist{177}{\textpm}%
1364     \doesglyphexist{182}{\textparagraph}%
1365     \doesglyphexist{183}{\textperiodcentered}%
1366     \doesglyphexist{186}{\textordmasculine}%
1367     \doesglyphexist{188}{\textonequarter}%
1368     \doesglyphexist{189}{\textonehalf}%
1369     \doesglyphexist{190}{\textthreequarters}%
1370     \doesglyphexist{214}{\texttimes}%
1371     \doesglyphexist{246}{\textdiv}%
1372 }{-1}{0}%

```

If all groups have all glyphs then we have the full encoding (subset 0).

```

1373     \ifsafesubencodingfound\else
1374         \def\subencodingresult{0}%
1375     \fi

```

If the font is missing some of the core glyphs we make a remark about this, because they will never display.

```

1376     \typeoutdetails{-----}%
1377     \typeout{TS1 encoding subset for #1\ifcoremisses \space(ignoring core misses)\fi
1378     \space (\ifnum\subencodingresult =
1379     \currsubencoding{#1} ok\else bad\fi)}%
1380     \typeout{Use sub-encoding \subencodingresult
1381     \ifnum\subencodingresult = \currsubencoding{#1}\else
1382     \space (not \currsubencoding{#1})\fi}
1383     \typeout{-----^^J}%
1384     \fi
1385 }

```

This tests all declarations (or most of them) that have been added to the kernel. It is called if no family is given interactively.

```

1386 \long\def\testallkerneldefinedfamilies{%
1387 \testallgroups{ccr}%     {0}
1388 \testallgroups{cubr}%    {0}

```

```

1389 %%\testallgroups{cmr}% {0} % don't test this one as it is the fallback
1390 % thus reports that the family is not found
1391 \testallgroups{cmss}% {0}
1392 \testallgroups{cmtl}% {0}
1393 \testallgroups{cmtt}% {0}
1394 \testallgroups{cmvtt}% {0}
1395 \testallgroups{pxr}% {0}
1396 \testallgroups{pxss}% {0}
1397 \testallgroups{pxtt}% {0}
1398 \testallgroups{qag}% {0}
1399 \testallgroups{qbk}% {0}
1400 \testallgroups{qcr}% {0}
1401 \testallgroups{qcs}% {0}
1402 \testallgroups{qhvc}% {0}
1403 \testallgroups{qhv}% {0}
1404 \testallgroups{qpl}% {0}
1405 \testallgroups{qtm}% {0}
1406 \testallgroups{qzc}% {0}
1407 \testallgroups{txr}% {0}
1408 \testallgroups{txss}% {0}
1409 \testallgroups{txtt}% {0}
1410 %
1411 % Next would claim to be 0 (or 2)
1412 %
1413 %%\testallgroups{lmr}% {1}
1414 %%\testallgroups{lmdh}% {1}
1415 %%\testallgroups{lmss}% {1}
1416 %%\testallgroups{lmssq}% {1}
1417 %%\testallgroups{lmvtt}% {1}
1418 %%\testallgroups{lmtt}% {1} % missing TM, SM and pertenthousand so really 2
1419 %
1420 % these are no longer in TeX Live
1421 %
1422 %%\testallgroups{ptmx}% {2} % gone for a long time it seems
1423 %%\testallgroups{ptmj}% {2} % ditto
1424 %%\testallgroups{ul8}% {2} % ditto
1425 %
1426 % next block has tofu chars so results are wrong
1427 %
1428 %%\testallgroups{bch}% {5} % tofu for blank, ohm
1429 %%\testallgroups{futj}% {5} % tofu for blank, interrobang/down, ohm
1430 %%\testallgroups{futs}% {5} % tofu for blank, ohm
1431 %%\testallgroups{futx}% {5} % probably (currently broken distrib)
1432 %%\testallgroups{pag}% {5} % tofu for blank, interrobang/down, ohm
1433 %%\testallgroups{pbk}% {5} % tofu for blank, interrobang/down, ohm
1434 %%\testallgroups{pcr}% {5} % tofu for blank, interrobang/down, ohm
1435 %%\testallgroups{phv}% {5} % tofu for blank, interrobang/down, ohm
1436 %%\testallgroups{pnc}% {5} % tofu for blank, interrobang/down, ohm
1437 %%\testallgroups{pplj}% {5} % tofu for blank
1438 %%\testallgroups{pplx}% {5} % tofu for blank
1439 %%\testallgroups{ppl}% {5} % tofu for blank interrobang/down
1440 %%\testallgroups{ptm}% {5} % tofu for blank, interrobang/down, ohm
1441 %%\testallgroups{pzc}% {5} % tofu for blank, interrobang/down, ohm
1442 %%\testallgroups{ul9}% {5} % tofu for blank, interrobang/down, ohm

```

```

1443 %\testallgroups{dayroms}%{6} % tofu for blank, interrobang/down, ohm
1444 %\testallgroups{dayrom}% {6} % tofu for blank, interrobang/down, ohm
1445 %\testallgroups{augie}%{8} % really only missing euro and full of tofu
1446 %\testallgroups{put}% {8}
1447 %\testallgroups{uag}% {8} % probably (currently broken distrib)
1448 %\testallgroups{ugq}% {8}
1449 %
1450 \testallgroups{zi4}% {9}
1451 %
1452 %% not installed normally
1453 %
1454 %\testallgroups{hls}% {5}
1455 %\testallgroups{hlst}% {5}
1456 %\testallgroups{hlct}% {5}
1457 %\testallgroups{hlh}% {5}
1458 %\testallgroups{hlx}% {8}
1459 %\testallgroups{hlce}% {8}
1460 %\testallgroups{hlcn}% {8}
1461 %\testallgroups{hlcw}% {8}
1462 %\testallgroups{hlcf}% {8}
1463
1464 \testallgroups{lato-LF}% {0} % with a bunch of tofu inside --- should probably be changed
1465 \testallgroups{opensans-TLF}%{0} % with a bunch of tofu inside --- should probably be change
1466 \testallgroups{cantarell-TLF}% {0} % with a bunch of tofu inside --- should probably be cha
1467 \testallgroups{fbb-LF}% {0} % missing centoldstyle ---> 2
1468 \testallgroups{tli}% {1} % with lots of tofu inside --- should probably be changed
1469 \testallgroups{Alegreya-OsF}% {2}
1470 \testallgroups{AlegreyaSans-OsF}% {2}
1471 \testallgroups{DejaVuSans-TLF}% {2}
1472 \testallgroups{DejaVuSansCondensed-TLF}% {2}
1473 \testallgroups{DejaVuSansMono-TLF}% {2} this is missing \textfractionsolidus which makes it 7
1474 \testallgroups{EBGaramond-LF}% {2}
1475 \testallgroups{Tempora-TLF}% {2}
1476 \testallgroups{Tempora-TOsF}% {2}
1477 \testallgroups{Arimo-TLF}% {3}
1478 \testallgroups{Crlt-TLF}% {3} changed from Carlito-
1479 \testallgroups{FiraSans-LF}% {3} should be 4
1480 \testallgroups{IBMPlexSans-TLF}% {3}
1481 \testallgroups{Merriwthr-OsF}% {3} changed from Merriweather- and should be 2
1482 \testallgroups{Montserrat-LF}% {3} now 2
1483 \testallgroups{MontserratAlternates-LF}%{3} now 2
1484 \testallgroups{SourceCodePro-TLF}% {3}
1485 \testallgroups{SourceCodePro-TOsF}% {3}
1486 \testallgroups{SourceSansPro-OsF}% {3}
1487 \testallgroups{SourceSerifPro-LF}% {3}
1488 \testallgroups{Tinos-TLF}% {3}
1489 \testallgroups{AccanthisADFStdNoThree-LF}%{4}
1490 \testallgroups{Cabin-TLF}% {4}
1491 \testallgroups{Caladea-TLF}% {4}
1492 \testallgroups{Chivo-LF}% {4}
1493 \testallgroups{ClearSans-TLF}% {4}
1494 \testallgroups{Coelacanth-LF}% {4}
1495 \testallgroups{CrimsonPro-LF}% {4}
1496 \testallgroups{FiraMono-TLF}% {4}

```

```

1497 \testallgroups{FiraMono-T0sF}%      {4}
1498 \testallgroups{Go-TLF}%             {4}
1499 \testallgroups{GoMono-TLF}%         {4}
1500 \testallgroups{InriaSans-LF}%       {4}
1501 \testallgroups{InriaSerif-LF}%      {4}
1502 \testallgroups{LibertinusSans-LF}%  {4}
1503 \testallgroups{LibertinusSerif-LF}% {4}
1504 \testallgroups{LibreBodoni-TLF}%    {4}
1505 \testallgroups{LibreFranklin-TLF}%  {4}
1506 \testallgroups{LinguisticsPro-LF}%  {4}
1507 \testallgroups{LinguisticsPro-0sF}% {4}
1508 \testallgroups{LinuxBiolinumT-LF}%  {4}
1509 \testallgroups{LinuxLibertineT-LF}% {4}
1510 \testallgroups{MerriwthrSans-0sF}%  {4}      name change and now 2
1511 \testallgroups{MintSpirit-LF}%      {4}
1512 \testallgroups{MintSpiritNoTwo-LF}% {4}
1513 \testallgroups{PTMono-TLF}%         {4}
1514 \testallgroups{PTSans-TLF}%         {4}
1515 \testallgroups{PTSansCaption-TLF}%   {4}
1516 \testallgroups{PTSansNarrow-TLF}%   {4}
1517 \testallgroups{PTSerif-TLF}%        {4}
1518 \testallgroups{PTSerifCaption-TLF}%  {4}
1519 \testallgroups{Raleway-TLF}%         {4}
1520 \testallgroups{Raleway-T0sF}%        {4}
1521 \testallgroups{Roboto-LF}%          {4}
1522 \testallgroups{RobotoMono-TLF}%     {4}
1523 \testallgroups{RobotoSlab-TLF}%     {4}
1524 \testallgroups{Rosario-LF}%         {4}
1525 \testallgroups{SticksTooText-LF}%   {4}
1526 \testallgroups{UniversalisADFStd-LF}%{4}
1527 \testallgroups{Almndr-0sF}%         {5}      name change
1528 \testallgroups{Baskervaldx-LF}%     {5}
1529 \testallgroups{BaskervilleF-LF}%    {5}      now 2
1530 \testallgroups{Bttr-TLF}%           {5}      name changed from Bitter-...
1531 \testallgroups{Cinzel-LF}%          {5}
1532 \testallgroups{CinzelDecorative-LF}%{5}
1533 \testallgroups{DejaVuSerif-TLF}%    {5}
1534 \testallgroups{DejaVuSerifCondensed-TLF}% {5}
1535 \testallgroups{GilliusADF-LF}%      {5}
1536 \testallgroups{charssil-TLF} %%      missing should be 5
1537 \testallgroups{GilliusADFCond-LF}%   {5}
1538 \testallgroups{GilliusADFNoTwo-LF}%  {5}
1539 \testallgroups{GilliusADFNoTwoCond-LF}%{5}
1540 \testallgroups{Lbstr-LF}%           {5}      name change and should be 7
1541 \testallgroups{OldStandard-TLF}%     {5}
1542 \testallgroups{PlyfrDisplay-LF}%     {5}      name change
1543 \testallgroups{PlyfrDisplay-0sF}%    {5}      name change
1544 \testallgroups{TheanoDidot-TLF}%     {5}
1545 \testallgroups{TheanoDidot-T0sF}%    {5}
1546 \testallgroups{TheanoModern-TLF}%     {5}
1547 \testallgroups{TheanoModern-T0sF}%    {5}
1548 \testallgroups{TheanoOldStyle-TLF}%  {5}
1549 \testallgroups{TheanoOldStyle-T0sF}%  {5}
1550 \testallgroups{Crimson-TLF}%        {6}

```

```

1551 \testallgroups{IBMPlexMono-TLF}% {6} now 3
1552 \testallgroups{IBMPlexSerif-TLF}% {6} now 3
1553 \testallgroups{LibertinusMono-TLF}%{6} should be 8
1554 \testallgroups{LibertinusSerifDisplay-LF}%{6}
1555 \testallgroups{LinuxLibertineDisplayT-LF}%{6}
1556 \testallgroups{LinuxLibertineMonoT-LF}%{6}
1557 \testallgroups{LinuxLibertineMonoT-TLF}%{6}
1558 \testallgroups{Ovrlck-LF}% {6} name changed
1559 \testallgroups{CormorantGaramond-LF}%{7}
1560 \testallgroups{Heuristica-TLF}% {7}
1561 \testallgroups{Heuristica-T0sF}% {7}
1562 \testallgroups{IMFELLEnglish-TLF}%{7}
1563 \testallgroups{LibreBskvl-LF}% {7} %% wrong name LibreBaskerville-TLF
1564 \testallgroups{LibreCsln-LF}% {7} changed from LibreCaslon-
1565 \testallgroups{Mrcls-LF}% {7} %% wrong name Marcellus-LF
1566 \testallgroups{NotoSans-LF}% {7}
1567 \testallgroups{NotoSansMono-TLF}%{7} now 2
1568 \testallgroups{NotoSansMono-T0sF}%{7} now 2
1569 \testallgroups{NotoSerif-LF}% {7}
1570 \testallgroups{Quattro-LF}% {7} changed from Quattrocento-
1571 \testallgroups{QuattroSans-LF}% {7} changed from QuattrocentoSans-
1572 \testallgroups{XCharter-TLF}% {7} now 2
1573 \testallgroups{XCharter-T0sF}% {7} now 2
1574 \testallgroups{erewhon-LF}% {7} now 2
1575 \testallgroups{ComicNeue-TLF}% {7}
1576 \testallgroups{ComicNeueAngular-TLF}%{7}
1577 \testallgroups{Frm-LF}%{7} % the superiors are missing; name changed from Forum-LF
1578 \testallgroups{Cochineal-TLF}% {8} now 5
1579 \testallgroups{AlgolRevived-TLF}%{9}
1580 }

```

There interaction with the user.

```

1581 \typeout{^^J=====}
1582 \typeout{| Enter font family to check (or <enter> for kernel defined families)}
1583 \typeout{=====}
1584 \typein[\FontFamilyToCheck]{}

1585 \if!\FontFamilyToCheck!
1586   \typeout{=====}
1587   \typeout{| Detailed output? (default no)}
1588   \typeout{=====}
1589   \typein[\Details]{}
1590   \if!\Details!
1591     \def\typeoutdetails#1{}
1592   \else
1593     \let\typeoutdetails\typeout
1594   \fi
1595   \testallkerneldefinedfamilies
1596 \else
1597   \let\typeoutdetails\typeout
1598   \testallgroups\FontFamilyToCheck
1599 \fi

1600 \stop
1601 </TS1check>

```

File 34

ltpageno.dtx

1 Page Numbering

Page numbers are produced by a page counter, used just like any other counter. The only difference is that `\c@page` contains the number of the next page to be output (the one currently being produced), rather than one minus it. Thus, it is normally initialized to 1 rather than 0. `\c@page` is defined to be `\count0`, rather than a count assigned by `\newcount`.

`\pagenumbering` The user sets the page number style with the `\pagenumbering{<foo>}` command, which sets the page counter to 1 and defines `\thepage` to be `\foo`. For example, `\pagenumbering{roman}` causes pages to be numbered i, ii, etc.

```
1 <*2kernel>
2 \message{page nos.,}
3 \countdef\c@page=0 \c@page=1
4 \def\c1@page{}
5 \def\pagenumbering#1{%
6   \global\c@page \c@page \gdef\thepage{\csname @#1\endcsname
7     \c@page}}
8 </2kernel>
```

File 35

ltxref.dtx

1 Cross Referencing

The user writes `\label{foo}` to define the following cross-references:

`\ref*{foo}`: value of most recently incremented referenceable counter. in the current environment. (Chapter, section, theorem, footnote and enumeration counters and other counters stepped with `\refstepcounter` are referenceable.)

`\pageref*{foo}`: page number at which `\label{foo}` command appeared. where `foo` can be any string of characters not containing `\`, `{` or `}`.

Note: The scope of the `\label` command is delimited by environments, so `\begin{theorem} \label{foo} ... \end{theorem} \label{bar}` defines `\ref{foo}` to be the theorem number and `\ref{bar}` to be the current section number.

Note: `\label` does the right thing in terms of spacing – i.e., leaving a space on both sides of it is equivalent to leaving a space on either side.

Note: the starred versions `\ref*` and `\pageref*` are provided to align with the use of `hyperref`. Without `hyperref` (or some other package using the starred form) the star is simply ignored.

Note: starting with 2023-06-01 `\label` stores also the current value of `\@currentlabelname` which should typically contain a (sanitized) title. (A reference command `\nameref` is provided by the `nameref` package.) `\label` also stores `\@currentHref` which if set should refer to a target name for links. This value is set and used by `hyperref`. Unlike the other values `\@currentHref` should be set globally. A fifth value `\@kernel@reserved@label@data` is reserved for the kernel to allow future extensions of the cross-reference system.

1.1 Cross Referencing

Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

```
1 <*2kernel>
2 \message{x-ref,}
```

This is implemented as follows. A referenceable counter `CNT` is incremented by the command `\refstepcounter{CNT}`, which sets `\@currentlabel == {CNT}{eval(\p@cnt\theCNT)}`. The command `\label{FOO}` then writes the following on file `\@auxout`:

```
\newlabel{FOO}{eval(\@currentlabel)}{eval(\thepage)}%
```

```
{eval(\@currentlabelname)}{eval(\@currentHref)}{eval(\@kernel@reserved@label@data)}}}
```

```
\ref{FOO} ==
BEGIN
  if \r@foo undefined
  then @refundefined := G T
  ??
  Warning: 'reference foo on page ... undefined'
```

```

        else \@car \eval(\r@F00)\@nil
      fi
    END

\pageref{foo} =
  BEGIN
    if \r@foo undefined
      then @refundefined := G T
         ??
         Warning: 'reference foo on page ... undefined'
      else \@cdr \eval(\r@F00)\@nil
    fi
  END

```

End of historical L^AT_EX 2.09 comments.

`\labelformat` A reference via `\ref` produces by default the data associated with the corresponding `\label` command (typically a number); any additional formatting has to be provided by the user. If, for example, references to equations are always to be typeset as “equation (*number*)”, one has to code “`equation (\ref{key})`”. With `\labelformat` there is a possibility to generate such frills automatically without resorting to low-level coding. The command takes two arguments: the first is the name of a counter and the second is its representation when referenced. This means that for a successful usage, one has to know the counter name being used for generating the label, though in practice this should not pose a problem. The current counter number is picked up as an argument. Here are two examples:

```

\labelformat{section}{section-#1}
\labelformat{equation}{equation-(#1)}

```

`\Ref` A side effect of using `\labelformat` is that, depending on the defined formatting, it becomes impossible to use `\ref` at the beginning of a sentence (if its replacement text starts with a lowercase letter). To overcome this problem we introduce the command `\Ref` that behave like `\ref` except that it uppercases the first token of the generated string.

To make `\Ref` work properly the very first token in the second argument of `\labelformat` has to be a simple ASCII or UTF-8 letter, otherwise the capitalization will fail or worse, you will end up with some error messages. If you actually need something more complicated in this place (e.g., an accented letter not written as a UTF-8 character) you have to explicitly surround it with braces, to identify the part that needs to be capitalized. For example, for figure references in the Hungarian language you might want to write `\labelformat{figure}{\a'bra~\thefigure}` or use `\labelformat{figure}{ábra~\thefigure}` which avoids the brace problem.

`\G@refundefinedtrue` This does not save on name-space (since `\G@refundefinedfalse` was never needed) but `\@refundined` it does make the implementation of such one-way switches more consistent. The extra macro to make the change is used since this change appears several times.

Note despite its name, `\G@refundinedtrue` does *not* correspond to an `\if` command, and there is no matching `...false`. It would be more natural to call the command `\G@refundined` (as inspection of the change log will reveal) but unfortunately such a change would break any package that had defined a `\ref`-like command that mimicked the definition of `\ref`, calling `\G@refundinedtrue`. Inspection of the T_EX archives

revealed several such packages, and so this command has been named ...true so that the definition of \ref need not be changed, and the packages will work without change.

```

3 % \newif\ifG@refundefined
4 % \def\G@refundefinedtrue{\global\let\ifG@refundefined\iftrue}
5 % \def\G@refundefinedfalse{\global\let\ifG@refundefined\iffalse}
6 \def\G@refundefinedtrue{%
7   \gdef\@refundefined{%
8     \@latex@warning@no@line{There were undefined references}}
9 \let\@refundefined\relax

```

(End of definition for \G@refundefinedtrue and \@refundefined.)

\ref Referencing a \label. RmS 91/10/25: added a few extra \reset@font, as suggested by
\pageref Bernd Raichle
\@setref RmS 92/08/14: made \ref and \pageref robust
RmS 93/09/08: Added setting of refundefined switch.

```

10 </2ekernel>
11 <*2ekernel | latexrelease>
12 <latexrelease>\IncludeInRelease{2023/06/01}%
13 <latexrelease>          {\@kernel@sref}{store five arguments}%
14 \def\@setref#1#2#3{%
15   \ifx#1\relax
16     \protect\G@refundefinedtrue
17     \nfss@text{\reset@font\bfseries ??}%
18     \@latex@warning{Reference ‘#3’ on page \thepage \space
19                   undefined}%
20   \else
21     \expandafter#2#1\@empty\@empty\@empty\null
22   \fi}
23 \long\def\@firstoffive#1#2#3#4#5{#1}
24 \long\def\@secondoffive#1#2#3#4#5{#2}
25 \def\@kernel@sref#1{\expandafter\@setref\csname r@#1\endcsname\@firstoffive{#1}}
26 \def\@kernel@spageref#1{\expandafter\@setref\csname r@#1\endcsname
27   \@secondoffive{#1}}
28 <latexrelease>\EndIncludeInRelease
29 <latexrelease>\IncludeInRelease{2022/06/01}%
30 <latexrelease>          {\@kernel@sref}{store five arguments}%
31 <latexrelease>\def\@setref#1#2#3{%
32 <latexrelease>   \ifx#1\relax
33 <latexrelease>     \protect\G@refundefinedtrue
34 <latexrelease>     \nfss@text{\reset@font\bfseries ??}%
35 <latexrelease>     \@latex@warning{Reference ‘#3’ on page \thepage \space
36                   undefined}%
37 <latexrelease>   \else
38 <latexrelease>     \expandafter#2#1\null
39 <latexrelease>   \fi}
40 <latexrelease>\let\@firstoffive\undefined
41 <latexrelease>\let\@secondoffive\undefined
42 <latexrelease>\def\@kernel@sref#1{\expandafter\@setref\csname r@#1\endcsname\@firstoftwo{#1}}
43 <latexrelease>\def\@kernel@spageref#1{\expandafter\@setref\csname r@#1\endcsname
44   \@secondoftwo{#1}}
45 <latexrelease>\EndIncludeInRelease
46 <latexrelease>\IncludeInRelease{0000/00/00}%

```

```

47 <latexrelease>                {\@kernel@sref}{store five arguments}%
48 <latexrelease>\def\@setref#1#2#3{%
49 <latexrelease>  \ifx#1\relax
50 <latexrelease>    \protect\G@refundefinedtrue
51 <latexrelease>    \nfss@text{\reset@font\bfseries ??}%
52 <latexrelease>    \@latex@warning{Reference ‘#3’ on page \thepage \space
53 <latexrelease>        undefined}%
54 <latexrelease>  \else
55 <latexrelease>    \expandafter#2#1\null
56 <latexrelease>  \fi}
57 <latexrelease>\let\@firstoffive\undefined
58 <latexrelease>\let\@secondoffive\undefined
59 <latexrelease>\let\@kernel@sref\undefined
60 <latexrelease>\let\@kernel@spageref\undefined
61 <latexrelease>\EndIncludeInRelease
62 <latexrelease>\IncludeInRelease{2022/06/01}%
63 <latexrelease>                {\ref}{Add starred reference commands}%
64 \let\@kernel@ref\@kernel@sref
65 \let\@kernel@pageref\@kernel@spageref
66 \NewDocumentCommand\ref{s}
67   {\IfBooleanTF{#1}{\@kernel@sref}{\@kernel@ref}}
68 \NewDocumentCommand\pageref{s}
69   {\IfBooleanTF{#1}{\@kernel@spageref}{\@kernel@pageref}}

```

As the commands are now protected we also need expandable versions for use in `\ifthenelse`:

```

70 \def\@kernel@pageref@exp#1{\csname cs_if_exist:cTF\endcsname
71   {r@#1}{\csname t1_item:cn\endcsname{r@#1}{2}}{0}}
72 \def\@kernel@ref@exp#1{\csname cs_if_exist:cTF\endcsname
73   {r@#1}{\csname t1_item:cn\endcsname{r@#1}{1}}{0}}
74 </2kernel | latexrelease>
75 <latexrelease>\EndIncludeInRelease
76 <latexrelease>\IncludeInRelease{0000/00/00}%
77 <latexrelease>                {\ref}{Add starred reference commands}%
78 <latexrelease>\def\ref#1{\expandafter\@setref\csname r@#1\endcsname\@firstoftwo{#1}}
79 <latexrelease>\def\pageref#1{\expandafter\@setref\csname r@#1\endcsname
80 <latexrelease>                \@secondoftwo{#1}}
81 <latexrelease>
82 <latexrelease>\EndIncludeInRelease
83 <*2kernel>

```

(End of definition for \ref, \pageref, and \@setref.)

`\newlabel` This command will be written to the `.aux` file to pass label information from one run to another.

`\@newl@bel` The internal form of `\newlabel` and `\bibcite`. Note that this macro does its work inside a group. That way the local assignments it needs to do don't clutter the save stack. This prevents large documents with many labels to run out of save stack.

```

84 \def\@newl@bel#1#2#3{%
85   \ifundefined{#1@#2}%
86     \relax
87     {\gdef \@multiplelabels {%
88       \@latex@warning@no@line{There were multiply-defined labels}}}%

```

```

89     \@latex@warning@no@line{Label ‘#2’ multiply defined}}%
90     \global\@namedef{#1@#2}{#3}}
91     \def\newlabel{\@newl@bel r}
92     \@onlypreamble\@newl@bel

```

(End of definition for \newlabel and \@newl@bel.)

`\if@multiplelabels` This is redefined to produce a warning if at least one label is defined more than once. It is executed by the `\enddocument` command.

`\@multiplelabels`

```

93     \let \@multiplelabels \relax

```

(End of definition for \if@multiplelabels and \@multiplelabels.)

\label The commands `\label` and `\refstepcounter` have been changed to allow `\protect`'ed commands to work properly. For example,

```

\def\thechapter{\protect\foo{\arabic{chapter}.\roman{section}}}

```

will cause a `\label{bar}` command to define `\ref{bar}` to expand to something like `\foo{4.d}`. Change made 20 Jul 88.

```

94     </2ekernel>
95     <*2ekernel | latexrelease>
96     <latexrelease>\IncludeInRelease{2023/06/01}%
97     <latexrelease>           {\label}{store five label arguments}%
98     \providecommand\@currentlabelname{}
99     \providecommand\@currentHref{}
100    \providecommand\@kernel@reserved@label@data{}
101    \NewHookWithArguments{label}{1}
102    \def\label#1{\@bsphack
103      \begingroup
104      \UseHookWithArguments{label}{1}{#1}%
105      \protected@write\@auxout{}%
106        {\string\newlabel{#1}{\@currentlabel}{\thepage}%
107         {\@currentlabelname}{\@currentHref}{\@kernel@reserved@label@data}}}%
108      \endgroup
109      \@esphack}
110    <latexrelease>\EndIncludeInRelease

```

(End of definition for \label. This function is documented on page 809.)

`\refstepcounter` Saved * for testing the argument of `\refstepcounter`.

```

111    \def\ltx@star@counter{*}

```

(End of definition for \refstepcounter.)

`\refstepcounter` Step the counter and allow for labels to point to its current value.

```

112    <latexrelease>\IncludeInRelease{2022/06/01}%
113    <latexrelease>           {\Ref}{Add starred version}%
114    \def\@currentcounter{}
115    <latexrelease>\EndIncludeInRelease
116    <latexrelease>\IncludeInRelease{2024/11/01}%
117    <latexrelease>           {\@currentHref}{set theHcounter representation}%

```

`refstepcounter` (*socket*) This socket takes the whole code as argument. The default kernel plug is identity. By changing the plug `hyperref` can add a conditional and e.g. suppress the processing in a PDF context.

```
118 \NewSocket{refstepcounter}{1}
```

`refstepcounter/target` (*socket*) This socket takes an argument, the counter name, and should at least set from it the target name `\@currentHref`. With `hyperref` it sets also the actual target. This is done with a socket so that the target name is not set more than once to (possibly) different names. The socket is not used in `\@kernel@refstepcounter`. The tagging code needs the target name so it is added after this socket.

```
119 \NewSocket{refstepcounter/target}{1}
```

`(refstepcounter/target)` (*plug*)

```
120 \NewSocketPlug{refstepcounter/target}{kernel}
121 {\xdef\@currentHref {#1.\csname theH#1\endcsname}}%
122 \AssignSocketPlug{refstepcounter/target}{kernel}

123 \def\refstepcounter#1{%
124   \UseSocket{refstepcounter}{%
125     \stepcounter{#1}%
126     \edef\reserved@a{#1}%
127     \ifx\reserved@a\ltx@star@counter\else
128       \let\@currentcounter\reserved@a
129     \fi
130     \protected@edef\@currentlabel
```

By generating the second `csname` first the `\p@...` command can grab it as an argument which can be helpful for more complicated typesetting arrangements.

The trick is to ensure that `\csname the#1\endcsname` is turned into a single token before `\p@...` is expanded further. This way, if the `\p@...` command is a macro with one argument it will receive `\the...` With the original kernel code (i.e., without the `\expandafter`) it will instead pick up `\csname` which would be disastrous.

Using `\expandafter` instead of braces delimiting the argument is better because, assuming that the `\p@...` command is not defined as a macro with one argument, the braces will stay and prohibit kerning that might otherwise happen between the glyphs generated by `\the...` and surrounding glyphs.

```
131     {\csname p@#1\expandafter\endcsname\csname the#1\endcsname}}%
132     \UseSocket{refstepcounter/target}{#1}%
133     \UseTaggingSocket{recordtarget}%
134   }%
135 }
```

This is a version of `\refstepcounter` which does not set and use targets.

```
\@kernel@refstepcounter 136 \def\@kernel@refstepcounter#1{%
137   \UseSocket{refstepcounter}{%
138     \stepcounter{#1}%
139     \edef\reserved@a{#1}%
140     \ifx\reserved@a\ltx@star@counter\else
141       \let\@currentcounter\reserved@a
142     \fi
143     \protected@edef\@currentlabel
144     {\csname p@#1\expandafter\endcsname\csname the#1\endcsname}}}%

```

```

145 <latexrelease>\EndIncludeInRelease
146 <latexrelease>\IncludeInRelease{2022/06/01}%
147 <latexrelease>          {\@currentHref}{set theHcounter representation}%
148 <latexrelease>\def\refstepcounter#1{\stepcounter{#1}%
149 <latexrelease>  \edef\@currentcounter{#1}%
150 <latexrelease>  \protected@edef\@currentlabel
151 <latexrelease>    {\csname p@#1\expandafter\endcsname\csname the#1\endcsname}%
152 <latexrelease>}
153 <latexrelease>\let\@kernel@refstepcounter\refstepcounter
154 <latexrelease>\EndIncludeInRelease

```

(End of definition for \refstepcounter and \@kernel@refstepcounter.)

```

155 <latexrelease>\IncludeInRelease{2022/06/01}%
156 <latexrelease>          {\Ref}{Add starred version}%

```

\labelformat A shortcut to set the \p@... macro for a counter. It will pick up the counter representation as an argument so that it can be specially formatted.

```

157 \def\labelformat#1{\expandafter\def\csname p@#1\endcsname##1}

```

(End of definition for \labelformat.)

\Ref This macro expands the result of \ref and then uppercases the first token. Only useful if the label was generated via \labelformat and contains some lower case letter at its start. If the label starts with a complicated construct (e.g., an accented letter that is provided via a command, e.g., \a instead of a UTF-8 character like ä) one has to surround everything that needs uppercasing in a brace group in the definition of \labelformat.⁴⁰

```

158 % \changes{v1.1s}{2024/12/10}{Replace \cs{@tempa} with \cs{reserved@a} (gh/1579)}
159 \def\@kernel@Ref#1{\protected@edef\reserved@a{\@kernel@ref{#1}}%
160   \expandafter\MakeUppercase\reserved@a}
161 \def\@kernel@sRef#1{\protected@edef\reserved@a{\@kernel@sref{#1}}%
162   \expandafter\MakeUppercase\reserved@a}
163 \NewDocumentCommand\Ref{s}
164   {\IfBooleanTF{#1}{\@kernel@sRef}{\@kernel@Ref}}

```

(End of definition for \Ref.)

```

165 </2kernel | latexrelease>
166 <latexrelease>\EndIncludeInRelease
167 <latexrelease>\IncludeInRelease{0000/00/00}%
168 <latexrelease>          {\label}{store five label arguments}%
169 <latexrelease>\let\@currenttitle\@undefined
170 <latexrelease>\let\@currenttarget\@undefined
171 <latexrelease>\let\@kernel@currentdata\@undefined
172 <latexrelease>\def\label#1{\@bsphack
173 <latexrelease>  \protected@write\@auxout{%
174 <latexrelease>    {\string\newlabel{#1}{\@currentlabel}{\thepage}}}%
175 <latexrelease>  \@esphack}
176 <latexrelease>\EndIncludeInRelease
177 <latexrelease>\IncludeInRelease{2020/10/01}%
178 <latexrelease>          {\Ref}{Add starred version}%
179 <latexrelease>\def\@currentcounter{}

```

⁴⁰There is one problem with this approach: the braces are kept in a normal \ref which might spoil kerning. Perhaps one day this needs redoing.

```

180 <latexrelease>\def\refstepcounter#1{\stepcounter{#1}%
181 <latexrelease> \edef\@currentcounter{#1}%
182 <latexrelease> \protected@edef\@currentlabel
183 <latexrelease> {\csname p@#1\expandafter\endcsname\csname the#1\endcsname}%
184 <latexrelease>}
185 <latexrelease>\def\labelformat#1{\expandafter\def\csname p@#1\endcsname##1}
186 <latexrelease>\DeclareRobustCommand\Ref[1]{\protected@edef\reserved@a{\ref{#1}}%
187 <latexrelease> \expandafter\MakeUppercase\reserved@a}
188 <latexrelease>\EndIncludeInRelease
189 <latexrelease>\IncludeInRelease{2019/10/01}%
190 <latexrelease> {\refstepcounter}{Add \labelformat and \Ref}%
191 <latexrelease>\let\@currentcounter\@undefined
192 <latexrelease>\def\refstepcounter#1{\stepcounter{#1}%
193 <latexrelease> \protected@edef\@currentlabel
194 <latexrelease> {\csname p@#1\expandafter\endcsname\csname the#1\endcsname}%
195 <latexrelease>}
196 <latexrelease>\def\labelformat#1{\expandafter\def\csname p@#1\endcsname##1}
197 <latexrelease>\DeclareRobustCommand\Ref[1]{\protected@edef\reserved@a{\ref{#1}}%
198 <latexrelease> \expandafter\MakeUppercase\reserved@a}
199 <latexrelease>\EndIncludeInRelease
200 <latexrelease>\IncludeInRelease{0000/00/00}%
201 <latexrelease> {\refstepcounter}{Add \labelformat and \Ref}%
202 <latexrelease>
203 <latexrelease>\def\refstepcounter#1{\stepcounter{#1}%
204 <latexrelease> \protected@edef\@currentlabel
205 <latexrelease> {\csname p@#1\endcsname\csname the#1\endcsname}%
206 <latexrelease>}
207 <latexrelease>\let\labelformat\@undefined
208 <latexrelease>\let\Ref\@undefined
209 <latexrelease>
210 <latexrelease>\EndIncludeInRelease
211 <*2ekernel>

```

`\@currentlabel` Default for `\label` commands that come before any environment.

```
212 \def\@currentlabel{}
```

(End of definition for \@currentlabel.)

```
213 </2ekernel>
```

File 36

ltproperties.dtx

Abstract

This code implements command to record and (expandably) reference document properties. It extends the standard `\label/\ref/\pageref` commands.

1 Introduction

The module allows to record the “current state” of various document properties (typically the content of macros and values of counters) and to access them in other places through a label. The list of properties that can be recorded and retrieved are not fix and can be extended by the user. The values of the properties are recorded in the `.aux` file and can be retrieved at the second compilation.

The module uses the ideas of properties and labels. A label is a document reference point: a name for the user. An property is something that \LaTeX can track, such as a page number, section number or name. The names of labels and properties may be arbitrary. Note that there is a single namespace for each.

2 Design discussion

The design here largely follows ideas from `zref`. In particular, there are two independent concepts: properties that can be recorded between runs, and labels which consist of lists of these properties. The reason for the split is that individual labels will want to record some but not all properties. For examples, a label concerned with position would track the x and y coordinates of the current point, but not for example the page number.

In the current implementation, properties share a single namespace. This allows multiple lists to re-use the same properties, for example page number, absolute page number, etc. This does mean that *changing* a standard property is an issue. However, some properties have complex definitions (again, see `zref` at present): having them in a single shared space avoids the need to copy code.

Labels could be implemented as `prop` data. That is not done at present as there is no obvious need to map to or copy the data. As such, faster performance is available using a hash table approach as in a “classical” set up. Data written to the `.aux` file uses simple paired *balanced text* not `keyvals`: this avoids any restrictions on names and again offers increased performance.

The `expl3` versions of the label command do not use `\@bsphack/\@esphack` to avoid double spaces, but the $\text{\LaTeX} 2_{\epsilon}$ command does as it lives at the document command level.

The reference commands are expandable.

Currently the code has nearly no impact on the main `\label` and `\ref` commands as too many external packages rely on the concrete implementation. There is one exception: the label names share the same namespace. That means that if both `\label{ABC}` and `\RecordProperties{ABC}{page}` are used there is a warning `Label ‘ABC’ multiply defined`.

3 Handling unknown labels and properties

With the standard `\label/\ref` commands the requested label is either in the `.aux`-file (and so known) or not. In the first case the stored value can be used, in the second case the reference commands print two question marks.

With flexible property lists a reference commands asks for the value of a specific property stored under a label name and we have to consider more variants:

- If the requested property is unknown (not declared) the system is not correctly set up and an error is issued.
- If the label is unknown, the default of the property is used.
- If the label is known, but doesn't provide a value for the property then again the default of the property is used.
- The command `\property_ref:nnn` allows to give a local default which is used instead of the property default in the two cases before.

4 Rerun messages

As the reference commands are expandable they can neither issue a message that the label or the label-property combination is unknown, nor can they trigger the rerun message at the end of the \LaTeX run.

Where needed such messages must therefore be triggered manually. For this two commands are provided: `\property_ref_undefined_warn:` and `\property_ref_undefined_warn:nn`. See below for a description.

5 Open points

- The `xpos` and `ypos` properties require that the position is stored first but there is no (public) engine independent interface yet. Code must use `\tex_savepos:D`.

6 Code interfaces

<code>\property_new:nnnn</code>	<code>\property_new:nnnn</code>	<code>{\property}</code>	<code>{\setpoint}</code>	<code>{\default}</code>	<code>{\code}</code>
<code>\property_gset:nnnn</code>	<code>\property_gset:nnnn</code>	<code>{\property}</code>	<code>{\setpoint}</code>	<code>{\default}</code>	<code>{\code}</code>

\LaTeX 2 ϵ -interface: see `\NewProperty`, `\SetProperty`.

Sets the `\property` to have the `\default` specified, and at the `\setpoint` (either `now` or `shipout`) to write the result of the `\code` as part of a label. The `\code` should be expandable. The expansion of `\code` (the value of the property) is written to the `.aux` file and read back from there at the next compilation. Values should assume that the standard \LaTeX catcode régime with `@` a letter is active then.

If the property is declared within a package it is suggested that its name is build from letters, hyphens and slashes, and is always structured as follows: `\package-name/\property-name`.

`\property_record:nN` `\property_record:nN {<label>} <clist var>`
`\property_record:nn` `\property_record:nn {<label>} {<clist>}`
`\property_record:(nV|ee)` L^AT_EX 2_ε-interface: see `\RecordProperties`.
Writes the list of properties given by the `<clist>` to the `.aux` file with the `<label>` specified.

`\property_ref:nn *` `\property_ref:nn {<label>} {<property>}`
`\property_ref:ee *` L^AT_EX 2_ε-interface: see `\RefProperty`.
Expands to the value of the `<property>` for the `<label>`, if available, and the default value of the property otherwise. If `<property>` has not been declared with `\property_new:nnnn` an error is issued. The command raises an internal, expandable, local flag if the reference can not be resolved.

`\property_ref:nnn *` `\property_ref:nnn {<label>} {<property>} {<local default>}`
`\property_ref:een *` L^AT_EX 2_ε-interface: see `\RefProperty`.
Expands to the value of the `<property>` for the `<label>`, if available, and to `<local default>` otherwise. If `<property>` has not been declared with `\property_new:nnnn` an error is issued. The command raises an internal, expandable local flag if the reference can not be resolved.

`\property_ref_undefined_warn:` `\property_ref_undefined_warn:`
L^AT_EX 2_ε-interface: not provided.
Triggers the standard warning
LaTeX Warning: There were undefined references.
at the end of the document if there was a recent `\property_ref:nn` or `\property_ref:nnn` which couldn't be resolved and so raised the flag. "Recent" means in the same group or in some outer group!

`\property_ref_undefined_warn:n` `\property_ref_undefined_warn:n {<label>}`
`\property_ref_undefined_warn:e`
L^AT_EX 2_ε-interface: not provided.
Triggers the standard warning
LaTeX Warning: There were undefined references.
at the end of the document if `<label>` is not known. At the point where it is called it also issues the warning
Reference '`<label>`' on page `<page>` undefined.

`\property_ref_undefined_warn:nn` `\property_ref_undefined_warn:nn {<label>} {<property>}`
`\property_ref_undefined_warn:ee`
L^AT_EX 2_ε-interface: see `\RefUndefinedWarn`.
Triggers the standard warning
LaTeX Warning: There were undefined references.
at the end of the document if the reference can not be resolved. At the point where it is called it also issues the warning
Reference '`<label>`' on page `<page>` undefined
if the label is unknown, or the more specific
Property '`<property>`' undefined for reference '`<label>`' on page `<page>`
if the label is known but doesn't provide a value for the requested property.

```

\property_if_exist_p:n * \property_if_exist_p:n {<property>}
\property_if_exist_p:e * \property_if_exist:nTF {<property>} {<true code>} {<false code>}
\property_if_exist:nTF * LATEX 2ε-interface: \IfPropertyExistsTF.
\property_if_exist:eTF * Tests if the <property> has been declared.

```

```

\property_if_recorded_p:n * \property_if_recorded_p:n {<label>}
\property_if_recorded_p:e * \property_if_recorded:nTF {<label>} {<true code>} {<false code>}
\property_if_recorded:nTF *
\property_if_recorded:eTF *

```

L^AT_EX 2_ε-interface: \IfLabelExistsTF
Tests if the <label> is known. This is also true if the label has been set with the standard \label command.

```

\property_if_recorded_p:nn * \property_if_recorded_p:nn {<label>} {<property>}
\property_if_recorded_p:ee * \property_if_recorded:nnTF {<label>} {<property>} {<true code>} {<false code>}
\property_if_recorded:nnTF *
\property_if_recorded:eeTF *

```

L^AT_EX 2_ε-interface: \IfPropertyRecordedTF.
Tests if the label <label> is known and if it provides a value of the <property>.

7 Auxiliary file interfaces

```

\new@label@record \new@label@record {<label>} {<data>}

```

This is a command only for use in the .aux file. It loads the key–value list of <data> to be available for the <label>.

8 L^AT_EX 2_ε interface

The LaTeX interfaces always expand label and property arguments. This means that one must be careful when using active chars or commands in the names. UTF8-chars are protected and should be safe, similar most babel shorthands.

```

\NewProperty \NewProperty {<property>} {<setpoint>} {<default>} {<code>}
\SetProperty \SetProperty {<property>} {<setpoint>} {<default>} {<code>}

```

Sets the <property> to have the <default> specified, and at the <setpoint> (either now or shipout) to write the result of the <code> as part of a label. The <code> should be expandable. The expansion of <code> (the value of the property) is written to the .aux file and read back from there at the next compilation (at which point normally the standard L^AT_EX catcode régime with @ a letter is active).

```

\RecordProperties \RecordProperties {<label>} {<clist>}

```

Writes the list of properties given by the <clist> to the .aux file with the <label> specified. Similar to the standard \label command the arguments are expanded. So <clist> can be a macro containing a list of properties. Also similar to the standard \label command, the command is surrounded by an \@bsphack/\@esphack pair to preserve spacing.

<u>\RefProperty</u> *	<u>\RefProperty</u> [<i>local default</i>] { <i>label</i> } { <i>property</i> }	Expands to the value of the <i>property</i> for the <i>label</i> , if available, and the default value of the property or – if given – to <i>local default</i> otherwise. If { <i>property</i> } has not been declared an error is issued.
<u>\IfPropertyExistsTF</u> <u>\IfPropertyExistsT</u> <u>\IfPropertyExistsF</u>	<u>\IfPropertyExistsTF</u> { <i>property</i> } { <i>true code</i> } { <i>false code</i> }	Tests if the <i>property</i> has been declared.
<u>\IfLabelExistsTF</u> <u>\IfLabelExistsT</u> <u>\IfLabelExistsF</u>	<u>\IfLabelExistsTF</u> { <i>label</i> } { <i>true code</i> } { <i>false code</i> }	Tests if the <i>label</i> has been recorded. This is also true if a label has been set with the standard <code>\label</code> command.
<u>\IfPropertyRecordedTF</u> <u>\IfPropertyRecordedT</u> <u>\IfPropertyRecordedF</u>	<u>\IfPropertyRecordedTF</u> { <i>label</i> } { <i>property</i> } { <i>true code</i> } { <i>false code</i> }	Tests if the label and a value of the <i>property</i> for the <i>label</i> are both known.
<u>\RefUndefinedWarn</u>	<u>\RefUndefinedWarn</u> { <i>label</i> } { <i>property</i> }	Triggers the standard warning <p style="margin-left: 2em;">LaTeX Warning: There were undefined references.</p> at the end of the document if the reference for <i>label</i> and <i>property</i> can not be resolved. At the point where it is called it also issues the warning <p style="margin-left: 2em;">Reference ‘<i>label</i>’ on page <i>page</i> undefined</p> if the label is unknown, or the more specific <p style="margin-left: 2em;">Property ‘<i>property</i>’ undefined for reference ‘<i>label</i>’ on page <i>page</i> if</p> the label is known but doesn’t provide a value for the requested property.

9 Pre-declared properties

<u>abspage</u>	(shipout) The absolute value of the current page: starts at 1 and increases monotonically at each shipout.
<u>page</u>	(shipout) The current page as given by <code>\thepage</code> : this may or may not be a numerical value, depending on the current style. Contrast with <code>\abspage</code> . You get this value also with the standard <code>\label/\pageref</code> .
<u>pagenum</u>	(shipout) The current page as arabic number. This is suitable for integer operations and comparisons.
<u>label</u>	(now) The content of <code>\@currentlabel</code> . This is the value that you get also with the standard <code>\label/\ref</code> .

title (now) The content of `\@currentlabelname`. This command is filled beside others by the `nameref` package and some classes (e.g. `memoir`).

target (now) The content of `\@currentHref`. This command is normally filled by for example `hyperref` and gives the name of the last destination it created.

pagetarget (shipout) The content of `\@currentHpage`. This command is filled for example by a recent version of `hyperref` and then gives the name of the last page destination it created.

counter (now) The content of `\@currentcounter`. This command contains after a `\refstepcounter` the name of the counter.

xpos (shipout) This stores the x and y coordinates of a point previously stored with `\pdfsavepos`/`\savepos`. E.g. (if `bidi` is used it can be necessary to save the position before and after the label):

```
\tex_savepos:D
\property_record:nn{myposition}{xpos,ypos}
\tex_savepos:D
```

10 The Implementation

```
1 <*2ekernel | latexrelease>
2 \ExplSyntaxOn
3 <@@=property>
4 <latexrelease>\NewModuleRelease{2023/11/01}{ltproperties}
5 <latexrelease>          {Cross-referencing~properties}
```

The approach here is based closely on that from `zref`; separate out lists of properties and the properties themselves, so the latter can be used multiple times and in varying combinations. However, not everything is a straight copy. Firstly, we treat lists of properties as simple comma lists: that allows us to have either saved or dynamic lists and to avoid another data structure. The cost is that errors are detected at point-of-use, but in any real case that should be true anyway (and is true for `\zref@labelbyprop` already). Secondly, we allow properties to have arbitrary names, as the code does not require them to tokenize as control sequences.

`\property_new:nnnn` As properties can be reset, they are not constants. But they also have various pieces
`\property_gset:nnnn` of required data. So we use the same approach as `color` and make them declarations.
`__property_gset:nnnn` Data-wise, we need the detail of the implementation, the default and a flag to show if
the code works now or at shipout. This last entry is done using `text` so needs a check.
We could use a set of `prop` here, but as we never need to map or copy the lists, we can
gain performance using the hash table approach.

```
6 \cs_new_protected:Npn \property_new:nnnn #1#2#3#4
```

```

7   {
8     \cs_if_free:cTF { __property_code_ #1 : }
9     {
10      \exp_args:Nx \__property_gset:nnnn { \tl_to_str:n {#1} }
11        {#2} {#3} {#4}
12    }
13    {
14      \msg_error:nn { property }{ exists }{#1}
15    }
16  }
17 \cs_new_protected:Npn \property_gset:nnnn #1#2#3#4
18 {
19   \__property_gset:ennn { \tl_to_str:n {#1} }
20     {#2} {#3} {#4}
21 }
22 \cs_new_protected:Npn \__property_gset:nnnn #1#2#3#4
23 {
24   \cs_gset:cpn { __property_code_ #1 : } {#4}
25   \tl_gclear_new:c { g__property_default_ #1 _tl }
26   \tl_gset:cn { g__property_default_ #1 _tl } {#3}
27   \bool_if_exist:cF { g__property_shipout_ #1 _tl }
28     { \bool_new:c { g__property_shipout_ #1 _tl } }
29   \str_case:nnF {#2}
30     {
31       { now } { { \bool_gset_false:c { g__property_shipout_ #1 _tl } } } }
32     { shipout }
33     { \bool_gset_true:c { g__property_shipout_ #1 _tl } }
34   }
35   { \msg_error:nnnn { property } { unknown-setpoint } {#1} {#2} }
36 }
37 \cs_generate_variant:Nn \__property_gset:nnnn {ennn}

```

(End of definition for `\property_new:nnnn`, `\property_gset:nnnn`, and `__property_gset:nnnn`.
These functions are documented on page 806.)

`\NewProperty` For consistency we expand the property name, but this doesn't warrant a variant of the
`\SetProperty` L3-commands.

```

38 \cs_new_protected:Npn \NewProperty #1#2#3#4
39 {
40   \protected@edef\reserved@a{#1}
41   \exp_args:No \property_new:nnnn {\reserved@a} {#2}{#3}{#4}
42 }
43 \cs_new_protected:Npn \SetProperty #1#2#3#4
44 {
45   \protected@edef\reserved@a{#1}
46   \exp_args:No \property_gset:nnnn {\reserved@a} {#2}{#3}{#4}
47 }

```

(End of definition for `\NewProperty` and `\SetProperty`. These functions are documented on page 808.)

`\property_record:nN` Writing data when it is labelled means expanding at this stage and possibly later too.
`\property_record:nn` That is all pretty easy using `expl3`: we accept a stray comma at the end of the list as
`\property_record:nV` that is easier to deal with than trying to tidy up, and there is no real downside.

```

48 \cs_new_protected:Npn \property_record:nN #1#2

```

```

\__property_record:nn
\__property_record:en
\__property_record_value:n
  \_property_record_value_aux:n
  \_property_record_value_aux:e

```

```

49 { \property_record:nV {#1} #2 }
50 \cs_new_protected:Npn \property_record:nn #1#2
51 { \__property_record:en { \tl_to_str:n {#1} } {#2} }
52 \cs_generate_variant:Nn \property_record:nn { nV , ee, oo }
53 \cs_new_protected:Npn \__property_record:nn #1#2
54 {
55   \protected@write \@auxout {}
56   {
57     \token_to_str:N \new@label@record
58     {#1}
59     { \clist_map_function:nN {#2} \__property_record_value:n }
60   }
61 }
62 \cs_generate_variant:Nn \__property_record:nn { e }
63 \cs_new:Npn \__property_record_value:n #1
64 { \__property_record_value_aux:e { \tl_to_str:n {#1} } }
65 \cs_new:Npn \__property_record_value_aux:n #1
66 {
67   \cs_if_exist:cTF { __property_code_ #1 : }
68   {
69     {#1}
70     {
71       \bool_if:cTF { g__property_shipout_ #1 _tl }
72       { \exp_not:c }
73       { \use:c }
74       { __property_code_ #1 : }
75     }
76   }
77   { \msg_expandable_error:nnn { property } { not-declared } {#1} }
78 }
79 \cs_generate_variant:Nn \__property_record_value_aux:n { e }

```

(End of definition for `\property_record:nN` and others. These functions are documented on page 807.)

`\RecordProperties`

```

80 \NewDocumentCommand\RecordProperties { m m }
81 {
82   \@bsphack
83   \protected@edef\reserved@a{#1}
84   \protected@edef\reserved@b{#2}
85   \property_record:oo {\reserved@a}{\reserved@b}
86   \@esphack
87 }

```

(End of definition for `\RecordProperties`. This function is documented on page 808.)

10.1 Reference commands

`l__property_ref_flag` A flag that is set if a reference couldn't be resolved.

```

88 \flag_new:n { l__property_ref_flag }

```

(End of definition for `l__property_ref_flag`.)

`\property_ref:nn` Search for the label/property combination, and if not found fall back to the default of the property.

`\property_ref:ee`

```

89 \cs_new:Npn \property_ref:nn #1#2
90 {
91   \__property_ref:een
92   { \tl_to_str:n {#1} }
93   { \tl_to_str:n {#2} }
94   { \tl_use:c { g__property_default_ #2 _tl } }
95 }
96 \cs_generate_variant:Nn \property_ref:nn {ee}

```

(End of definition for `\property_ref:nn`. This function is documented on page 807.)

`\property_ref:nnn` This allows to set a local default value which overrides the default value of the property.

`\property_ref:een`

`__property_ref:nnn`

`__property_ref:een`

```

97 \cs_new:Npn \property_ref:nnn #1#2#3
98 {
99   \__property_ref:een
100   { \tl_to_str:n {#1} }
101   { \tl_to_str:n {#2} }
102   {#3}
103 }
104 \cs_new:Npn \__property_ref:nnn #1#2#3
105 {
106   \tl_if_exist:cTF { g__property_label_ #1 _ #2 _tl }
107   { \tl_use:c { g__property_label_ #1 _ #2 _tl } }
108   {
109     \flag_if_raised:nF
110     { l__property_ref_flag } { \flag_raise:n { l__property_ref_flag } }

```

We test for the default of the property only to check if the property has been declared.

```

111   \tl_if_exist:cTF { g__property_default_ #2 _tl }
112   { #3 }
113   { \msg_expandable_error:nnn { property } { not-declared } {#2} }
114 }
115 }
116 \cs_generate_variant:Nn \__property_ref:nnn { ee }
117 \cs_generate_variant:Nn \property_ref:nnn { een}

```

(End of definition for `\property_ref:nnn` and `__property_ref:nnn`. This function is documented on page 807.)

`\RefProperty` Search for the label/property combination, and if not found fall back to the default of the property or the given default.

```

118 \NewExpandableDocumentCommand \RefProperty { o m m }
119 {
120   \IfNoValueTF {#1}
121   {
122     \property_ref:ee {#2}{#3}
123   }
124   {
125     \property_ref:een {#2}{#3}{#1}
126   }
127 }

```

(End of definition for `\RefProperty`. This function is documented on page 809.)

```

\new@label@record A standard recursion loop.
\__property_data:nnn 128 \cs_new_protected:Npn \new@label@record #1#2
129 {
130   \tl_if_exist:cTF { r@#1 }
131   {
132     \gdef \@multiplelabels
133     { \@latex@warning@no@line { There-were-multiply-defined~labels } }
134     \@latex@warning@no@line { Label-‘#1’~multiply-defined }
135   }
136   {
137     \tl_new:c { r@#1 }
138     \tl_gset:cn { r@#1 }{#2}
139   }
140   \__property_data:nnn {#1} #2 { \q_recursion_tail } { ? } \q_recursion_stop
141 }
142 \cs_new_protected:Npn \__property_data:nnn #1#2#3
143 {
144   \quark_if_recursion_tail_stop:n {#2}
145   \tl_gclear_new:c { g__property_label_ \tl_to_str:n {#1} _ \tl_to_str:n {#2} _tl }
146   \tl_gset:cn { g__property_label_ \tl_to_str:n {#1} _ \tl_to_str:n {#2} _tl } {#3}
147   \__property_data:nnn {#1}
148 }

```

This command is used in `\enddocument` to test if some label values have changed.

```

149 \cs_new_protected:Npn \@kernel@new@label@record@testdef #1 #2
150 {
151   \tl_if_eq:cnF { r@#1 } {#2}
152   { \@tempwattrue }
153 }

```

(End of definition for `\new@label@record` and `__property_data:nnn`. This function is documented on page 808.)

10.2 Tests and warnings

`\property_if_exist_p:n` Tests if property has been declared.

```

\property_if_exist:nTF 154 \prg_new_conditional:Npnn \property_if_exist:n #1 { p , T , F , TF }
155 % #1 property
156 {
157   \cs_if_exist:cTF { __property_code_ #1 : }
158   {
159     \prg_return_true:
160   }
161   {
162     \prg_return_false:
163   }
164 }
165 \prg_generate_conditional_variant:Nnn \property_if_exist:n {e} { p , T , F , TF }

```

(End of definition for `\property_if_exist:nTF`. This function is documented on page 808.)

`\IfPropertyExistsTF`

`\IfPropertyExistsT`

`\IfPropertyExistsF`

```

166 \cs_new_eq:NN \IfPropertyExistsTF \property_if_exist:eTF
167 \cs_new:Npn \IfPropertyExistsT #1#2 {\property_if_exist:eTF {#1}{#2}{}}
168 \cs_new:Npn \IfPropertyExistsF #1 {\property_if_exist:eTF {#1}{}}

```


(End of definition for `\IfPropertyExistsTF`, `\IfPropertyExistsT`, and `\IfPropertyExistsF`. These functions are documented on page 809.)

`\property_if_recorded_p:n` Tests if the label has been set. This can then be used to setup e.g. rerun messages.

```

\property_if_recorded:nTF
169 \prg_new_conditional:Npnn \property_if_recorded:n #1 { p , T , F, TF }
170   % #1 label
171   {
172     \tl_if_exist:cTF { r@#1 }
173     {
174       \prg_return_true:
175     }
176     {
177       \prg_return_false:
178     }
179   }
180 \prg_generate_conditional_variant:Nnn \property_if_recorded:n {e} { p , T , F, TF }

```

(End of definition for `\property_if_recorded:nTF`. This function is documented on page 808.)

`\IfLabelExistsTF`
`\IfLabelExistsT`
`\IfLabelExistsF`

```

181 \cs_new_eq:NN \IfLabelExistsTF \property_if_recorded:eTF
182 \cs_new:Npn \IfLabelExistsT #1#2 { \property_if_recorded:eTF {#1}{#2}{ } }
183 \cs_new:Npn \IfLabelExistsF #1 { \property_if_recorded:eTF {#1}{ } }

```

(End of definition for `\IfLabelExistsTF`, `\IfLabelExistsT`, and `\IfLabelExistsF`. These functions are documented on page 809.)

`\property_if_recorded_p:nn` tests if the label/property combination has been set This can then be used to setup e.g. rerun messages.

```

\property_if_recorded:nnTF
184 \prg_new_conditional:Npnn \property_if_recorded:nn #1#2 { p , T , F, TF }
185   % #1 label #2 property
186   {
187     \tl_if_exist:cTF { g__property_label_ \tl_to_str:n {#1} _ \tl_to_str:n {#2} _tl }
188     {
189       \prg_return_true:
190     }
191     {
192       \prg_return_false:
193     }
194   }
195 \prg_generate_conditional_variant:Nnn \property_if_recorded:nn {ee} { p , T , F, TF }

```

(End of definition for `\property_if_recorded:nnTF`. This function is documented on page 808.)

`\IfPropertyRecordedTF`
`\IfPropertyRecordedT`
`\IfPropertyRecordedF`

```

196 \cs_new_eq:NN \IfPropertyRecordedTF \property_if_recorded:eeTF
197 \cs_new:Npn \IfPropertyRecordedT #1#2#3 { \property_if_recorded:eeTF {#1}{#2}{#3}{ } }
198 \cs_new:Npn \IfPropertyRecordedF #1#2#3 { \property_if_recorded:eeTF {#1}{#2}{ }{#3} }

```

(End of definition for `\IfPropertyRecordedTF`, `\IfPropertyRecordedT`, and `\IfPropertyRecordedF`. These functions are documented on page 809.)

`\property_ref_undefined_warn:` `\G@refundefinedtrue` is defined in `ltxref` and redefines a warning message.

```
199 \cs_new_protected:Npn \property_ref_undefined_warn:
200 {
201   \flag_if_raised:nT { l__property_ref_flag }
202   {
203     \G@refundefinedtrue
204   }
205 }
```

(End of definition for `\property_ref_undefined_warn:`. This function is documented on page 807.)

`\property_ref_undefined_warn:n`

```
206 \cs_new_protected:Npn \property_ref_undefined_warn:n #1 %#1 label
207 {
208   \property_if_recorded:nF {#1}
209   {
210     \G@refundefinedtrue
211     \@latex@warning{Reference~‘#1’~on~page~\thepage\space undefined}%
212   }
213 }
```

(End of definition for `\property_ref_undefined_warn:n`. This function is documented on page 807.)

`\property_ref_undefined_warn:nn`

`\property_ref_undefined_warn:ee`

`\RefUndefinedWarn`

```
214 \cs_new_protected:Npn \property_ref_undefined_warn:nn #1#2 %#1 label, #2 property
215 {
216   \property_if_recorded:nTF {#1}
217   {
218     \property_if_recorded:nnF {#1}{#2}
219     {
220       \G@refundefinedtrue
221       \@latex@warning
222       { Property~‘#2’~undefined~for~reference~‘#1’~on~page~\thepage }
223     }
224   }
225   {
226     \G@refundefinedtrue
227     \@latex@warning { Reference~‘#1’~on~page~\thepage\space undefined }%
228   }
229 }
```

```
230 \cs_generate_variant:Nn \property_ref_undefined_warn:nn {ee}
```

```
231 \cs_set_eq:NN \RefUndefinedWarn \property_ref_undefined_warn:ee
```

(End of definition for `\property_ref_undefined_warn:nn` and `\RefUndefinedWarn`. These functions are documented on page 807.)

10.3 Predeclared properties

`abspage`

```
232 \property_new:nnnn { abspage } { shipout }
233 { 0 } { \int_use:N \g_shipout_readonly_int }
```

(End of definition for `abspage`. This variable is documented on page 809.)

page

```
234 \property_new:nmmn { page } { shipout } { 0 } { \thepage }
```

(End of definition for page. This variable is documented on page 809.)

pagenum

```
235 \property_new:nmmn { pagenum } { shipout } { 0 } { \the \value { page } }
```

(End of definition for pagenum. This variable is documented on page 809.)

label

```
236 \property_new:nmmn { label } { now } { ?? } { \@currentlabel }
```

(End of definition for label. This variable is documented on page 809.)

title

```
237 \property_new:nmmn { title } { now }
```

```
238 { \exp_not:n { \textbf { ?? } } } { \@currentlabelname }
```

(End of definition for title. This variable is documented on page 810.)

target

```
239 \property_new:nmmn { target } { now } { } { \@currentHref }
```

(End of definition for target. This variable is documented on page 810.)

target

```
240 \newcommand\@currentHpage{}
```

```
241 \property_new:nmmn { pagetarget } { shipout } { } { \@currentHpage }
```

(End of definition for target. This variable is documented on page 810.)

counter

```
242 \property_new:nmmn { counter } { now } { } { \@currentcounter }
```

(End of definition for counter. This variable is documented on page 810.)

xpos

ypos

```
243 \property_new:nmmn { xpos } { shipout } { 0 } { \int_use:N \tex_lastxpos:D }
```

```
244 \property_new:nmmn { ypos } { shipout } { 0 } { \int_use:N \tex_lastypos:D }
```

(End of definition for xpos and ypos. These variables are documented on page 810.)

10.4 Messages

```
245 \msg_new:nnnn { property } { exists }
246   { Property~'#1'~ has~ already~ been~ declared. }
247   { There~ already~ exists~ a~ property~ declaration~ with~ this~
248     name.\\
249     Please~ use~ a~ different~ name~ for~ your~ property.}
250
251 \msg_new:nnnn { property } { not-declared }
252   { Property~'#1'~not-declared. }
253   {
254     LaTeX~has~been~asked~to~use~property~'#1',~but~this~
255     name~has~not~been~declared.
256   }
257 \msg_new:nnnn { property } { unknown-setpoint }
258   { Unknown~keyword~'#2'~for~setting~property~'#1'. }
259   {
260     LaTeX~has~been~asked~to~set~the~property~'#1',~but~the~keyword~
261     '#2'~is~not~one~of~the~two~known~values:~'now'~or~'shipout'.
262   }
263 %
264 <latexrelease>\IncludeInRelease{0000/00/00}{ltproperties}
265 <latexrelease>           {cross-referencing-properties-(undo)}%
266 <latexrelease>
267 <latexrelease>\let \NewProperty \@undefined
268 <latexrelease>\let \SetProperty \@undefined
269 <latexrelease>
270 <latexrelease>\let \RecordProperties \@undefined
271 <latexrelease>\let \RefProperty \@undefined
272 <latexrelease>\let \RefUndefinedWarn \@undefined
273 <latexrelease>
274 <latexrelease>\let \IfPropertyExistsTF \@undefined
275 <latexrelease>\let \IfLabelExistsTF \@undefined
276 <latexrelease>\let \IfPropertyRecordedTF \@undefined
277 <latexrelease>
278 <latexrelease>\let\new@label@record \@undefined
279 <latexrelease>\let\@kernel@new@label@record@testdef\@undefined
280 <latexrelease>\EndModuleRelease
281 \ExplSyntaxOff
282 </2ekernel | latexrelease>
283
284   Reset module prefix:
285 <@@=>
```

File 37

ltmiscen.dtx

1 Miscellaneous Environments

This section implements the basic environment mechanism, and also a few specific environments including `document`, The math environments and related commands, the ‘flushing’ environments, (`center`, `flushleft`, `flushright`), and `verbatim`.

```
1 ⟨*2ekernel)
2 \message{environments,}
```

1.1 Environments

`\begin{foo}` and `\end{foo}` are used to delimit environment `foo`.

`\begin{foo}` starts a group and calls `\foo` if it is defined, otherwise it does nothing.

`\end{foo}` checks to see that it matches the corresponding `\begin` and if so, it calls `\endfoo` and does an `\endgroup`. Otherwise, `\end{foo}` does nothing.

If `\end{foo}` needs to ignore blanks after it, then `\endfoo` should globally set the `@ignore` switch true with `\@ignoretrue` (this will automatically be global).

NOTE: `\@@end` is defined to be the `\end` command of T_EX82.

`\enddocument` is the user’s command for ending the manuscript file.

`\stop` is a panic button — to end T_EX in the middle.

Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

```
\enddocument ==
BEGIN
  \@checkend{document}    %% checks for unmatched \begin
  \clearpage
  \begingroup
  if @filesw = true
  then close file @mainaux
  if G@refundefined = true
  then LaTeX Warning: 'There are undefined references.' fi
  if @multiplelabels = true
  then LaTeX Warning:
    'One or more label(s) multiply defined.'
  else
  \@setckpt {ARG1}{ARG2} == null
  \newlabel{LABEL}{VAL} ==
  BEGIN
    \reserved@a == VAL
    if def(\reserved@a) = def(\r@LABEL)
    else @tempswa := true      fi
  END
  \bible{LABEL}{VAL} == null
  BEGIN
    \reserved@a == VAL
    if def(\reserved@a) = def(\g@LABEL)
    else @tempswa := true      fi
```

```

                                END
                                @tempswa := false
                                make @ a letter
                                \input \jobname.AUX
                                if @tempswa = true
                                    then LaTeX Warning: 'Label may have changed.
                                        Rerun to get cross-references right.'
                                fi
                                fi
                                fi
                                \endgroup
                                finish up
                                END

```

```

\@writefile{EXT}{ENTRY} ==
  if tf@EXT undefined
    else \write\tf@EXT{ENTRY}
  fi

```

End of historical L^AT_EX 2.09 comments.

`\@currentenv` The name of the current environment. Initialized to `document` to so that `\end{document}` works correctly.

```

3 \def\@currentenv{document}

```

(End of definition for \@currentenv.)

```

\if@ignore
\@ignoretrue
\@ignorefalse
4 \def\@ignorefalse{\global\let\if@ignore\iffalse}
5 \def\@ignoretrue {\global\let\if@ignore\iftrue}
6 \@ignorefalse

```

(End of definition for \if@ignore, \@ignoretrue, and \@ignorefalse.)

`\ignorespacesafterend`

```

7 \let\ignorespacesafterend\@ignoretrue

```

(End of definition for \ignorespacesafterend.)

~~`\end{document}`~~
`\enddocument`

```

8 </2ekernel>
9 <*2ekernel | latexrelease>
10 <latexrelease>\IncludeInRelease{2023/11/01}%
11 <latexrelease>          {\enddocument}{check property labels}%
12 \def\enddocument{%

```

The `\end{document}` hook is executed first. If necessary it can contain a `\clearpage` to output dangling floats first. In this position it can also contain something like `\end{foo}` so that the whole document effectively starts and ends with some special environment. However, this must be used with care, eg if two applications would use this without knowledge of each other the order of the environments will be wrong after all. `\AtEndDocument` is redefined at this point so that and such commands that get into the hook do not chase their tail...

```

13 \@kernel@before@enddocument
14 \UseOneTimeHook{enddocument}%
15 \@kernel@after@enddocument

```

```

16 \@checkend{document}%
17 \clearpage
18 \UseOneTimeHook{enddocument/afterlastpage}%
19 \@kernel@after@enddocument@afterlastpage
20 \beginingroup
21   \if@filesw
22     \immediate\closeout\@mainaux
23     \let\@setckpt\@gobbletwo
24     \let\@newlabel\@testdef
25
26     \let\newlabel@record\@kernel@newlabel@record@testdef

```

The previous line is equiv to setting

```

\def\newlabel{\@testdef r}%
\def\bibcite{\@testdef b}%

```

We use \@@input to load the .aux file, so that it doesn't show up in the list of files produced by \listfiles.

```

26   \@tempswafalse
27   \makeatletter \@input\jobname.aux
28   \fi
29   \UseOneTimeHook{enddocument/afteraux}%

```

Next hook is expect to contain only code for writing info messages on the terminal.

```

30   \UseOneTimeHook{enddocument/info}%
31   \endgroup
32   \UseOneTimeHook{enddocument/end}%
33   \deadcycles\z@%%end}

```

The public hooks used in \enddocument:

```

34 \NewHook{enddocument}
35 \NewHook{enddocument/afterlastpage}
36 \NewHook{enddocument/afteraux}
37 \NewHook{enddocument/info}
38 \NewHook{enddocument/end}

```

This is one of the few places where we already add data and rules to a hook already in the kernel.

If we roll back we have to drop stuff before adding chunks, otherwise the code will just be appended, and thus doubled. This would result in a harmless warning during the format generation, because in that case the code chunk label doesn't exist, and therefore can't be dropped.

```

39 <latexrelease>\RemoveFromHook{enddocument/info}[kernel/filelist]
40 <latexrelease>\RemoveFromHook{enddocument/info}[kernel/warnings]
41 <latexrelease>\RemoveFromHook{enddocument/info}[kernel/release]
42 \AddToHook{enddocument/info}[kernel/filelist]{\@dofilelist}
43 \AddToHook{enddocument/info}[kernel/warnings]{\@enddocument@kernel@warnings}
44 \AddToHook{enddocument/info}[kernel/release]{%
45   \let\show@release@info\wlog
46   \show@release@info{ *****}%
47   \the\LaTeXReleaseInfo
48   \show@release@info{ *****}}
49
50 \DeclareHookRule{enddocument/info}{kernel/release}{before}{kernel/filelist}
51 \DeclareHookRule{enddocument/info}{kernel/filelist}{before}{kernel/warnings}

```

(End of definition for \enddocument.)

\@enddocument@kernel@warnings

```
52 \def\@enddocument@kernel@warnings{%
```

First we check for font size substitution bigger than \fontsubfuzz. The \relax is necessary because this is a macro not a register.

```
53 \ifdim \font@submax >\fontsubfuzz\relax
```

In case you wonder about the \@gobbletwo inside the message below, this is a horrible hack to remove the tokens \on@line. that are added by \@font@warning at the end.

```
54 \@font@warning{Size substitutions with differences\MessageBreak
55 up to \font@submax\space have occurred.\@gobbletwo}%
56 \fi
```

The macro \@defaultsubs is initially \relax but gets redefined to produce a warning if there have been some default font substitutions.

```
57 \@defaultsubs
```

The macro \@refundefined is initially \relax but gets redefined to produce a warning if there are undefined refs.

```
58 \@refundefined
```

If a label is defined more than once, \@tempswa will always be true and thus produce a “Label(s) may ...” warning. But since a rerun will not solve that problem (unless one uses a package like varioref that generates labels on the fly), we suppress this message.

```
59 \if@filesw
60 \ifx \@multiplelabels \relax
61 \if@tempswa
62 \@latex@warning@no@line{Label(s) may have changed.
63 Rerun to get cross-references right}%
64 \fi
65 \else
66 \@multiplelabels
67 \fi
68 \ifx \@extra@page@added \relax
69 \@latex@warning@no@line{Temporary extra page added at the end.
70 Rerun to get it removed}%
71 \fi
```

We could think of adding a warning that nothing can be corrected while \nofiles is in force. In the past the warnings related to the aux file are simply suppressed in this case.

```
72 \fi
73 }
```

(End of definition for \@enddocument@kernel@warnings.)

```
74 </2kernel | latexrelease>
75 <latexrelease>\EndIncludeInRelease
76 <latexrelease>\IncludeInRelease{2020/10/01}%
77 <latexrelease> \enddocument}{Use Hooks}%
78 <latexrelease>\def\enddocument{%
79 <latexrelease> \@kernel@before@enddocument
80 <latexrelease> \UseOneTimeHook{enddocument}%
81 <latexrelease> \@kernel@after@enddocument
```



```

82 <latexrelease> \@checkend{document}%
83 <latexrelease> \clearpage
84 <latexrelease> \UseOneTimeHook{enddocument/afterlastpage}%
85 <latexrelease> \@kernel@after@enddocument@afterlastpage
86 <latexrelease> \begingroup
87 <latexrelease> \if@filesw
88 <latexrelease> \immediate\closeout\@mainaux
89 <latexrelease> \let\@setckpt\@gobbletwo
90 <latexrelease> \let\@newl@bel\@testdef
91 <latexrelease> \@tempswafalse
92 <latexrelease> \makeatletter \@input\jobname.aux
93 <latexrelease> \fi
94 <latexrelease> \UseOneTimeHook{enddocument/afteraux}%
95 <latexrelease> \UseOneTimeHook{enddocument/info}%
96 <latexrelease> \endgroup
97 <latexrelease> \UseOneTimeHook{enddocument/end}%
98 <latexrelease> \deadcycles\z@\@end}
99 <latexrelease>\NewHook{enddocument}
100 <latexrelease>\NewHook{enddocument/afterlastpage}
101 <latexrelease>\NewHook{enddocument/afteraux}
102 <latexrelease>\NewHook{enddocument/info}
103 <latexrelease>\NewHook{enddocument/end}

```

If we roll back we have to drop stuff before adding chunks, otherwise the code will just be appended, and thus doubled.

```

104 <latexrelease>\RemoveFromHook{enddocument/info}[kernel/filelist]
105 <latexrelease>\RemoveFromHook{enddocument/info}[kernel/warnings]
106 <latexrelease>\RemoveFromHook{enddocument/info}[kernel/release]

107 <latexrelease>\AddToHook{enddocument/info}[kernel/filelist]{\@dofilelist}
108 <latexrelease>\AddToHook{enddocument/info}[kernel/warnings]{\@enddocument@kernel@warnings}
109 <latexrelease>\AddToHook{enddocument/info}[kernel/release]{%
110 <latexrelease> \let\show@release@info\wlog
111 <latexrelease> \show@release@info{ *****}%
112 <latexrelease> \the\LaTeXReleaseInfo
113 <latexrelease> \show@release@info{ *****}}
114 <latexrelease>

115 <latexrelease>\DeclareHookRule{enddocument/info}{kernel/release}{before}{kernel/filelist}
116 <latexrelease>\DeclareHookRule{enddocument/info}{kernel/filelist}{before}{kernel/warnings}
117 <latexrelease>\def\@enddocument@kernel@warnings{%
118 <latexrelease> \ifdim \font@submax >\fontsubfuzz\relax
119 <latexrelease> \@font@warning{Size substitutions with differences\MessageBreak
120 <latexrelease> up to \font@submax\space have occurred.\@gobbletwo}%
121 <latexrelease> \fi
122 <latexrelease> \@defaultsubs
123 <latexrelease> \@refundefined
124 <latexrelease> \if@filesw
125 <latexrelease> \ifx \@multiplelabels \relax
126 <latexrelease> \if@tempswa
127 <latexrelease> \@latex@warning@no@line{Label(s) may have changed.
128 <latexrelease> Rerun to get cross-references right}%
129 <latexrelease> \fi
130 <latexrelease> \else
131 <latexrelease> \@multiplelabels
132 <latexrelease> \fi

```

```

133 <latexrelease> \ifx \@extra@page@added \relax
134 <latexrelease> \@latex@warning@no@line{Temporary extra page added at the end.
135 <latexrelease> Rerun to get it removed}%
136 <latexrelease> \fi
137 <latexrelease> \fi
138 <latexrelease>}
139 <latexrelease>\EndIncludeInRelease
140 <latexrelease>\IncludeInRelease{0000/00/00}%
141 <latexrelease> \{\enddocument}\{Use Hooks}%
142 <latexrelease>
143 <latexrelease>\def\enddocument{%
144 <latexrelease> \let\AtEndDocument\@firstofone
145 <latexrelease> \@enddocumenthook
146 <latexrelease> \@checkend{document}%
147 <latexrelease> \clearpage
148 <latexrelease> \begingroup
149 <latexrelease> \if@filesw
150 <latexrelease> \immediate\closeout\@mainaux
151 <latexrelease> \let\@setckpt\@gobbletwo
152 <latexrelease> \let\@newl@bel\@testdef
153 <latexrelease> \@tempwafalse
154 <latexrelease> \makeatletter \@input\jobname.aux
155 <latexrelease> \fi
156 <latexrelease> \@dofilelist
157 <latexrelease> \ifdim \font@submax >\fontsubfuzz\relax
158 <latexrelease> \@font@warning{Size substitutions with differences\MessageBreak
159 <latexrelease> up to \font@submax\space have occurred.\@gobbletwo}%
160 <latexrelease> \fi
161 <latexrelease> \@defaultsubs
162 <latexrelease> \@refundefined
163 <latexrelease> \if@filesw
164 <latexrelease> \ifx \@multiplelabels \relax
165 <latexrelease> \if@tempswa
166 <latexrelease> \@latex@warning@no@line{Label(s) may have changed.
167 <latexrelease> Rerun to get cross-references right}%
168 <latexrelease> \fi
169 <latexrelease> \else
170 <latexrelease> \@multiplelabels
171 <latexrelease> \fi
172 <latexrelease> \fi
173 <latexrelease> \endgroup
174 <latexrelease> \deadcycles\z@\@end}
175 <latexrelease>
176 <latexrelease>\let\@enddocument@kernel@warnings\@undefined
177 <latexrelease>
178 <latexrelease>\EndIncludeInRelease
179 <*2kernel>

```

\@kernel@before@enddocument The \@kernel@before@enddocument hook is slightly different because we initialize it with \par so that \enddocument always returns to vertical mode as its first action.

```

180 </2kernel>
181 <*2kernel | latexrelease>
182 <latexrelease>\IncludeInRelease{2021/06/01}%
183 <latexrelease> \{\@kernel@before@enddocument}\{kernel before hook}%

```

```

184 \def\@kernel@before@enddocument{\par}
185 </2ekernel | latexrelease>
186 <latexrelease>\EndIncludeInRelease
    The rollback code renders it harmless.
187 <latexrelease>\IncludeInRelease{0000/00/00}%
188 <latexrelease>          {\@kernel@before@enddocument}{kernel before hook}%
189 <latexrelease>
190 <latexrelease>\let\@kernel@before@enddocument\@empty
191 <latexrelease>
192 <latexrelease>\EndIncludeInRelease
193 <*2ekernel>

```

(End of definition for \@kernel@before@enddocument.)

`\@testdef`

```

194 \def\@testdef #1#2#3{%
195   \def\reserved@a{#3}\expandafter \ifx \csname #1@#2\endcsname
196   \reserved@a \else \@tempwattrue \fi}

```

(End of definition for \@testdef.)

Reading data from auxiliary files (like `.toc` normally happens in vertical mode and it therefore doesn't matter if line endings are converted to spaces by \TeX during that process.

However, especially the `.toc` file might be read in L-R mode (in cases the `\tableofcontents` attempts to put, say, a list of sub-sections as a paragraph). In that case the newlines after a line like

```
\contentsline {subsubsection}{\numberline {1.1.1}A C-head}{2}
```

might result in spurious spaces (e.g., when that level is not included).

That could be fixed by reading in the file using `\endlinechar=-1` but that has the danger that it drops some valid endlines that should be converted to spaces (for example, when the user edited the TOC and then used `\nofiles` to preserve it).

So the approach taken instead is this:

- `\addcontentsline` adds the command `\protected@file@percent` to the end of the second argument of `\@writefile` that is written to the `.aux`. As the name indicates this is a protected macro so it doesn't change if it is written out.
- When the `.aux` is read back in at the end of the run, `\@writefile` is executed and writes its second argument unmodified to the file with the extension given by its first argument. Or rather that was how it was in the past.
- Instead we change `\@writefile` slightly: basically it looks at the second argument and if the last token in there is `\protected@file@percent` then it is replaced by a percent character and that is then written out. If not (for example, if the data came from a user issued `\addtocontents`, or from some package that uses `\@writefile` for writing its own files) then the command behaves exactly as before.

`\protected@file@percent` Dummy cs to be replaced by a percent sign inside `\@writefile`. If it survives (when used incorrectly) it will expand to nothing in a typesetting context.

```

197 </2ekernel>
198 <*2ekernel | latexrelease>

```

```

199 \latexrelease\IncludeInRelease{2018/12/01}%
200 \latexrelease)          {\protected@file@percent}{Mask line endings}%
201 \protected\def\protected@file@percent{}

```

(End of definition for \protected@file@percent.)

`\add@percent@to@temptokena` Helper function which is used to inspect a sequence of tokens (the second argument of `\@writefile` and if the last token is `\protected@file@percent` it will replace it by a harmless percent. The result is saved in `\@temptokena` for later use.

```

202 \catcode'\^^A=9
203 \long\gdef\add@percent@to@temptokena
204   #1\protected@file@percent#2\add@percent@to@temptokena

```

When we call this macro in `\@writefile` we stick in `\@empty` at the beginning, so that in case the tokenlist consists of a single brace group the braces aren't stripped. The `\expandafter` then expands this extra token away again.

```

205   {\expandafter\ifx\expandafter X\detokenize{#2}X\expandafter\dont@add@percent@to@temptokena
206     \expandafter\do@add@percent@to@temptokena\fi{#1}}
207 \long\def\dont@add@percent@to@temptokena#1{%
208   \@temptokena\expandafter{#1}}

```

`latexrelease` will read this code in high-speed mode in certain situations. During that it will only look for `\if` tests but not actually execute the `\catcode` change above. As a result it will drop anything after the `%` character in the definition. Therefore the `\fi` needs to be on the next line and we need locally another comment character to avoid getting spaces into the definition—a weird problem :-)

```

209 \begingroup
210 \catcode'\%=12
211 \catcode'\^^A=14
212 \long\gdef\do@add@percent@to@temptokena#1{\@temptokena\expandafter{#1}\^^A

```

Can't be on the same line as the `%` — see above.

```

213   }}
214 \endgroup

```

(End of definition for `\add@percent@to@temptokena`.)

`\@writefile`

```

215 \long\def\@writefile#1#2{%
216   \@ifundefined{tf@#1}\relax
217   {%

```

If we write to the file we first prepare `#2` using `\add@percent@to@temptokena` and then write the token register out.

```

218     \add@percent@to@temptokena
219     \@empty#2\protected@file@percent
220     \add@percent@to@temptokena
221     \immediate\write\csname tf@#1\endcsname{\the\@temptokena}%
222   }%
223 }

```

```

224 </2ekernel | latexrelease>
225 <latexrelease>\EndIncludeInRelease
226 <latexrelease>\IncludeInRelease{0000/00/00}%
227 <latexrelease>          {\protected@file@percent}{Mask line endings}%
228 <latexrelease>\let\protected@file@percent\@undefined
229 <latexrelease>\let\add@percent@to@temptokena\@undefined
230 <latexrelease>\let\do@add@percent@to@temptokena\@undefined
231 <latexrelease>\let\dont@add@percent@to@temptokena\@undefined
232 <latexrelease>\long\def\@writefile#1#2{%
233 <latexrelease>  \ifundefined{tf@#1}\relax
234 <latexrelease>    {\@temptokena{#2}}%
235 <latexrelease>    \immediate\write\csname tf@#1\endcsname{\the\@temptokena}%
236 <latexrelease>    }%
237 <latexrelease>}
238 <latexrelease>\EndIncludeInRelease
239 <*2ekernel>

```

(End of definition for \@writefile.)

`\stop`

```

240 \def\stop{\clearpage\deadcycles\z\let\par\@par\@end}

```

(End of definition for \stop.)

Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

```

241 \everypar{\@nodocument} %% To get an error if text appears before the
242 \nullfont                %% \begin{document}

```

`\begin`, `\end`, and `\@checkend` changed so `\end{document}` will catch an unmatched `\begin`. Changed 24 May 89 as suggested by Frank Mittelbach and Rainer Sch\"opf.

```

\begin{NAME} ==
BEGIN
  IF \NAME undefined THEN \reserved@a == BEGIN report error END
  ELSE \reserved@a ==
    (\@currenvir :=L NAME) \NAME
  FI
  @ignore :=G F      %% Added 30 Nov 88
  \begingroup
  \@endpe := F
  \@currenvir :=L NAME
  \NAME
END

```

```

\end{NAME} ==
BEGIN
  \endNAME
  \@checkend{NAME}
  \endgroup
  IF @endpe = T      %% @endpe set True by \@endparenv

```

```

        THEN \@doendpe           %% \@doendpe redefines \par and \everypar
                                %% to suppress paragraph indentation in
        FI                       %% immediately following text
        IF @ignore = T
            THEN @ignore :=G F
                \ignorespaces
        FI
    END

\@checkend{NAME} ==
BEGIN
    IF \@currentenv = NAME
        ELSE \@badend{NAME}
    FI
END

```

End of historical L^AT_EX 2.09 comments.

`\begin`

```

243 </2kernel>
244 <*2kernel | latexrelease>
245 <latexrelease>\IncludeInRelease{2020/10/01}%
246 <latexrelease>          {\begin}{Use hook system}%
247 \protected\def\begin#1{%
248     \UseHook{env/#1/before}%
249     \@ifundefined{#1}%
250         {\def\reserved@a{\@latex@error{Environment #1 undefined}\@eha}}%
251         {\def\reserved@a{\def\@currentenv{#1}%
252             \edef\@currentvline{\on@line}%
253             \@execute@begin@hook{#1}%
254             \csname #1\endcsname}}}%
255     \@ignorefalse
256     \begingroup\@endpfalse\reserved@a}

```

Before the `\document` code is executed we have to first undo the `\endgroup` as there should be none for this environment to avoid that changes on top-level unnecessarily go to T_EX's savestack, and we have to initialize all hooks in the hook system. So we need to test for this environment name. But once it has been found all this testing is no longer needed and so we redefine `\@execute@begin@hook` to simply use the hook.

```

257 \def\@execute@begin@hook #1{%
258     \expandafter\ifx\csname #1\endcsname\document
259     \endgroup
260     \gdef\@execute@begin@hook##1{\UseHook{env/##1/begin}}%
261     \expl@@@initialize@all@@
262     \fi

```

If this is an environment before `\begin{document}` we just run the hook so this can be outside the test.

```

263     \UseHook{env/#1/begin}%
264 }
265 </2kernel | latexrelease>
266 <latexrelease>\EndIncludeInRelease

```

```

267 <latexrelease>\IncludeInRelease{2019/10/01}%
268 <latexrelease>          {\begin}{Making \begin/\end robust}%
269 <latexrelease>\DeclareRobustCommand\begin[1]{%
270 <latexrelease>  \ifundefined{#1}%
271 <latexrelease>    {\def\reserved@a{\@latex@error{Environment #1 undefined}\@eha}}%
272 <latexrelease>    {\def\reserved@a{\def\@currentvir{#1}%
273 <latexrelease>      \edef\@currentvline{\on@line}%
274 <latexrelease>      \csname #1\endcsname}}}%
275 <latexrelease>  \ignorefalse
276 <latexrelease>  \begingroup\@endpfalse\reserved@a}
277 <latexrelease>\EndIncludeInRelease

```

A version that doesn't start out with `\relax` when in typesetting mode would be the following, but since `\begin` issues a `\begingroup` it wouldn't help much with respect to allowing things like `\noalign` or `\multicolumn` inside.

```

278 %\edef\begin
279 %  {\unexpanded{%
280 %    \ifx\protect\@typeset@protect
281 %      \expandafter\@gobble
282 %    \fi
283 %    \protect
284 %  }%
285 %  \expandafter\noexpand\csname begin \endcsname
286 %  }
287 %\@namedef{begin }#1{%
288 %  \ifundefined{#1}%
289 %    {\def\reserved@a{\@latex@error{Environment #1 undefined}\@eha}}%
290 %    {\def\reserved@a{\def\@currentvir{#1}%
291 %      \edef\@currentvline{\on@line}%
292 %      \csname #1\endcsname}}}%
293 %  \ignorefalse
294 %  \begingroup\@endpfalse\reserved@a}
295 <latexrelease>\IncludeInRelease{0000/00/00}%
296 <latexrelease>          {\begin}{Making \begin/\end robust}%
297 <latexrelease>\def\begin#1{%
298 <latexrelease>  \ifundefined{#1}%
299 <latexrelease>    {\def\reserved@a{\@latex@error{Environment #1 undefined}\@eha}}%
300 <latexrelease>    {\def\reserved@a{\def\@currentvir{#1}%
301 <latexrelease>      \edef\@currentvline{\on@line}%
302 <latexrelease>      \csname #1\endcsname}}}%
303 <latexrelease>  \ignorefalse
304 <latexrelease>  \begingroup\@endpfalse\reserved@a}
305 <latexrelease>

```

Also undo the internal commands as some packages unfortunately test for their existence instead of using `\IfFormatAtLeastTF`.

```

306 <latexrelease>\expandafter\let\csname begin \endcsname\@undefined
307 <latexrelease>
308 <latexrelease>\EndIncludeInRelease
309 <*2ekernel>

```

(End of definition for \begin.)

`\end` The top level definition for `\end`.

```

310 </2ekernel>
311 <*2ekernel | latexrelease>
312 <latexrelease>\IncludeInRelease{2019/10/01}%
313 <latexrelease>          {\end}{Making \begin/\end robust}%

```

While `\begin` was made robust simply by using `\DeclareRobustCommand` we need to be a bit more subtle with `\end` as there are packages out there that try to look into the top-level contents of `\end{foo}` (that is at the expansion of `\endfoo`) to see if it contains certain macros. This is done by hitting `\end{foo}` with three `\expandafters`, the first to get

```
\csname endfoo\endcsname          \@checkend{foo}% etc.
```

the second to expand the `\csname`, i.e., to get to

```
\endfoo                          \@checkend{foo}% etc.
```

and the third to finally get to the top-level content of `\endfoo`, i.e.

```
<top-level content of \endfoo> \@checkend{foo}% etc.
```

Therefore a robust replacement should produce the same results after three expansions (there first is obviously different).

Basically the definition of `\end` should either produce `\protect\end_` (when not doing typesetting) or it should produce `\end_` (without the `\protect`) when doing typesetting. Furthermore, it should (when in typesetting mode) show exactly the same result as `\end_` (which is the original fragile definition of `\end`) when you expand either of them twice, i.e.,

```
\endfoo                          \@checkend{foo}% etc.
```

That is achieved with the code below (which is worth studying carefully).

There is some trickery involved here: in particular we use `\romannumeral` to change a single expansion into three successive expansions in one go. That primitive expands until it has scanned a number (0 in this case, so it doesn't produce any output) and so it allows us to place arbitrary many `\expandafters` inside that are all going to be executed when `\romannumeral` is hit by a single `\expandafter`.

```

314 \edef\end
315   {\unexpanded{%
316     \romannumeral
317     \ifx\protect\@typeset@protect
318     \expandafter      %1
319     \expandafter      %2
320     \expandafter      %1
321     \expandafter      %3 expands the \csname inside \end<space>
322     \expandafter      %1
323     \expandafter      %2 expands \end<space>
324     \expandafter      %1 expands the \else
325     \z@
326     \else
327     \expandafter\z@\expandafter\protect
328     \fi
329   }%
330   \expandafter\noexpand\csname end \endcsname
331 }
332 </2ekernel | latexrelease>
333 <latexrelease>\EndIncludeInRelease

```


And here is the original definition of `\end` the way it was in L^AT_EX for several decades now hidden in `\end_`.

```

334 <latexrelease>\IncludeInRelease{0000/00/00}%
335 <latexrelease>          {\end}{Making \begin/\end robust}%
336 <latexrelease>\def\end#1{%
337 <latexrelease>  \csname end#1\endcsname\@checkend{#1}%
338 <latexrelease>  \expandafter\endgroup\if@endpe\@doendpe\fi
339 <latexrelease>  \if@ignore\@ignorefalse\ignorespaces\fi}
340 <latexrelease>
341 <latexrelease>\EndIncludeInRelease
342 <*2ekernel>

```

(End of definition for `\end`.)

`\end\verbvisiblespace` The internal version with a space at the end.

```

343 </2ekernel>
344 <*2ekernel | latexrelease>
345 <latexrelease>\IncludeInRelease{2024/11/01}%
346 <latexrelease>          {\end!space}{New @endpe handling}%
347 \@namedef{end }#1{%
348   \romannumeral
349   \IfHookEmptyTF{env/#1/end}%
350     {\expandafter\z@}%
351     {\z@\UseHook{env/#1/end}}}%
352   \csname end#1\endcsname\@checkend{#1}%

```

We can now close the environment group and due to the new `\if@endpe` handling we no longer need to `\expandafter` out of the group.

```

353 %   \expandafter\endgroup\if@endpe\@doendpe\fi
354   \endgroup
355   \UseHook{env/#1/after}%
356   \if@ignore\@ignorefalse\ignorespaces\fi
357 }
358 </2ekernel | latexrelease>
359 <latexrelease>\EndIncludeInRelease

```

Version that adds hooks (so different from the 2019 version). It fixes tlb3722 but the change should perhaps be made in `tabularx` instead.

```

360 %   \begin{macrocode}
361 <latexrelease>\IncludeInRelease{2020/10/01}%
362 <latexrelease>          {\end!space}{Use hook system}%
363 <latexrelease>
364 <latexrelease>\@namedef{end }#1{%
365 <latexrelease>  \romannumeral
366 <latexrelease>    \IfHookEmptyTF{env/#1/end}%
367 <latexrelease>      {\expandafter\z@}%
368 <latexrelease>      {\z@\UseHook{env/#1/end}}}%
369 <latexrelease>    \csname end#1\endcsname\@checkend{#1}%
370 <latexrelease>    \expandafter\endgroup\if@endpe\@doendpe\fi
371 <latexrelease>    \UseHook{env/#1/after}%
372 <latexrelease>    \if@ignore\@ignorefalse\ignorespaces\fi
373 <latexrelease>}
374 <latexrelease>\EndIncludeInRelease

```

Version without the fix for tlb3722 for the record:

```

@namedef{end }#1{%
  \UseHook{env/#1/end}%
  \csname end#1\endcsname\@checkend{#1}%
  \expandafter\endgroup\if@endpe\@doendpe\fi
  \UseHook{env/#1/after}%
  \if@ignore\@ignorefalse\ignorespaces\fi}%
375 <latexrelease>\IncludeInRelease{2019/10/01}%
376 <latexrelease>          {\end!space}{Making \begin/\end robust}%
377 <latexrelease>
378 <latexrelease>\@namedef{end }#1{%
379 <latexrelease>  \csname end#1\endcsname\@checkend{#1}%
380 <latexrelease>  \expandafter\endgroup\if@endpe\@doendpe\fi
381 <latexrelease>  \if@ignore\@ignorefalse\ignorespaces\fi}
382 <latexrelease>\EndIncludeInRelease

383 <latexrelease>\IncludeInRelease{0000/00/00}%
384 <latexrelease>          {\end!space}{Making \begin/\end robust}%

```

Undo the internal command as some packages unfortunately test for their existence instead of using `\IfFormatAtLeastTF`.

```

385 <latexrelease>\expandafter\let\csname end \endcsname\undefined
386 <latexrelease>
387 <latexrelease>\EndIncludeInRelease
388 <*2ekernel>

```

(End of definition for `\end\verbvisiblespace`.)

`\@checkend`

```

389 \def\@checkend#1{\def\reserved@a{#1}\ifx
390   \reserved@a\@currenvir \else\@badend{#1}\fi}

```

(End of definition for `\@checkend`.)

`\@currenvline`

We do need a default value for `\@currenvline` on top-level since the document environment cancels the brace group. This means that a mismatch with `\begin{document}` will not produce a line number. Thus the outer default must be `\@empty` or we will end up with two spaces.

```

391 \let\@currenvline\@empty

```

(End of definition for `\@currenvline`.)

`\AtBeginEnvironment`
`\AtEndEnvironment`
`\BeforeBeginEnvironment`
`\AfterEndEnvironment`

We provide 4 high-level hook interfaces directly, the others only when `etoolbox` is loaded

```

392 </2ekernel>
393 <*2ekernel | latexrelease>
394 <latexrelease>\IncludeInRelease{2020/10/01}%
395 <latexrelease>          {\AtBeginEnvironment}{Hooks for environments}%
396 \newcommand\AtBeginEnvironment[2][.]  {\AddToHook{env/#2/begin}[#1]}
397 \newcommand\AtEndEnvironment[2][.]    {\AddToHook{env/#2/end}[#1]}
398 \newcommand\BeforeBeginEnvironment[2][.]{\AddToHook{env/#2/before}[#1]}
399 \newcommand\AfterEndEnvironment[2][.]{\AddToHook{env/#2/after}[#1]}
400 </2ekernel | latexrelease>
401 <latexrelease>\EndIncludeInRelease

```

```

402 <latexrelease>\IncludeInRelease{0000/00/00}%
403 <latexrelease>                {\AtBeginEnvironment}{Hooks for environments}%
404 <latexrelease>
405 <latexrelease>\let\AtBeginEnvironment\@undefined
406 <latexrelease>\let\AtEndEnvironment\@undefined
407 <latexrelease>\let\BeforeBeginEnvironment\@undefined
408 <latexrelease>\let\AfterEndEnvironment\@undefined
409 <latexrelease>
410 <latexrelease>\EndIncludeInRelease
411 <*2ekernel>

```

(End of definition for `\AtBeginEnvironment` and others. These functions are documented on page 223.)

1.2 Center, Flushright, Flushleft

```
412 \message{center,}
```

Historical *L^AT_EX* 2.09 comments (not necessarily accurate any more):

```

\center, \flushright and \flushleft set
  \rightskip = 0pt or \@flushglue (as appropriate)
  \leftskip  = 0pt or \@flushglue (as appropriate)
  \parindent = 0pt
  \parfillskip = 0pt. (except \flushleft)
  \\\          == \par \vskip -\parskip
  \\[LENGTH] == \\\ \vskip LENGTH
  \\\*         == \par \penalty 10000 \vskip -\parskip
  \\[LEN]      == \\\* \vskip LENGTH

```

They invoke the `trivlist` environment to handle vertical spacing before and after them.

`\centering`, `\raggedright` and `\raggedleft` are the declaration analogs of the above.

`\raggedright` has a more universal effect, however. It sets `\@rightskip := flushglue`. Every environment, like the list environments, that set `\rightskip` to its 'normal' value set it to `\@rightskip`

End of historical *L^AT_EX* 2.09 comments.

```
\@centercr
```

```

413 </2ekernel>
414 <*2ekernel | latexrelease>
415 <latexrelease>\IncludeInRelease{2020/02/02}%
416 <latexrelease>                {\@centercr}{Make robust}%
417 \protected\def\@centercr{\ifhmode \unskip\else \@nolnerr\fi
418     \par\@ifstar{\nobreak\@xcentercr}\@xcentercr}
419 </2ekernel | latexrelease>

```

```

420 <latexrelease>\EndIncludeInRelease
421 <latexrelease>\IncludeInRelease{0000/00/00}%
422 <latexrelease>          {\@centercr}{Make robust}%
423 <latexrelease>
424 <latexrelease>\def\@centercr{\ifhmode \unskip\else \@nolnerr\fi
425 <latexrelease>          \par\@ifstar{\nobreak\@xcentercr}\@xcentercr}
426 <latexrelease>
427 <latexrelease>\EndIncludeInRelease
428 <*2ekernel>

```

(End of definition for \@centercr.)

\@xcentercr

```

429 \def\@xcentercr{\addvspace{-\parskip}\@ifnextchar
430   [\@icentercr\ignorespaces}

```

(End of definition for \@xcentercr.)

\@icentercr

```

431 </2ekernel>
432 <*2ekernel | latexrelease>
433 <latexrelease>\IncludeInRelease{2020/10/01}%
434 <latexrelease>          {\@icentercr}{centering, etc support calc}%
435 \def\@icentercr[#1]{\vspace@calcify{#1}\ignorespaces}
436 </2ekernel | latexrelease>
437 <latexrelease>\EndIncludeInRelease
438 <latexrelease>\IncludeInRelease{0000/00/00}%
439 <latexrelease>          {\@icentercr}{centering, etc support calc}%
440 <latexrelease>
441 <latexrelease>\def\@icentercr[#1]{\vskip #1\ignorespaces}
442 <latexrelease>\EndIncludeInRelease
443 <*2ekernel>

```

(End of definition for \@icentercr.)

center (*env.*) We use \relax to prevent \item scanning too far.

```

444 \def\center{\trivlist \centering\item\relax}
445 \def\endcenter{\endtrivlist}
446 </2ekernel>
447 <*2ekernel | latexrelease>
448 <latexrelease>\IncludeInRelease{2020/10/01}%
449 <latexrelease>          {\centering}{Set finalhyphendemerits}%

```

\centering

```

450 \DeclareRobustCommand\centering{%
451   \let\\\@centercr
452   \rightskip\@flushglue\leftskip\@flushglue
453   \finalhyphendemerits=\z@
454   \parindent\z@\parfillskip\z@skip}

```

(End of definition for \centering.)

`\raggedright`

```
455 \DeclareRobustCommand\raggedright{%  
456   \let\\@centercr\@rightskip\@flushglue \rightskip\@rightskip  
457   \finalhyphendemerits=\z@  
458   \leftskip\z@skip  
459   \parindent\z@}
```

(End of definition for \raggedright.)

`\raggedleft`

```
460 \DeclareRobustCommand\raggedleft{%  
461   \let\\@centercr  
462   \rightskip\z@skip\leftskip\@flushglue  
463   \finalhyphendemerits=\z@  
464   \parindent\z@\parfillskip\z@skip}
```

(End of definition for \raggedleft.)

```
465 </2ekernel | latexrelease>  
466 <latexrelease>\EndIncludeInRelease  
467 <latexrelease>\IncludeInRelease{2019/10/01}%  
468 <latexrelease>           {\centering}{Make commands robust}%  
469 <latexrelease>  
470 <latexrelease>\DeclareRobustCommand\centering{%  
471 <latexrelease>   \let\\@centercr  
472 <latexrelease>   \rightskip\@flushglue\leftskip\@flushglue  
473 <latexrelease>   \parindent\z@\parfillskip\z@skip}  
474 <latexrelease>\DeclareRobustCommand\raggedright{%  
475 <latexrelease>   \let\\@centercr\@rightskip\@flushglue \rightskip\@rightskip  
476 <latexrelease>   \leftskip\z@skip  
477 <latexrelease>   \parindent\z@}  
478 <latexrelease>\DeclareRobustCommand\raggedleft{%  
479 <latexrelease>   \let\\@centercr  
480 <latexrelease>   \rightskip\z@skip\leftskip\@flushglue  
481 <latexrelease>   \parindent\z@\parfillskip\z@skip}  
482 <latexrelease>\EndIncludeInRelease  
483 <latexrelease>  
484 <latexrelease>\IncludeInRelease{0000/00/00}%  
485 <latexrelease>           {\centering}{Make commands robust}%  
486 <latexrelease>  
487 <latexrelease>\kernel@make@fragile\centering  
488 <latexrelease>\kernel@make@fragile\raggedright  
489 <latexrelease>\kernel@make@fragile\raggedleft  
490 <latexrelease>  
491 <latexrelease>\EndIncludeInRelease  
492 <*2ekernel)
```

`\@rightskip`

```
493 \newskip\@rightskip \@rightskip \z@skip
```

(End of definition for \@rightskip.)

`flushleft (env.)` We use `\relax` to prevent `\item` scanning too far.

```
494 \def\flushleft{\trivlist \raggedright\item\relax}
```

```
495 \def\endflushleft{\endtrivlist}
```

`flushright` (*env.*) We use `\relax` to prevent `\item` scanning too far.

```
496 \def\flushright{\trivlist \raggedleft\item\relax}
497 \def\endflushright{\endtrivlist}
```

1.3 Verbatim

```
498 \message{verbatim,}
```

The `verbatim` environment uses the fixed-width `\ttfamily` font, turns blanks into spaces, starts a new line for each carriage return (or sequence of consecutive carriage returns), and interprets *every* character literally. I.e., all special characters `\`, `{`, `$`, etc. are `\catcode`'d to 'other'.

The command `\verb` produces in-line verbatim text, where the argument is delimited by any pair of characters. E.g., `\verb #...#` takes '...' as its argument, and sets it verbatim in `\ttfamily` font.

The `*`-variants of these commands are the same, except that spaces print as the \TeX book's space character instead of as blank spaces.

`\@vobeyspaces`

`\@vobeytabs`

```
499 </2kernel>
500 <latexrelease>\IncludeInRelease{2023/11/01}%
501 <latexrelease>          {\@vobeytabs}{0beyed tabs}%
502 <*2kernel | latexrelease>
503 {\catcode'\ =\active%
504 \gdef\@vobeyspaces{\catcode'\ \active\let \@xobeysp\@vobeytabs}}
505 {\catcode'\^^I=\active
506 \gdef\@vobeytabs{\catcode'\^^I\active\let^^I\@xobeytab}%
507 \global\let^^I=\space
508 }
509 </2kernel | latexrelease>
510 <latexrelease>\EndIncludeInRelease
511 <latexrelease>\IncludeInRelease{0000/00/00}%
512 <latexrelease>          {\@vobeytabs}{0beyed tabs}%
513 <latexrelease>{\catcode'\ =\active%
514 <latexrelease>\gdef\@vobeyspaces{\catcode'\ \active\let \@xobeysp}}
515 <latexrelease>\let\@vobeytabs\@undefined
516 <latexrelease>\EndIncludeInRelease
517 <*2kernel>
```

(End of definition for `\@vobeyspaces` and `\@vobeytabs`.)

`\@xobeysp`

(End of definition for `\@xobeysp`.)

`\@xverbatim`

`\@sxverbatim`

```
518 \begingroup \catcode '|=0 \catcode '[= 1
519 \catcode']=2 \catcode '\{=12 \catcode '\}=12
520 \catcode'\=12 |gdef|\@xverbatim#1\end{verbatim} [#1|end[verbatim]]
521 |gdef|\@sxverbatim#1\end{verbatim*} [#1|end[verbatim*]]
522 |endgroup
```

(End of definition for `\@xverbatim` and `\@sxverbatim`.)

`\@verbatim` Real start of verbatim environment We use `\relax` to prevent `\item` scanning too far.

```
523 </2kernel>
524 <*2kernel | latexrelease>
525 <latexrelease>\IncludeInRelease{2017-04-15}{\@verbatim}%
526 <latexrelease>          {Disable hyphenation in verbatim}%
527 \def\@verbatim{\trivlist \item\relax
528   \if@minipage\else\vskip\parskip\fi
529   \leftskip\@totalleftmargin\rightskip\z@skip
530   \parindent\z@\parfillskip\@flushglue\parskip\z@skip
```

Added `\@@par` to clear possible `\parshape` definition from a surrounding list (the verbatim guru says). Switch language when in vertical mode.

```
531   \@@par
```

Set `\language` here to suppress hyphenation. Done this way rather than setting `\hyphenchar` as that is a global setting.

```
532   \language\l@nohyphenation
533   \@tempwafalse
534   \def\par{%
535     \if@tempswa
```

A `\leavevmode` added: needed if, for example, a blank verbatim line is the first thing in a list item (wow!).

```
536     \leavevmode \null \@@par\penalty\interlinepenalty
537   \else
538     \@tempswatrue
539     \ifhmode\@@par\penalty\interlinepenalty\fi
540   \fi}%
```

To allow customization we hide the font used in a separate macro.

```
541   \let\do\@makeother \dospecials
542   \obeylines \verbatim@font \@noligs
```

To avoid a breakpoint after the labels box, we remove the penalty put there by the list macros: another use of `\unpenalty`!

```
543   \everypar \expandafter{\the\everypar \unpenalty}%
544 }
545 </2kernel | latexrelease>
546 <latexrelease>\EndIncludeInRelease
547 <latexrelease>\IncludeInRelease{0000-00-00}{\@verbatim}%
548 <latexrelease>          {Disable hyphenation in verbatim}%
549 <latexrelease>\def\@verbatim{\trivlist \item\relax
550 <latexrelease>   \if@minipage\else\vskip\parskip\fi
551 <latexrelease>   \leftskip\@totalleftmargin\rightskip\z@skip
552 <latexrelease>   \parindent\z@\parfillskip\@flushglue\parskip\z@skip
553 <latexrelease>   \@@par
554 <latexrelease>   \@tempwafalse
555 <latexrelease>   \def\par{%
556 <latexrelease>     \if@tempswa
557 <latexrelease>       \leavevmode \null \@@par\penalty\interlinepenalty
558 <latexrelease>     \else
559 <latexrelease>       \@tempswatrue
560 <latexrelease>       \ifhmode\@@par\penalty\interlinepenalty\fi
561 <latexrelease>     \fi}%
562 <latexrelease>   \let\do\@makeother \dospecials
```

```

563 <latexrelease> \obeylines \verbatim@font \@noligs
564 <latexrelease> \hyphenchar\font\m@ne
565 <latexrelease> \everypar \expandafter{\the\everypar \unpenalty}%
566 <latexrelease>}
567 <latexrelease>\EndIncludeInRelease
568 <*2ekernel>

```

(End of definition for \@verbatim.)

verbatim (RmS 93/09/19) Protected against ‘missing item’ error message triggered by empty verbatim environment.

```

569 \def\verbatim{\@verbatim \frenchspacing\@vobeyspaces \@xverbatim}
570 \def\endverbatim{\if@newlist \leavevmode\fi\endtrivlist}

```

(End of definition for \verbatim and \endverbatim.)

\verbatim@font Macro to select the font used for verbatim typesetting. It also does other work if necessary for the font used.

```

571 \def\verbatim@font{\normalfont\ttfamily}

```

(End of definition for \verbatim@font.)

```

572 </2ekernel>
573 <*2ekernel | latexrelease>
574 <latexrelease>\IncludeInRelease{2018/12/01}%
575 <latexrelease>          {\verbvisiblespace}{Setup visible space for \verb}%

```

\asciispace The character in slot 32, in typewriter fonts (historically) a visible space but in other fonts a real space or something else

```

576 \DeclareRobustCommand\asciispace{\char 32 }

```

(End of definition for \asciispace.)

\verbvisiblespace This defines how to get a visible space in **\verb*** and friends. In classic T_EX this is just the slot 32, but in T_U encoded fonts we switch fonts and take the character from cmtt.

```

577 \ifx\Umathcode\@undefined
578   \let\verbvisiblespace\asciispace          % Pdftex version
579 \else
580   \DeclareRobustCommand\verbvisiblespace
581     {\leavevmode{\usefont{OT1}{cmtt}{m}{n}\asciispace}} % xetex/luatex version
582 \fi

```

(End of definition for \verbvisiblespace.)

\@verbvisiblespacebox The box to hold the visible space character if it isn’t in slot 32 in the current typewriter font.

```

583 \newbox\@verbvisiblespacebox

```

(End of definition for \@verbvisiblespacebox.)

verbatim* (*env.*) For **verbatim*** we also set up the correct visible space character definition and then run **\@vobeyspaces**. As this code is not called as part of the normal verbatim environment (the method is done the other way around this time) we don’t have to check if space is already active—it shouldn’t be.

```

584 \@namedef{verbatim*}{\@verbatim
585   \@setupverbvisiblespace
586   \frenchspacing\@vobeyspaces\@sxverbatim}
587 \expandafter\let\csname endverbatim*\endcsname =\endverbatim

```



```

588 </2ekernel | latexrelease>
589 <latexrelease>\EndIncludeInRelease
590 <latexrelease>\IncludeInRelease{0000/00/00}%
591 <latexrelease>          {\verbvisiblespace}{Setup visible space for \verb}%
592 <latexrelease>
593 <latexrelease>\@namedef{verbatim*}{\@verbatim\@sxverbatim}
594 <latexrelease>
595 <latexrelease>\let\asciispace          \@undefined
596 <latexrelease>\let\verbvisiblespace    \@undefined
597 <latexrelease>\let\@setupverbvisiblespace\@undefined
598 <latexrelease>\let\@verbvisiblespacebox \@undefined
599 <latexrelease>\EndIncludeInRelease
600 <*2ekernel>

```

`\@setupverbvisiblespace` In pdfTeX a catcode 12 space will produce the character in slot 32 which is assumed to be a visible space character (in a typewriter font in OT1 or T1 encoding). In XeTeX or LuaTeX a font in TU encoding is normally used and that has a real space in this slot. So what we do in this case is this: we check the definition of `\verbvisiblespace` and if it is `\asciispace` we assume that the char32 can be used (e.g., in pdfTeX). We then redefine `\@xobeysp` so that after running `\@xobeyspaces` we get characters from slot 32 for each active space.

```

601 </2ekernel>
602 <*2ekernel | latexrelease>
603 <latexrelease>\IncludeInRelease{2023/11/01}%
604 <latexrelease>          {\@setupverbvisiblespace}{Setup visible tab for \verb}%
605 \def\@setupverbvisiblespace{%
606   \ifx\verbvisiblespace\asciispace
607     \let\@xobeysp\asciispace
608   \else

```

Otherwise we measure the width of a character in the mono-spaced current font and place a `\verbvisiblespace` into a box of the right width which we are then using as the character for a space. By default this will be the space character from OT1 cmtt but by changing `\verbvisiblespace` one could use, for example, the `\textvisiblespace` of the current typewriter font.

```

609   \setbox\z@\hbox{x}%
610   \setbox\@verbvisiblespacebox\hbox to\wd\z@{\hss\verbvisiblespace\hss}%
611   \def\@xobeysp{\leavevmode\copy\@verbvisiblespacebox}%
612 \fi
613 \@setupverbvisibletab
614 }
615 </2ekernel | latexrelease>
616 <latexrelease>\EndIncludeInRelease
617 <latexrelease>\IncludeInRelease{2018/12/01}%
618 <latexrelease>          {\@setupverbvisiblespace}{Setup visible space for \verb}%
619 <latexrelease>\def\@setupverbvisiblespace{%
620 <latexrelease>   \ifx\verbvisiblespace\asciispace
621 <latexrelease>     \let\@xobeysp\asciispace
622 <latexrelease>   \else
623 <latexrelease>     \setbox\z@\hbox{x}%
624 <latexrelease>     \setbox\@verbvisiblespacebox\hbox to\wd\z@{\hss\verbvisiblespace\hss}%
625 <latexrelease>     \def\@xobeysp{\leavevmode\copy\@verbvisiblespacebox}%
626 <latexrelease>   \fi

```

```

627 <latexrelease>}
628 <latexrelease>\EndIncludeInRelease
629 <latexrelease>\IncludeInRelease{0000/00/00}%
630 <latexrelease>          {\@setupverbvisiblespace}{Setup visible space for \verb}%
631 <latexrelease>\let\@setupverbvisiblespace\@undefined
632 <latexrelease>\EndIncludeInRelease
633 <*2ekernel>

```

(End of definition for \@setupverbvisiblespace.)

`\@setupverbvisibletab` A redirection: just a simple wrapper.

```

634 </2ekernel>
635 <latexrelease>\IncludeInRelease{2023/11/01}%
636 <latexrelease>          {\@setupverbvisibletab}{Setup visible tab for \verb}%
637 <*2ekernel | latexrelease>
638 \def\@setupverbvisibletab{\let\@xobeytab\@xobeysp}
639 </2ekernel | latexrelease>
640 <latexrelease>\EndIncludeInRelease
641 <latexrelease>\IncludeInRelease{0000/00/00}%
642 <latexrelease>          {\@setupverbvisibletab}{Setup visible tab for \verb}%
643 <latexrelease>\let\@setupverbvisibletab\@undefined
644 <latexrelease>\EndIncludeInRelease
645 <*2ekernel>

```

(End of definition for \@setupverbvisibletab.)

`\@sverb` Definitions of `\@sverb` and `\@verb` changed so `\verb+ foo+` does not lose leading blanks
`\@@sverb` when it comes at the beginning of a line. Change made 24 May 89. Suggested by Frank Mittelbach and Rainer Schöpf.

```

646 </2ekernel>
647 <*2ekernel | latexrelease>
648 <latexrelease>\IncludeInRelease{2023/11/01}%
649 <latexrelease>          {\@sverb}{Support visible tabs}%

```

If the users types `\verb !~! foo` then surprisingly we would get the space as the delimiter and thus “!~!foo” in the output. To avoid this scenario we check if `#1` has the character code of a space, if so we recurse otherwise we call `\@@sverb` (which is the original definition of `\@sverb`).

```

650 \def\@sverb#1{\if\noexpand#1 \expandafter\@sverb\else\@@sverb{#1}\fi}

651 \def\@@sverb#1{%
652   \catcode'#1\active
653   \lccode'\~'#1%
654   \gdef\verb@balance@group{\verb@egroup
655     \@latex@error{\noexpand\verb illegal in argument}\@ehc}%
656   \aftergroup\verb@balance@group
657   \lowercase{\let~\verb@egroup}%

```

If `\@sverb` is called from `\@verb` then space is already active and supposed to produce a real space. In this case we do nothing. Otherwise we run `\@setupverbvisiblespace` to setup the right visible space char and afterwards `\@vobeyspaces` to make it the definition for the active space character.

```

658   \ifnum0%
659     \ifnum\catcode'\ =\active\else 1\fi
660     \ifnum\catcode'\~I=\active\else 1\fi

```

```

661     =0 %
662     \else \@setupverbvisiblespace \@vobeyspaces \fi
663 }
664 </2ekernel | latexrelease>
665 <latexrelease>\EndIncludeInRelease
666 <latexrelease>\IncludeInRelease{2020/10/01}%
667 <latexrelease>          {\@sverb}{Drop spaces before \verb delimiter}%
668 <latexrelease>\def\@sverb#1{%
669 <latexrelease>  \catcode'#1\active
670 <latexrelease>  \lccode'\~'#1%
671 <latexrelease>  \gdef\verb@balance@group{\verb@egroup
672 <latexrelease>    \@latex@error{\noexpand\verb illegal in argument}\@ehc}%
673 <latexrelease>  \aftergroup\verb@balance@group
674 <latexrelease>  \lowercase{\let~\verb@egroup}%
675 <latexrelease>  \ifnum\catcode'\ =\active
676 <latexrelease>  \else \@setupverbvisiblespace \@vobeyspaces \fi
677 <latexrelease>}
678 <latexrelease>\EndIncludeInRelease
679 <latexrelease>\IncludeInRelease{2018/12/01}%
680 <latexrelease>          {\@sverb}{Setup visible space for \verb}%
681 <latexrelease>
682 <latexrelease>\def\@sverb#1{%
683 <latexrelease>  \catcode'#1\active
684 <latexrelease>  \lccode'\~'#1%
685 <latexrelease>  \gdef\verb@balance@group{\verb@egroup
686 <latexrelease>    \@latex@error{\noexpand\verb illegal in command argument}\@ehc}%
687 <latexrelease>  \aftergroup\verb@balance@group
688 <latexrelease>  \lowercase{\let~\verb@egroup}%
689 <latexrelease>  \ifnum\catcode'\ =\active
690 <latexrelease>  \else \@setupverbvisiblespace \@vobeyspaces \fi
691 <latexrelease>}
692 <latexrelease>\let\@sverb\undefined
693 <latexrelease>\EndIncludeInRelease
694 <latexrelease>
695 <latexrelease>\IncludeInRelease{0000/00/00}%
696 <latexrelease>          {\@sverb}{Setup visible space for \verb}%
697 <latexrelease>\def\@sverb#1{%
698 <latexrelease>  \catcode'#1\active
699 <latexrelease>  \lccode'\~'#1%
700 <latexrelease>  \gdef\verb@balance@group{\verb@egroup
701 <latexrelease>    \@latex@error{\noexpand\verb illegal in command argument}\@ehc}%
702 <latexrelease>  \aftergroup\verb@balance@group
703 <latexrelease>  \lowercase{\let~\verb@egroup}}%
704 <latexrelease>
705 <latexrelease>\EndIncludeInRelease
706 <*2ekernel>

```

(End of definition for \@sverb and \@sverb.)

\@makeother

```
707 \def\@makeother#1{\catcode'#112\relax}
```

(End of definition for \@makeother.)

`\verb@balance@group`

```
708 \let\verb@balance@group\@empty
(End of definition for \verb@balance@group.)
```

`\verb@egroup`

```
709 \def\verb@egroup{\global\let\verb@balance@group\@empty\egroup}
(End of definition for \verb@egroup.)
```

`\verb@eol@error`

```
710 \begingroup
711 \obeylines%
712 \gdef\verb@eol@error{\obeylines%
713 \def~M{\verb@egroup\@latex@error{%
714 \noexpand\verb ended by end of line}\@ehc}}%
715 \endgroup
(End of definition for \verb@eol@error.)
```

`\verb` Typesetting a small piece verbatim.

```
716 </2ekernel>
717 <*2ekernel | latexrelease>
718 <latexrelease>\IncludeInRelease{2017-04-15}{\verb}%
719 <latexrelease> {Disable hyphenation in verb}%
720 \def\verb{\relax\ifmmode\hbox\else\leavevmode\null\fi
721 \bgroup
722 \verb@eol@error \let\do\@makeoother \dospecials
723 \verbatim@font\@noligs
```

Set `\language` here to suppress hyphenation. Done this way rather than setting `\hyphenchar` as that is a global setting.

```
724 \language\l@nohyphenation
725 \@ifstar\@sverb\@verb}
726 </2ekernel | latexrelease>
727 <latexrelease>\EndIncludeInRelease
728 <latexrelease>\IncludeInRelease{0000-00-00}{\verb}%
729 <latexrelease> {Disable hyphenation in verb}%
730 <latexrelease>\def\verb{\relax\ifmmode\hbox\else\leavevmode\null\fi
731 <latexrelease> \bgroup
732 <latexrelease> \verb@eol@error \let\do\@makeoother \dospecials
733 <latexrelease> \verbatim@font\@noligs
734 <latexrelease> \@ifstar\@sverb\@verb}
735 <latexrelease>\EndIncludeInRelease
736 <*2ekernel>
```

(End of definition for `\verb`.)

`\@verb`

```
737 \def\@verb{\@vobeyspaces \frenchspacing \@sverb}
(End of definition for \@verb.)
```

`\verbatim@nolig@list`

```
738 \def\verbatim@nolig@list{\do\‘\do\<\do\>\do\,\do\’\do\~}
```

(End of definition for \verbatim@nolig@list.)

`\do@noligs`

```
739 \def\do@noligs#1{%  
740   \catcode'#1\active  
741   \begingroup  
742     \lccode'\~'#1\relax  
743     \lowercase{\endgroup\def~{\leavevmode\kern\z@\char'#1}}
```

(End of definition for \do@noligs.)

`\@noligs` To stay compatible with packages that use `\@noligs` we keep it.

```
744 \def\@noligs{\let\do\do@noligs \verbatim@nolig@list}
```

(End of definition for \@noligs.)

```
745 </2kernel>
```

File 38

ltmath.dtx

1 Math setup

This file contains a lot of the original plain T_EX code, as well as the L^AT_EX environments for math. It still needs sorting out.

```
1 (*2ekernel)
2 \message{math definitions,}
```

1.1 Math commands based on plain T_EX

1.1.1 The log-like functions

`\log` The standard operators:

```
3 \DeclareRobustCommand\log{\mathop{\operator@font log}\nolimits}
4 \DeclareRobustCommand\lg{\mathop{\operator@font lg}\nolimits}
5 \DeclareRobustCommand\ln{\mathop{\operator@font ln}\nolimits}
6 \DeclareRobustCommand\lim{\mathop{\operator@font lim}}
7 \DeclareRobustCommand\limsup{\mathop{\operator@font lim\,sup}}
8 \DeclareRobustCommand\liminf{\mathop{\operator@font lim\,inf}}
9 \DeclareRobustCommand\sin{\mathop{\operator@font sin}\nolimits}
10 \DeclareRobustCommand\arcsin{\mathop{\operator@font arcsin}\nolimits}
11 \DeclareRobustCommand\sinh{\mathop{\operator@font sinh}\nolimits}
12 \DeclareRobustCommand\cos{\mathop{\operator@font cos}\nolimits}
13 \DeclareRobustCommand\arccos{\mathop{\operator@font arccos}\nolimits}
14 \DeclareRobustCommand\cosh{\mathop{\operator@font cosh}\nolimits}
15 \DeclareRobustCommand\tan{\mathop{\operator@font tan}\nolimits}
16 \DeclareRobustCommand\arctan{\mathop{\operator@font arctan}\nolimits}
17 \DeclareRobustCommand\tanh{\mathop{\operator@font tanh}\nolimits}
18 \DeclareRobustCommand\cot{\mathop{\operator@font cot}\nolimits}
19 \DeclareRobustCommand\coth{\mathop{\operator@font coth}\nolimits}
20 \DeclareRobustCommand\sec{\mathop{\operator@font sec}\nolimits}
21 \DeclareRobustCommand\csc{\mathop{\operator@font csc}\nolimits}
22 \DeclareRobustCommand\max{\mathop{\operator@font max}}
23 \DeclareRobustCommand\min{\mathop{\operator@font min}}
24 \DeclareRobustCommand\sup{\mathop{\operator@font sup}}
25 \DeclareRobustCommand\inf{\mathop{\operator@font inf}}
26 \DeclareRobustCommand\arg{\mathop{\operator@font arg}\nolimits}
27 \DeclareRobustCommand\ker{\mathop{\operator@font ker}\nolimits}
28 \DeclareRobustCommand\dim{\mathop{\operator@font dim}\nolimits}
29 \DeclareRobustCommand\hom{\mathop{\operator@font hom}\nolimits}
30 \DeclareRobustCommand\det{\mathop{\operator@font det}}
31 \DeclareRobustCommand\exp{\mathop{\operator@font exp}\nolimits}
32 \DeclareRobustCommand\Pr{\mathop{\operator@font Pr}}
33 \DeclareRobustCommand\gcd{\mathop{\operator@font gcd}}
34 \DeclareRobustCommand\deg{\mathop{\operator@font deg}\nolimits}
```

(End of definition for \log.)

`\bmod` And some operators have to be done by hand:

```

35 \DeclareRobustCommand\bmod{%
36   \nonscript\mskip-\medmuskip\mkern5mu%
37   \mathbin{\operator@font mod}\penalty900\mkern5mu%
38   \nonscript\mskip-\medmuskip}

```

(End of definition for \bmod.)

\pmod

```

39 \DeclareRobustCommand\pmod[1]{%
40   \allowbreak\mkern18mu({\operator@font mod}\,\,\#1)}

```

(End of definition for \pmod.)

1.1.2 Biggggg

\big Variants on **\big** and friends for use with delimiters:

```

41 \DeclareRobustCommand\bigl{\mathopen\big}
42 \DeclareRobustCommand\bigm{\mathrel\big}
43 \DeclareRobustCommand\bigg{\mathclose\big}
44 \DeclareRobustCommand\Bigl{\mathopen\Big}
45 \DeclareRobustCommand\Bigm{\mathrel\Big}
46 \DeclareRobustCommand\Bigg{\mathclose\Big}
47 \DeclareRobustCommand\biggl{\mathopen\bigg}
48 \DeclareRobustCommand\biggm{\mathrel\bigg}
49 \DeclareRobustCommand\biggr{\mathclose\bigg}
50 \DeclareRobustCommand\Biggl{\mathopen\Bigg}
51 \DeclareRobustCommand\Biggm{\mathrel\Bigg}
52 \DeclareRobustCommand\Biggr{\mathclose\Bigg}

```

(End of definition for \big.)

1.1.3 The UNSORTED Rest

The other math commands are lifted from plain T_EX.

\jot

```

53 \newdimen\jot
54 \jot=3pt

```

(End of definition for \jot.)

\interdisplaylinepenalty

```

55 \newcount\interdisplaylinepenalty
56 \interdisplaylinepenalty=100

```

(End of definition for \interdisplaylinepenalty.)

\choose

```

57 \def\choose{\atopwithdelims()}

```

(End of definition for \choose.)

\brack

```

58 \def\brack{\atopwithdelims[]}

```

(End of definition for \brack.)

```

\brace
59 \def\brace{\atopwithdelims\{\}}
(End of definition for \brace.)

\mathpalette
60 \def\mathpalette#1#2{%
61   \mathchoice
62     {#1\displaystyle{#2}}%
63     {#1\textstyle{#2}}%
64     {#1\scriptstyle{#2}}%
65     {#1\scriptscriptstyle{#2}}
(End of definition for \mathpalette.)

\root
\rootbox
\root 66 \newbox\rootbox
\root 67 \def\root#1\of{%
\root 68   \setbox\rootbox\hbox{\m@th\scriptscriptstyle{#1}}%
\root 69   \mathpalette\root}

70 \def\root#1#2{%
71   \setbox\z@\hbox{\m@th#1\sqrt{#2}}%
72   \dimen@ht\z@ \advance\dimen@-dp\z@
73   \mkern5mu\raise.6\dimen@\copy\rootbox
74   \mkern-10mu\box\z@}
(End of definition for \root, \rootbox, and \root.)

\phantom
\hphantom 75 \newif\ifv@
\vphantom 76 \newif\ifh@

77 </2ekernel>
78 <*2ekernel | latexrelease>
79 <latexrelease>\IncludeInRelease{2019/10/01}%
80 <latexrelease>           {\vphantom}{Make commands robust}%
81 \DeclareRobustCommand\vphantom{\v@true\h@false\ph@nt}
82 \DeclareRobustCommand\hphantom{\v@false\h@true\ph@nt}
83 \DeclareRobustCommand\phantom{\v@true\h@true\ph@nt}

84 \DeclareRobustCommand\mathstrut{\vphantom{}}

\mathstrut 85 </2ekernel | latexrelease>
86 <latexrelease>\EndIncludeInRelease
87 <latexrelease>\IncludeInRelease{0000/00/00}%
88 <latexrelease>           {\vphantom}{Make commands robust}%
89 <latexrelease>
90 <latexrelease>\kernel@make@fragile\vphantom
91 <latexrelease>\kernel@make@fragile\hphantom
92 <latexrelease>\kernel@make@fragile\phantom
93 <latexrelease>\kernel@make@fragile\mathstrut
94 <latexrelease>
95 <latexrelease>\EndIncludeInRelease
96 <*2ekernel>

```



```

97 \def\ph@nt{%
98   \ifmmode
99     \expandafter\mathpalette\expandafter\mathph@nt
100   \else
101     \expandafter\makeph@nt
102   \fi}
103 \def\makeph@nt#1{%
104   \setbox\z@\hbox{\color@begingroup#1\color@endgroup}\finph@nt}
105 \def\mathph@nt#1#2{%
106   \setbox\z@\hbox{\$m@th#1{#2}$}\finph@nt}
107 \</2ekernel)
108 \<*2ekernel | latexrelease)
109 \<latexrelease)\IncludeInRelease{2018/12/01}%
110 \<latexrelease)      {\finph@nt}{Start LR-mode}%
111 \def\finph@nt{%
112   \setbox\tw@\null
113   \ifv@ \ht\tw@\ht\z@ \dp\tw@\dp\z@\fi
114   \ifh@ \wd\tw@\wd\z@\fi
115   \leavevmode@ifvmode\box\tw@}
116 \</2ekernel | latexrelease)
117 \<latexrelease)\EndIncludeInRelease
118 \<latexrelease)\IncludeInRelease{0000/00/00}%
119 \<latexrelease)      {\finph@nt}{Start LR-mode}%
120 \<latexrelease)\def\finph@nt{%
121 \<latexrelease)  \setbox\tw@\null
122 \<latexrelease)  \ifv@ \ht\tw@\ht\z@ \dp\tw@\dp\z@\fi
123 \<latexrelease)  \ifh@ \wd\tw@\wd\z@\fi \box\tw@}
124 \<latexrelease)\EndIncludeInRelease
125 \<*2ekernel)

```

(End of definition for \phantom and others.)

\smash

```

126 \DeclareRobustCommand\smash{%
127   \relax % \relax, in case this comes first in \halign
128   \ifmmode
129     \expandafter\mathpalette\expandafter\mathsm@sh
130   \else
131     \expandafter\makesm@sh
132   \fi}
133 \def\makesm@sh#1{%
134   \setbox\z@\hbox{\color@begingroup#1\color@endgroup}\finsm@sh}
135 \</2ekernel)
136 \<*2ekernel | latexrelease)
137 \<latexrelease)\IncludeInRelease{2022/11/01}%
138 \<latexrelease)      {\mathsm@sh}{Guard against reboxing}%
139 \def\mathsm@sh#1#2{%
140   \setbox\z@\hbox{\$m@th#1{#2}$}%

```

The empty brace groups in front of the smashed box (which is placed by \finsm@sh) ensures that a \smash in math is not just producing a single box with its dimensions altered, but a box plus this second ord atom. The reason is that T_EX sometimes reboxes

a box if its the only thing in a place like the denominator of a fraction. This would then undo the smashing and the additional ord atom prevents that. Two ord atoms in a row do not alter the horizontal spacing in a formula so this is otherwise transparent.

```

141   {} \finsm@sh}
142 </2ekernel | latexrelease>
143 <latexrelease> \EndIncludeInRelease
144 <latexrelease> \IncludeInRelease{0000/00/00}%
145 <latexrelease>           {\mathsm@sh}{Guard against reboxing}%
146 <latexrelease> \def \mathsm@sh#1#2{%
147 <latexrelease>   \setbox\z@\hbox{\$ \m@th#1{#2}$} \finsm@sh}
148 <latexrelease> \EndIncludeInRelease
149 <*2ekernel>

```

```

150 </2ekernel>
151 <*2ekernel | latexrelease>
152 <latexrelease> \IncludeInRelease{2018/12/01}%
153 <latexrelease>           {\finsm@sh}{Start LR-mode}%
154 \def \finsm@sh{\ht\z@\z@ \dp\z@\z@ \leavevmode@ifvmode\box\z@}
155 </2ekernel | latexrelease>
156 <latexrelease> \EndIncludeInRelease
157 <latexrelease> \IncludeInRelease{0000/00/00}%
158 <latexrelease>           {\finsm@sh}{Start LR-mode}%
159 <latexrelease> \def \finsm@sh{\ht\z@\z@ \dp\z@\z@ \box\z@}
160 <latexrelease> \EndIncludeInRelease
161 <*2ekernel>

```

(End of definition for \smash.)

`\buildrel`

```

162 \def \buildrel#1\over#2{\mathrel{\mathop{\kern\z@#2}\limits^{#1}}}

```

(End of definition for \buildrel.)

```

163 </2ekernel>
164 <*2ekernel | latexrelease>
165 <latexrelease> \IncludeInRelease{2019/10/01}%
166 <latexrelease>           {\cases}{Make commands robust}%

```

`\cases`

```

167 \DeclareRobustCommand*\cases[1]{\left\{\!,\vcenter{\normalbaselines\m@th
168   \ialign{\##\hfil$&\quad\##}\hfil\crr#1\crr}\right.}

```

(End of definition for \cases.)

`\matrix`

```

169 \DeclareRobustCommand*\matrix[1]{\null\!,\vcenter{\normalbaselines\m@th
170   \ialign{\hfil$##$\hfil&&\quad\hfil$##$\hfil\crr
171   \mathstrut\crr\noalign{\kern-\baselineskip}
172   #1\crr\mathstrut\crr\noalign{\kern-\baselineskip}}}\!,}

```

(End of definition for \matrix.)

`\pmatrix`

```

173 \DeclareRobustCommand*\pmatrix[1]{\left(\matrix{#1}\right)}

```

(End of definition for `\pmatrix`.)

```
174 </2ekernel | latexrelease>
175 <latexrelease>\EndIncludeInRelease
176 <latexrelease>\IncludeInRelease{0000/00/00}%
177 <latexrelease>          {\cases}{Make commands robust}%
178 <latexrelease>
179 <latexrelease>\kernel@make@fragile\cases
180 <latexrelease>\kernel@make@fragile\matrix
181 <latexrelease>\kernel@make@fragile\pmatrix
182 <latexrelease>
183 <latexrelease>\EndIncludeInRelease
184 <*2ekernel>
```

`\bordermatrix`

```
185 \def\bordermatrix#1{\begingroup \m@th
186   \@tempdima 8.75\p@
187   \setbox\z@\vbox{%
188     \def\crr{\crr\noalign{\kern2\p@\global\let\crr\endline}}%
189     \ialign{##$\hfil\kern2\p@\kern\@tempdima&\thinspace\hfil$##$\hfil
190       &&\quad\hfil$##$\hfil\crr
191       \omit\strut\hfil\crr\noalign{\kern-\baselineskip}}%
192     #1\crr\omit\strut\crr}%
193   \setbox\tw@\vbox{\unvcopy\z@\global\setbox\@ne\lastbox}%
194   \setbox\th@\hbox{\unhbox\@ne\unskip\global\setbox\@ne\lastbox}%
195   \setbox\tw@\hbox{${\kern\wd\@ne\kern-\@tempdima\left(\kern-\wd\@ne
196     \global\setbox\@ne\vbox{\box\@ne\kern2\p@}}%
197     \vcenter{\kern-\ht\@ne\unvbox\z@\kern-\baselineskip}\,,\right)}$}%
198   \null\; \vbox{\kern\ht\@ne\box\tw@}\endgroup}
```

(End of definition for `\bordermatrix`.)

`\openup`

```
199 \protected\def\openup{\afterassignment\@openup\dimen@}
200 \def\@openup{\advance\lineskip\dimen@
201   \advance\baselineskip\dimen@
202   \advance\lineskiplimit\dimen@}
```

(End of definition for `\openup`.)

`\displaylines`

```
203 \newif\ifdt@p
204 \def\disply{\global\dt@ptrue\openup\jot\m@th
205   \everycr{\noalign{\ifdt@p \global\dt@pfalse \ifdim\prevdepth>-1000\p@
206     \vskip-\lineskiplimit \vskip\normallineskiplimit \fi
207     \else \penalty\interdisplaylinepenalty \fi}}
208 \def\@lign{\tabskip\z@skip\everycr{}} % restore inside \disply
209 \def\displaylines#1{\disply \tabskip\z@skip
210   \halign{\hb@xt@\displaywidth{${\@lign\hfil\displaystyle##\hfil}$}\crr
211     #1\crr}}
```

(End of definition for `\displaylines`.)

```

\sp
\sb 212 \let\sp=~
    213 \let\sb=_

(End of definition for \sp and \sb.)

\tmspace Originally LATEX only provided a small set of spacing commands for use in text and math,
\ some of the commands like \; were only supported in math mode. amsmath normalized
\thinspace and provided all of them in text and math. This code has now been moved to the kernel
\! so that it is generally available.
\negthinspace 214 </2kernel>
\: 215 <*2kernel | latexrelease>
\medspace 216 <latexrelease>\IncludeInRelease{2020/10/01}%
\negmedspace 217 <latexrelease> \tmspace}{amsmath spacing commands}%
\;
\thickspace \tmspace is really meant to be an internal command so it doesn't necessarily has to be
\negthickspace robust but it was robust in amsmath so we leave it like that.
218 \DeclareRobustCommand\tmspace[3]{%
219 \ifmmode\mskip#1#2\else\leavevmode@ifvmode\kern#1#3\fi\relax}

In amsmath the text kern is .1667em. For compatibility reasons we keep the longer one.
220 \DeclareRobustCommand\,\{\tmspace+\thinmuskip{.16667em}}
221 \let\thinspace\,
222 \DeclareRobustCommand\!{\tmspace-\thinmuskip{.16667em}}
223 \let\negthinspace\!
224 \DeclareRobustCommand\:{\tmspace+\medmuskip{.2222em}}
225 \let\medspace\{

LATEX has a second name for this in its manual:
226 \let\>=\:
227 \DeclareRobustCommand\negmedspace{\tmspace-\medmuskip{.2222em}}
228 \DeclareRobustCommand\;{\tmspace+\thickmuskip{.2777em}}
229 \let\thickspace\;
230 \DeclareRobustCommand\negthickspace{\tmspace-\thickmuskip{.2777em}}
231 </2kernel | latexrelease>
232 <latexrelease>\EndIncludeInRelease
233 <latexrelease>\IncludeInRelease{0000/00/00}%
234 <latexrelease> \tmspace}{amsmath spacing commands}%
235 <latexrelease>
236 <latexrelease>\let\tmspace\@undefined
237 <latexrelease>\DeclareRobustCommand{\,}{%
238 <latexrelease> \relax\ifmmode\mskip\thinmuskip\else\thinspace\fi}
239 <latexrelease>\DeclareRobustCommand\thinspace{\leavevmode@ifvmode\kern .16667em }
240 <latexrelease>\DeclareRobustCommand\negthinspace{\leavevmode@ifvmode\kern-.16667em }
241 <latexrelease>\def\>{\mskip\medmuskip}
242 <latexrelease>\let\:=\>
243 <latexrelease>\def\;{\mskip\thickmuskip}
244 <latexrelease>\def\!{\mskip-\thinmuskip}
245 <latexrelease>

```

```

246 <latexrelease>\let\negmedspace\@undefined
247 <latexrelease>\let\negthickspace\@undefined
248 <latexrelease>
249 <latexrelease>\EndIncludeInRelease
250 <*2ekernel>

```

(End of definition for \tmspace and others.)

`*`

```

251 \DeclareRobustCommand\*\discretionary{\thinspace\the\textfont2\char2}{-}{-}

```

*(End of definition for *.)*

`\:` Nickname for the medium space since `\>` is not available inside `tabbing`.

```

252 %\let\:=\>

```

(End of definition for \:.)

`\active@math@prime` This is the definition of the active math prime.

```

253 \def\active@math@prime{\bgroup\prim@s}

```

(End of definition for \active@math@prime.)

`\prime@s`

```

254 {\catcode'\='=active \global\let'\active@math@prime}

```

```

255 \def\prim@s{%

```

```

256   \prime\futurelet\@let@token\pr@m@s}

```

```

257 \def\pr@m@s{%

```

```

258   \ifx'\@let@token

```

```

259     \expandafter\pr@@@s

```

```

260   \else

```

```

261     \ifx'\@let@token

```

```

262       \expandafter\expandafter\expandafter\pr@@@t

```

```

263     \else

```

```

264       \egroup

```

```

265     \fi

```

```

266   \fi}

```

```

267 \def\pr@@@s#1{\prim@s}

```

```

268 \def\pr@@@t#1#2{#2\egroup}

```

(End of definition for \prime@s.)

```

269 {\catcode'\_='=active \gdef_{\_}} % _ in math is

```

```

270                                     % either subscript or \_

```

1.2 Math Environments

`\(` Produces \dots with checks that `\(` isn't used in math mode, and that `\)` is only used in math mode begun with `\(`.

```

271 </2ekernel>
272 <latexrelease>\IncludeInRelease{2015/01/01}{\{ }\{Make \(\ robust}\%
273 <*2ekernel | latexrelease>
274 \DeclareRobustCommand\({%
275   \relax\ifmmode\@badmath\else$\fi}%
276 \DeclareRobustCommand\)%{%
277   \relax\ifmmode\ifinner$\else\@badmath\fi\else \@badmath\fi}%
278 </2ekernel | latexrelease>
279 <latexrelease>\EndIncludeInRelease
280 <latexrelease>\IncludeInRelease{0000/00/00}{\{ }\{Make \(\ robust}\%
281 <latexrelease>\def\({%
282 <latexrelease>  \relax\ifmmode\@badmath\else$\fi}%
283 <latexrelease>\expandafter\let\csname\string( \endcsname\@undefined
284 <latexrelease>\def\)%{%
285 <latexrelease>  \relax\ifmmode\ifinner$\else\@badmath\fi\else \@badmath\fi}%
286 <latexrelease>\expandafter\let\csname\string) \endcsname\@undefined
287 <latexrelease>\EndIncludeInRelease
288 <*2ekernel>

```

(End of definition for `\(` and `\)`.)

`\[` Produces $$$\dots$$$ with checks that `\[` isn't used in math mode, and that `\]` is only used in display math mode (though there is no real test that this display math started with `\[` and not with $$$$).

```

289 </2ekernel>
290 <latexrelease>\IncludeInRelease{2015/01/01}{\{ }\{Make \[ robust}\%
291 <*2ekernel | latexrelease>
292 \DeclareRobustCommand\[{%
293   \relax\ifmmode
294     \@badmath
295   \else
296     \ifvmode
297       \nointerlineskip
298       \makebox[.6\linewidth]{}%
299     \fi
300     $$$%$$ BRACE MATCH HACK
301   \fi
302 }%
303 \DeclareRobustCommand\]%{%
304   \relax\ifmmode
305     \ifinner
306     \@badmath
307   \else
308     $$$%$$ BRACE MATCH HACK
309   \fi
310   \else
311     \@badmath
312   \fi
313   \ignorespaces
314 }%

```

```

315 </2ekernel | latexrelease>
316 <latexrelease>\EndIncludeInRelease
317 <latexrelease>\IncludeInRelease{0000/00/00}{\[]}{Make \[ robust]}%
318 <latexrelease>\def\[%
319 <latexrelease> \relax\ifmmode
320 <latexrelease> \@badmath
321 <latexrelease> \else
322 <latexrelease> \ifvmode
323 <latexrelease> \nointerlineskip
324 <latexrelease> \makebox[.6\linewidth]{}%
325 <latexrelease> \fi
326 <latexrelease> $$$%$$ BRACE MATCH HACK
327 <latexrelease> \fi
328 <latexrelease>}%
329 <latexrelease>\expandafter\let\csname\string[ \endcsname\@undefined
330 <latexrelease>\def\[%
331 <latexrelease> \relax\ifmmode
332 <latexrelease> \ifinner
333 <latexrelease> \@badmath
334 <latexrelease> \else
335 <latexrelease> $$$%$$ BRACE MATCH HACK
336 <latexrelease> \fi
337 <latexrelease> \else
338 <latexrelease> \@badmath
339 <latexrelease> \fi
340 <latexrelease> \ignorespaces
341 <latexrelease>}%
342 <latexrelease>\expandafter\let\csname\string[ \endcsname\@undefined
343 <latexrelease>\EndIncludeInRelease
344 <*2ekernel)

```

(End of definition for \[and \].)

math (*env.*) Disguises for (\dots) and $[\dots]$.

```

displaymath (env.) 345 \let\math=\(
346 \let\endmath=\)
347 \def\displaymath{\[]
348 \def\enddisplaymath{\}\@ignoretrue}

```

equation (*env.*) Numbered equations, using the counter `\c@equation`. *Note:* The document style must define `\theequation` etc., and do the appropriate `\@addtoreset`. It should also redefine `\@eqnnum` if another format for the equation number is desired other than the standard (...), or to move the equation numbers to the flushleft. (See comment on the `\def` of `\@eqnnum`.)

```

349 \@definecounter{equation}
350 \def\equation{$$\refstepcounter{equation}}
351 \def\endequation{\eqno \hbox{\@eqnnum}$$\@ignoretrue}

```

(End of definition for `\c@equation`.)

\@eqnnum Produces the equation number for equation and eqnarray environments. The following definition is for flushright numbers; for flushleft numbers, see `leqno.clo`. The equation number is set in black roman type even if an eqnarray environment appears in an italic environment.

```

352 \def\@eqnnum{\normalfont \normalcolor (\theequation)}

```

(End of definition for `\@eqnnum`.)

`\stackrel` A disguise for plain T_EX's `\stackrel`.
353 `\DeclareRobustCommand\stackrel[2]{\mathrel{\mathop{#2}\limits^{#1}}}`

(End of definition for `\stackrel`.)

`\frac` A disguise for plain T_EX's `\over`.
354 `\DeclareRobustCommand\frac[2]{\begingroup#1\endgroup\over#2}`

(End of definition for `\frac`.)

`\sqrt` Add an optional argument to plain's `\sqrt` to give the *n*th root of an expression $\sqrt[n]{e}$.
`\@sqrt` 355 `\DeclareRobustCommand\sqrt{\@ifnextchar[\@sqrt\sqrtsign}`
356 `\def\@sqrt[#1]{\root #1\of}`

(End of definition for `\sqrt` and `\@sqrt`.)

`\eqnarray` (comment) Here's the `\eqnarray` environment: Default is for left-hand side of equations to be flushright. To make them flushleft, `\let\@eqnset = \hfil`.

`\@eqnset`
`\@eqnsw` 357 `\newcount\@eqcnt`
`\@eqnset` 358 `\newcount\@eqpen`
359 `\newif\if@eqnsw\@eqnswtrue`
360 `\newskip\@centering`
361 `\@centering = 0pt plus 1000pt`

To get a proper `\@currentlabel` we have to redefine it for the whole display. Note that we can't use `\refstepcounter` as this results in `\@currentlabel` getting restored at the wrong and thus always writing the first label to the `.aux` file.

```
362 \def\eqnarray{%
363   \stepcounter{equation}%
364   \def\@currentlabel{\p@equation\theequation}%
365   \def\@currentcounter{equation}%
366   \global\@eqnswtrue
367   \m@th
368   \global\@eqcnt\z@
369   \tabskip\@centering
370   \let\\\@eqncr
371   $$\everycr{\halign to\displaywidth\bgroup
372     \hskip\@centering$\displaystyle\tabskip\z@skip{##}$\@eqnset
373     &\global\@eqcnt\@ne\hskip \tw@\arraycolsep \hfil{##}$\hfil
374     &\global\@eqcnt\tw@ \hskip \tw@\arraycolsep
375     $\displaystyle{##}$\hfil\tabskip\@centering
376     &\global\@eqcnt\thr@@ \hb@xt@\z@\bgroup\hss##\egroup
377     \tabskip\z@skip
378   \cr
379 }
380 \def\endeqnarray{%
381   \@eqncr
382   \egroup
383   \global\advance\c@equation\m@ne
384   $$\@ignoretrue
385 }
386 \let\@eqnset=\relax
```


(End of definition for \@eqcnt and others.)

\nonumber Switches off equation numbering.

```
387 \def\nonumber{\global\@eqnswfalse}
```

(End of definition for \nonumber.)

\@eqnocr

\@xeqnocr

\@yeqnocr

```
388 \protected\def\@eqnocr{%
389   {\ifnum0='}\fi
390   \@ifstar{%
391     \global\@eqpen\@M\@yeqnocr
392   }{%
393     \global\@eqpen\interdisplaylinepenalty \@yeqnocr
394   }%
395 }
396 \def\@yeqnocr{\@testopt\@xeqnocr\z@skip}
397 \</2kernel)
398 \<*2kernel | latexrelease)
399 \<latexrelease)\IncludeInRelease{2020/10/01}%
400 \<latexrelease)          {\@xeqnocr}{eqnarray support calc syntax}%
401 \def\@xeqnocr[#1]{%
402   \ifnum0='{ \fi}%
403   \@@eqnocr
404   \noalign{\penalty\@eqpen\vskip\jot\@vspace@calcify{#1}}%
405 }
406 \</2kernel | latexrelease)
407 \<latexrelease)\EndIncludeInRelease
408 \<latexrelease)\IncludeInRelease{0000/00/00}%
409 \<latexrelease)          {\@xeqnocr}{eqnarray support calc syntax}%
410 \<latexrelease)
411 \<latexrelease)\def\@xeqnocr[#1]{%
412 \<latexrelease)   \ifnum0='{ \fi}%
413 \<latexrelease)   \@@eqnocr
414 \<latexrelease)   \noalign{\penalty\@eqpen\vskip\jot\vskip #1\relax}%
415 \<latexrelease)}
416 \<latexrelease)\EndIncludeInRelease
417 \<*2kernel)
```

(End of definition for \@eqnocr, \@xeqnocr, and \@yeqnocr.)

\@@eqnocr

```
418 \def\@@eqnocr{\let\reserved@a\relax
419   \ifcase\@eqcnt \def\reserved@a{& & } \or \def\reserved@a{& & }%
420   \or \def\reserved@a{&} \else
421     \let\reserved@a\empty
422     \@latex@error{Too many columns in eqnarray environment}\@ehc\fi
423   \reserved@a \if@eqnsw\@eqnnum\stepcounter{equation}\fi
424   \global\@eqnswtrue\global\@eqcnt\z@\cr}
```

(End of definition for \@@eqnocr.)

`\use{qnrcr}` Here's the `eqnarray*` environment:

```
425 \let\@seqnocr=\@eqnocr
426 \@namedef{eqnarray*}{\protected\def\@eqnocr{\nonumber\@seqnocr}\eqnarray}
427 \@namedef{endeqnarray*}{\nonumber\endeqnarray}
(End of definition for \@seqnocr.)
```

`\lefteqn` `\lefteqn{FORMULA}` typesets FORMULA in display math style flushleft in a box of width zero.

```
428 \def\lefteqn#1{\rlap{$\displaystyle #1$}}
(End of definition for \lefteqn.)
```

`\ensuremath` In math mode, `\ensuremath{text}` is equivalent to `text`; in LR or paragraph mode, it is equivalent to `$text$`. `\relax` is not needed in front of the `\ifmmode` as `\protect` will be `\let` to `\relax`. This version (due to Donald Arseneau) avoids duplicating its argument in the 'then' and 'else' part of the `\ifmath` which is necessary in nested 'tabular' like environments. See `amslatex/2104`.

```
429 \DeclareRobustCommand{\ensuremath}{%
430   \ifmmode
431     \expandafter\@firstofone
432   \else
433     \expandafter\@ensuredmath
434   \fi}
```

(End of definition for `\ensuremath`.)

`\@ensuredmath` The `\relax` stops `\ensuremath{}` starting display math.

```
435 \long\def\@ensuredmath#1{${\relax#1$}}
(End of definition for \@ensuredmath.)
```

Lua \TeX contains new math primitives to place expression over or under horizontally extensible glyphs. Before Lua \TeX 1.14 these did not work correctly with the `\mathstyle` primitive and sometimes did not use cramped style in consistent ways. For newer version, we opt into the corrected behavior.

```
436 \ifx\mathdefaultsmode\@undefined\else
437   \mathdefaultsmode=1
438 \fi
```

`\eqno` Ensure the (deprecated) `$$..\eqno 1 $$` ignores spaces.

```
\leqno
439 </2kernel>
440 < *2kernel | latexrelease >
441 < latexrelease > \IncludeInRelease{2023/06/01}%
442 < latexrelease >           {\eqno}{add ignorespaces to eqno}%
443 \let\@kernel@eqno\eqno
444 \let\@kernel@leqno\leqno
445 \protected\def\eqno{\@kernel@eqno\aftergroup\ignorespaces}
446 \protected\def\leqno{\@kernel@leqno\aftergroup\ignorespaces}
447 </2kernel | latexrelease >
448 < latexrelease > \EndIncludeInRelease
449 < latexrelease > \IncludeInRelease{0000/00/00}%
450 < latexrelease >           {\eqno}{add ignorespaces to eqno}%
451 < latexrelease > \let\eqno\@kernel@eqno
452 < latexrelease > \let\leqno\@kernel@leqno
453 < latexrelease > \EndIncludeInRelease
```

(End of definition for `\eqno` and `\leqno`.)

1.3 External options to the standard document classes

1.3.1 Left equation numbering

`\@eqnnum` To put the equation number on the left side of an equation we have to use a little trick. The number is shifted `\displaywidth` to the left inside a box of (approximately) zero width. This fails when the equation is too wide, the equation number than may overprint the equation itself.

```
454 \*leqno
455 \renewcommand\@eqnnum{\hb@xt@.01\p@{}}%
456 \rlap{\normalfont\normalcolor
457 \hskip -\displaywidth(\theequation)}}
458 \leqno
```

(End of definition for `\@eqnnum`.)

1.3.2 Flush left equations

To get the displayed math environments to print the contents flush left (with an indentation) we have to redefine all of L^AT_EX 2_ε's displayed math environments.

`\mathindent` The amount of indentation of the equations is stored in a register.

```
459 \*fleqn
460 \newskip\mathindent
461 \AtEndOfClass{\mathindent\leftmargin}
```

The setting of `\mathindent` has to be deferred until the class file has been processed, because `\leftmargin` is still 0pt wide at the moment `fleqn.clo` is read in.

(End of definition for `\mathindent`.)

`\[` Begin display math;

```
462 \IncludeInRelease{2015/01/01}{\[]}{Make \[ robust}%
463 \DeclareRobustCommand\[\relax
464 \ifmode\@badmath
465 \else
466 \begin{trivlist}%
467 \@beginparpenalty\predisplaypenalty
468 \@endparpenalty\postdisplaypenalty
469 \item[]\leavevmode
470 \hb@xt@\linewidth\bgroup $\m@th\displaystyle %$
471 \hskip\mathindent\bgroup
472 \fi}
473 \EndIncludeInRelease
474 \IncludeInRelease{0000/00/00}{\[]}{Make \[ robust}%
475 \renewcommand\[\relax
476 \ifmode\@badmath
477 \else
478 \begin{trivlist}%
479 \@beginparpenalty\predisplaypenalty
480 \@endparpenalty\postdisplaypenalty
481 \item[]\leavevmode
```

```

482             \hb@xt@\linewidth\bgroup $\m@th\displaystyle %$
483             \hskip\mathindent\bgroup
484             \fi}
485 \EndIncludeInRelease

```

(End of definition for \[.)

\] end display math;

```

486 \IncludeInRelease{2015/01/01}{\]}{Make \] robust}%
487 \DeclareRobustCommand\]{\relax
488         \ifmmode
489             \egroup $\hfil% $
490             \egroup
491             \end{trivlist}%
492         \else \@badmath
493         \fi}
494 \EndIncludeInRelease

```

```

495 \IncludeInRelease{0000/00/00}{\]}{Make \] robust}%
496 \renewcommand\]{\relax
497         \ifmmode
498             \egroup $\hfil% $
499             \egroup
500             \end{trivlist}%
501         \else \@badmath
502         \fi}
503 \EndIncludeInRelease

```

(End of definition for \].)

equation (*env.*) The equation environment

```

504 \renewenvironment{equation}%
505     {\@beginparpenalty\predisplaypenalty
506     \@endparpenalty\postdisplaypenalty
507     \refstepcounter{equation}%
508     \trivlist \item[]\leavevmode
509     \hb@xt@\linewidth\bgroup $\m@th% $
510     \displaystyle
511     \hskip\mathindent}%

```

Ensure that there is at least a space between formula and equation number so that they don't bump in each other.

```

512         {${\hskip .3em minus.3em\hfil % $
513         \displaywidth\linewidth\hbox{\@eqnum}%
514         \egroup
515         \endtrivlist}

```

eqnarray (*env.*) The eqnarray environment

```

516 \renewenvironment{eqnarray}{%
517     \stepcounter{equation}%
518     \def\@currentlabel{\p@equation\theequation}%
519     \def\@currentcounter{equation}%
520     \global\@eqnswtrue\m@th
521     \global\@eqcnt\z@
522     \tabskip\mathindent

```

```

523 \let\=\@eqnocr
524 \setlength\abovedisplayskip{\topsep}%
525 \ifvmode
526   \addtolength\abovedisplayskip{\partopsep}%
527 \fi

```

When the documentclass uses a non-zero `\parskip` setting the `\topsep` might have a negative value to compensate for that. Therefore we add `\parskip` to `\abovedisplayskip`.

```

528 \addtolength\abovedisplayskip{\parskip}%
529 \setlength\belowdisplayskip{\abovedisplayskip}%
530 \setlength\belowdisplayshortskip{\abovedisplayskip}%
531 \setlength\abovedisplayshortskip{\abovedisplayskip}%
532 $$\everycr{}\halign to\linewidth% $$
533 \bgroup
534   \hskip\@centering
535   $\displaystyle\tabskip\z@skip{##}$\@eqnse1&%
536   \global\@eqcnt\@ne \hskip \tw@\arraycolsep \hfil${##}$\hfil&%
537   \global\@eqcnt\tw@ \hskip \tw@\arraycolsep
538   $\displaystyle{##}$\hfil \tabskip\@centering&%
539   \global\@eqcnt\thr@@
540   \hb@xt@\z@\bgroup\hss#\egroup\tabskip\z@skip\cr}%
541   {\@eqnocr
542 \egroup
543 \global\advance\c@equation\m@ne$$$ $$
544 \@ignoretrue
545 }
546 </fleqn>

```

File 39

ltlists.dtx

1 List, and related environments

The generic commands for creating an indented environment – `enumerate`, `itemize`, `quote`, etc – are:

```
\list{<LABEL>}{<COMMANDS>} ... \endlist
```

which can be invoked by the user as the list environment. The LABEL argument specifies item labeling. COMMANDS contains commands for changing the horizontal and vertical spacing parameters.

Each item of the environment is begun by the command `\item[ITEMLABEL]` which produces an item labeled by ITEMLABEL. If the argument is missing, then the LABEL argument of the `\list` command is used as the item label.

The label is formed by putting `\makelabel{<ITEMLABEL>}` in an hbox whose width is either its natural width or else `\labelwidth`, whichever is larger. The `\list` command defines `\makelabel` to have the default definition:

```
\makelabel{<ARG>} == BEGIN \hfil ARG END
```

which, for a label of width less than `\labelwidth`, puts the label flushright, `\labelsep` to the left of the item's text. However, `\makelabel` can be `\let` to another command by the `\list`'s COMMANDS argument.

A `\usecounter{<foo>}` command in the second argument causes the counter *foo* to be initialized to zero, and stepped by every `\item` command without an argument. (`\label` commands within the list refer to this counter.)

When you leave a list environment, returning either to an enclosing list or normal text mode, LaTeX begins a new paragraph if and only if you leave a blank line after the `\end` command. This is accomplished by the `\@endparenv` command.

Blank lines are ignored every other reasonable place—i.e.:

- Between the `\begin{list}` and the first `\item`,
- Between the `\item` and the text of that item.
- Between the end of the last item and the `\end{list}`.

For an environment like quotation, in which items are not labeled, the entire environment is a single item. It is defined by letting `\quotation == \list{}{...}\item\relax`. (Note the `\relax`, there in case the first character in the environment is a '['.) The spacing parameters provide a great deal of flexibility in designing the format, including the ability to let the indentation of the first paragraph be different from that of the subsequent ones.

The trivlist environment is equivalent to a list environment whose second argument sets the following parameter values:

`\leftmargin = 0`: causes no indentation of left margin

`\labelwidth = 0`: see below for precise effect this has.

`\itemindent = 0`: with a null label, makes first paragraph have no indentation. Succeeding paragraphs have `\parindent` indentation. To give first paragraph same indentation, set `\itemindent = \parindent` before the `\item[]`.

Every `\item` in a trivlist environment must have an argument—in many cases, this will be the null argument (`\item[]`). The trivlist environment is mainly used for paragraphing environments, like `verbatim`, in which there is no margin change. It provides the same vertical spacing as the list environment, and works reasonably well when it occurs immediately after an `\item` command in an enclosing list.

1.1 List and Trivlist

The following variables are used inside a list environment:

`\totalleftmargin` The distance that the prevailing left margin is indented from the outermost left margin,

`\linewidth` The width of the current line. Must be initialized to `\hsize`.

`\listdepth` A count for holding current list nesting depth.

`\makelabel` A macro with a single argument, used to generate the label from the argument (given or implied) of the `\item` command. Initialized to `\@mklab` by the `\list` command. This command must produce some stretch—i.e., an `\hfil`.

`\inlabel` A switch that is false except between the time an `\item` is encountered and the time that `TEX` actually enters horizontal mode. Should be tested by commands that can be messed up by the list environment's use of `\everypar`.

`\box\@labels` When `\inlabel = true`, it holds the labels to be put out by `\everypar`.

`\noparitem` A switch set by `\list` when `\inlabel = true`. Handles the case of a `\list` being the first thing in an item.

`\noparlist` A switch set true for a list that begins an item. No `\topsep` space is added before or after `\item`'s such a list.

`\newlist` Set true by `\list`, set false by the first text (by `\everypar`).

`\noitemarg` Set true when executing an `\item` with no explicit argument. Used to save space. To save time, make two separate `\@item` commands.

`\nmbrrlist` Set true by `\usecounter` command, causes list to be numbered.

`\listctr` `\def`'ed by `\usecounter` to name of counter.

`\noskipsec` A switch set true by a sectioning command when it is creating an in-text heading with `\everypar`.

Throughout a list environment, `\hsize` is the width of the current line, measured from the outermost left margin to the outermost right margin. Environments like `tabbing` should use `\linewidth` instead of `\hsize`.

Here are the parameters of a list that can be set by commands in the `\list's` COMMANDS argument. These parameters are all `TEX` skips or dimensions (defined by `\newskip` or `\newdimen`), so the usual `TEX` or `LATEX` commands can be used to set them. The commands will be executed in `vmode` if and only if the `\list` was preceded by a `\par` (or something like an `\end{list}`), so the spacing parameters can be set according to whether the list is inside a paragraph or is its own paragraph.

1.2 Vertical Spacing (skips)

`\topsep`: Space between first item and preceding paragraph.

`\partopsep`: Extra space added to `\topsep` when environment starts a new paragraph (is called in vmode).

`\itemsep`: Space between successive items.

`\parsep`: Space between paragraphs within an item – the `\parskip` for this environment.

1.3 Penalties

`\@beginparpenalty`: put at the beginning of a list

`\@endparpenalty`: put at end of list

`\@itempenalty`: put between items.

1.4 Horizontal Spacing (dimens)

`\leftmargin`: space between left margin of enclosing environment (or of page if top level list) and left margin of this list. Must be nonnegative.

`\rightmargin`: analogous.

`\listparindent`: extra indentation at beginning of every paragraph of a list except the one started by the `\item` command. May be negative! Usually, labeled lists have `\listparindent` equal to zero.

`\itemindent`: extra indentation added right BEFORE an item label.

`\labelwidth`: nominal width of box that contains the label. If the natural width of the label \leq `\labelwidth`, then the label is flushed right inside a box of width `\labelwidth` (with an `\hfil`). Otherwise, a box of the natural width is employed, which causes an indentation of the text on that line.

`\labelsep`: space between end of label box and text of first item.

1.5 Default Values

Defaults for the list environment are set as follows. First, `\rightmargin`, `\listparindent` and `\itemindent` are set to 0pt. Then, one of the commands `\@listi`, `\@listii`, ... , `\@listvi` is called, depending upon the current level of the list. The `\@list ...` commands should be defined by the document style. A convention that the document style should follow is to set `\leftmargin` to `\leftmargini`, ..., `\leftmarginvi` for the appropriate level. Items that aren't changed may be left alone, but everything that could possibly be changed must be reset. *Historical L^AT_EX 2.09 comments (not necessarily accurate any more)*:

```
\list{LABEL}{COMMANDS} ==  
BEGIN  
  if \@listdepth > 5  
    then LaTeX error: 'Too deeply nested'  
    else \@listdepth :=G \@listdepth + 1
```



```

fi
\rightmargin := 0pt
\listparindent := 0pt
\itemindent := 0pt
\eval{@list \romannumeral\the\@listdepth} %% Set default values:
\itemlabel :=L LABEL
\makelabel == \@mklab
@nmbrlist :=L false
COMMANDS

\@trivlist % commands common to \list and \trivlist

\parskip :=L \parsep
\parindent :=L \listparindent
\linewidth :=L \linewidth - \rightmargin - \leftmargin
\totalleftmargin :=L \@totalleftmargin + \leftmargin
\parshape 1 \@totalleftmargin \linewidth
\ignorespaces % gobble space up to \item
END

\endlist == BEGIN \@listdepth :=G \@listdepth -1
\endtrivlist
END

\@trivlist ==
BEGIN
if @newlist = T then \@noitemerr fi
%% This command removed for some forgotten reason.
\@topsepadd :=L \topsep
if @noskipsec then leave vertical mode fi %% Added 11 Jun 85
if vertical mode
then \@topsepadd :=L \@topsepadd + \partopsep
else \unskip \par % remove glue from end of last line
fi
if @inlabel = true
then @noparitem :=L true
@noparlist :=L true
else @noparlist :=L false
\@topsep :=L \@topsepadd
fi
\@topsep :=L \@topsep + \parskip %% Change 4 Sep 85
\leftskip :=L 0pt % Restore paragraphing parameters
\rightskip :=L \@rightskip
\parfillskip :=L 0pt + 1fil

NOTE: \@setpar called on every \list in case \par has been
temporarily munged before the \list command.
\setpar{if @newlist = false then {\@@par} fi}
\newlist :=G T
\@outerparskip :=L \parskip

```

```

END

\trivlist ==
BEGIN
\parsep := \parskip
@nmbrlist := F
\@trivlist
\labelwidth := 0
\leftmargin := 0
\itemindent := \parindent
\@itemlabel :=L "empty" %% added 93/12/13
\makelabel{LABEL} == LABEL
END

\endtrivlist ==
BEGIN
if @inlabel = T then \indent fi
if horizontal mode then \unskip \par fi
if @nparlist = true
else if \lastskip > 0
then \@tempkipa := \lastskip
\vskip - \lastskip
\vskip \@tempkipa -\@outerparskip + \parskip
fi
\@endparenv
fi
END

\@endparenv ==
BEGIN
\addpenalty{@endparpenalty}
\addvspace{@topsepadd}
\endgroup %% ends the \begin command's \begingroup
\par == BEGIN
\@restorepar
\everypar{}
\par
END
\everypar == BEGIN remove \lastbox \everypar{} END
\begingroup %% to match the \end commands \endgroup
END

\item == BEGIN if math mode then WARNING fi
if next char = [
then \@item
else @noitemarg := true
\@item[@itemlabel]
END

\@item[LAB] ==

```

```

BEGIN
if @noparitem = true
then @noparitem := false
      % NOTE: then clause hardly every taken,
      % so made a macro \@donoparitem
\box\@labels :=G \hbox{\hskip -\leftmargin
                    \box\@labels
                    \hskip \leftmargin }
if @minipage = false then
  \@tempskipa := \lastskip
  \vskip -\lastskip
  \vskip \@tempskipa + \@outerparskip - \parskip
fi
else if @inlabel = true
  then \indent \par % previous item empty.
  fi
if hmode then 2 \unskip's
      % To remove any space at end of prev.
      % paragraph that could cause a blank line.
  \par
fi
if @newlist = T
  then if @nobreak = T % Kludge if list follows \section
        then \addvspace{\@outerparskip - \parskip}
        else \addpenalty{\@beginparpenalty}
              \addvspace{\@topsep}
              \addvspace{-\parskip} %% added 4 Sep 85
        fi
  else \addpenalty{\@itempenalty}
        \addvspace{\itemsep}
  fi
  @inlabel :=G true
fi

\everypar{ @minipage :=G F
           @newlist :=G F
           if @inlabel = true
             then @inlabel :=G false
                  \hskip -\parindent
                  \box\@labels
                  \penalty 0
                  %% 3 Oct 85 - allow line break here
                  \box\@labels :=G null
             fi
           \everypar{} }
@nobreak :=G false
if @noitemarg = true
then @noitemarg := false
  if @nmbulist
  then \refstepcounter{\@listctr}

```

```

fi      fi
\@tempboxa :=L \hbox{\makelabel{LAB}}
\box\@labels :=G \@labels \hskip \itemindent
              \hskip - (\labelwidth + \labelsep)
              if \wd \@tempboxa > \labelwidth
                then \box\@tempboxa
                else \hbox to \labelwidth {\unhbox\@tempboxa}
              fi
              \hskip\labelsep

\ignorespaces           %gobble space up to text
END

```

```

\makelabel{LABEL} == ERROR   %% default to catch lonely \item

```

```

\usecounter{CTR} == BEGIN @nmbrlist :=L true
                        \@listctr == CTR
                        \setcounter{CTR}{0}
                        END

```

DEFINE \dimen's and \count
End of historical L^AT_EX 2.09 comments.

```

\topsep
\partopsep 1 (*2kernel)
\itemsep   2 \newskip\topsep
\parsep    3 \newskip\partopsep
\@topsep   4 \newskip\itemsep
\@topsepadd 5 \newskip\parsep
\@outerparskip 6 \newskip\@topsep
              7 \newskip\@topsepadd
              8 \newskip\@outerparskip

```

(End of definition for \topsep and others.)

```

\leftmargin
\rightmargin 9 \newdimen\leftmargin
\listparindent 10 \newdimen\rightmargin
\itemindent  11 \newdimen\listparindent
\labelwidth  12 \newdimen\itemindent
\labelsep    13 \newdimen\labelwidth
\linewidth   14 \newdimen\labelsep
\@totalleftmargin 15 \newdimen\linewidth
              16 \newdimen\@totalleftmargin \@totalleftmargin=\z@

```

(End of definition for \leftmargin and others.)

```

\leftmargini
\leftmarginii 17 \newdimen\leftmargini
\leftmarginiii 18 \newdimen\leftmarginii
\leftmarginiv 19 \newdimen\leftmarginiii
\leftmarginv 20 \newdimen\leftmarginiv
\leftmarginvi 21 \newdimen\leftmarginv
                22 \newdimen\leftmarginvi

```

(End of definition for \leftmargini and others.)

```

\@listdepth
\@itempenalty 23 \newcount\@listdepth \@listdepth=0
\@beginparpenalty 24 \newcount\@itempenalty
\@endparpenalty 25 \newcount\@beginparpenalty
                26 \newcount\@endparpenalty

```

(End of definition for \@listdepth and others.)

```

\@labels
                27 \newbox\@labels

```

(End of definition for \@labels.)

```

\if@inlabel
\@inlabelfalse 28 \newif\if@inlabel \@inlabelfalse
\@inlabeltrue

```

(End of definition for \if@inlabel, \@inlabelfalse, and \@inlabeltrue.)

```

\if@newlist
\@newlistfalse 29 \newif\if@newlist \@newlistfalse
\@newlisttrue

```

(End of definition for \if@newlist, \@newlistfalse, and \@newlisttrue.)

```

\if@noparitem
\@noparitemfalse 30 \newif\if@noparitem \@noparitemfalse
\@noparitemtrue

```

(End of definition for \if@noparitem, \@noparitemfalse, and \@noparitemtrue.)

```

\if@noparlist
\@noparlistfalse 31 \newif\if@noparlist \@noparlistfalse
\@noparlisttrue

```

(End of definition for \if@noparlist, \@noparlistfalse, and \@noparlisttrue.)

```

\if@noitemarg
\@noitemargfalse 32 \newif\if@noitemarg \@noitemargfalse
\@noitemargtrue

```

(End of definition for \if@noitemarg, \@noitemargfalse, and \@noitemargtrue.)

```

\if@newlist
\@newlistfalse 33 \newif\if@nbrlist \@nbrlistfalse
\@newlisttrue

```

(End of definition for \if@newlist, \@newlistfalse, and \@newlisttrue.)

`list\list)`

```
34 \def\list#1#2{%
35   \ifnum \@listdepth >5\relax
36     \@toodeep
37   \else
38     \global\advance\@listdepth\@ne
39   \fi
40   \rightmargin\z@
41   \listparindent\z@
42   \itemindent\z@
43   \csname @list\romannumeral\the\@listdepth\endcsname
44   \def\@itemlabel{#1}%
45   \let\makelabel\@mklab
46   \@nmbrrlistfalse
47   #2\relax
48   \@trivlist
49   \parskip\parsep
50   \parindent\listparindent
51   \advance\linewidth -\rightmargin
52   \advance\linewidth -\leftmargin
53   \advance\@totalleftmargin \leftmargin
54   \parshape \@ne \@totalleftmargin \linewidth
55   \ignorespaces}
```

(End of definition for \list.)

`\par@deathcycles`

```
56 \newcount\par@deathcycles
```

(End of definition for \par@deathcycles.)

`\@trivlist` Because `\par` is sometimes made a no-op it is possible for a missing `\item` to produce a loop that does not fill memory and so never gets trapped by T_EX. We thus need to trap this here by setting `\par` to count the number of times a paragraph is called with no progress being made started.

```
57 \def\@trivlist{%
58   \if@noskipsec \leavevmode \fi
59   \@topsepadd \topsep
60   \ifvmode
61     \advance\@topsepadd \partopsep
62   \else
63     \unskip \par
64   \fi
65   \if@inlabel
66     \@nparitemtrue
67     \@nparlisttrue
68   \else
69     \if@newlist \@noitemerr \fi
70     \@nparlistfalse
71     \@topsep \@topsepadd
72   \fi
73   \advance\@topsep \parskip
74   \leftskip \z@skip
75   \rightskip \@rightskip
76   \parfillskip \@flushglue
```

```

77 \par@deathcycles \z@
78 \@setpar{\if@newlist
79     \advance\par@deathcycles \@ne
80     \ifnum \par@deathcycles >\@m
81         \noitemerr
82         {\@@par}%
83     \fi
84     \else
85         {\@@par}%
86     \fi}%
87 \global \@newlisttrue
88 \@outerparskip \parskip}

```

(End of definition for \@trivlist.)

~~trivlist~~(*list*.)

```

89 \def\trivlist{%
90 \parsep\parskip
91 \@nbrlistfalse
92 \@trivlist
93 \labelwidth\z@
94 \leftmargin\z@
95 \itemindent\z@

```

We initialise \@itemlabel so that a trivlist with an \item not having an optional argument doesn't produce an error message.

```

96 \let\@itemlabel\@empty
97 \def\makelabel##1{##1}}

```

(End of definition for \trivlist.)

\endlist

```

98 \def\endlist{%
99 \global\advance\@listdepth\@m@ne
100 \endtrivlist}

```

(End of definition for \endlist.)

The definition of \trivlist used to be in lt-space.dtx so that other commands could be 'let to it'. They now use \def.

\endtrivlist

```

101 \def\endtrivlist{%
102 \if@inlabel
103 \leavevmode
104 \global \@inlabelfalse
105 \fi
106 \if@newlist
107 \noitemerr
108 \global \@newlistfalse
109 \fi
110 \ifhmode\unskip \par

```

We also check if we are in math mode and issue an error message if so (hoping that \@currenvir resolves suitably). Otherwise the usual "perhaps a missing item" error will get triggered later which is confusing.

```

111 \else

```

```

112   \@inmatherr{\end{\@currenvir}}%
113   \fi
114   \if@nolist \else
115   \ifdim\lastskip >\z@
116     \@tempkipa\lastskip \vskip -\lastskip
117     \advance\@tempkipa\parskip \advance\@tempkipa -\@outerparskip
118     \vskip\@tempkipa
119   \fi
120   \@endparenv
121 \fi
122 }

```

(End of definition for \endtrivlist.)

`\@endparenv` To suppress the paragraph indentation in text immediately following a paragraph-making environment, `\everypar` is changed to remove the space, and `\par` is redefined to restore `\everypar`. Instead of redefining `\par` and `\everypar`, `\@endparenv` was changed to set the `@endpe` switch, letting `\end` redefine `\par` and `\everypar`.

This allows paragraph-making environments to work right when called by other environments. (Changed 27 Oct 86)

In 2024 this logic was partially replaced with a new algorithm:

- `\if@endpe` is now set globally to `true` or `false`.
- In addition `\@endpetrue` initiates an `\aftergroup` call to `\propagate@doendpe` if it is used inside a group.
- `\propagate@doendpe` in turn checks the status of `\if@endpe` and if that is `true` it calls `\@doendpe` otherwise it does nothing.
- `\@doendpe` in turn calls `\@endpetrue` and also makes the necessary changes to `\par` and `\everypar` so that they handle as before any empty line that follows the environment.
- Because of the `\@endpetrue` we get another `\aftergroup`, so the mechanism slowly migrates out of several groups if those follow immediately after the end of the environment. If, however, there is a new paragraph started or an explicit `\par` before the next group ends then this will result in a call to `\@endpefalse` and the migration stops (note that `\propagate@doendpe` is still called once after the group but does nothing).
- Using this approach something like

```

{% some customization here
\begin{equation}
  x=y
\end{equation}}
some text

```

is still correctly identified as a paragraph continuation so that there is no indentation before `some text`.

- We can get away with using global settings of `\if@endpe` even in nested situations (without keeping track of the status in a stack), because the switch change is made only at the very end of such environments, basically directly before the `\endgroup` in `\end`, and it is later set back to `false` by the next `\everypar` or the next `\par`. Even if the environment is called without using `\begin \end`, the situation doesn't change (or rather cause a problem).
- However, there is one scenario where the new approach would change the behavior. If a box is being built, e.g., with `\setbox`, we have now the case that a `\@endpetrue` inside would migrate out into a context in which it should not be true (because the box might get used elsewhere, e.g., a float). In the past, due to local switch changes, that didn't happen, i.e., `\if@endpe` would revert to `false` at the end of the box definition.
- Thus, to avoid that one has to explicitly set it back to false at the end of such constructions, just as we also need to prevent colors from migrating out. Thus the correct place to do this is in `\color@endgroup` because that is always called at that point.

```

123 \def\@endparenv{%
124   \addpenalty\@endparpenalty\addvspace\@topsepadd\@endpetrue}
125 <latexrelease>\IncludeInRelease{2015/01/01}{\@doendpe}{clubpenalty fix}%
126 \def\@doendpe{\@endpetrue
127   \def\par{\@restorepar

```

If a section heading changes `\clubpenalty` to keep lines after it together then this modification is restored via the `\everypar` mechanism at the start of the next paragraph. As we destroy the contents of this token here we explicitly set `\clubpenalty` back to its default.

```

128         \clubpenalty\@clubpenalty
129         \@endpefalse
130         \everypar{\par}%

```

Use `\setbox0=\lastbox` instead of `\hskip -\parindent` so that a `\noindent` becomes a no-op when used before a line immediately following a list environment(23 Oct 86).

```

131   \everypar
132     {\setbox\z@\lastbox}%
133     \everypar{\@endpefalse}}
134 <latexrelease>\EndIncludeInRelease
135 <latexrelease>\IncludeInRelease{0000/00/00}{\@doendpe}{clubpenalty fix}%
136 <latexrelease>\def\@doendpe{\@endpetrue
137 <latexrelease>   \def\par{\@restorepar\everypar{\par\@endpefalse}\everypar
138 <latexrelease>     {\setbox\z@\lastbox}\everypar{\@endpefalse}}
139 <latexrelease>\EndIncludeInRelease

```

(End of definition for \@endparenv and \@doendpe.)

```

\if@endpe As outlined above these are no longer simple switches, but we keep the name because it
\@endpefalse is used all over the place.
\@endpetrue
140 \newif\if@endpe
\propagate@doendpe

```

```

141 </2ekernel>
142 <*2ekernel | latexrelease>
143 <latexrelease>\IncludeInRelease{2024/11/01}%
144 <latexrelease>          {\@endpetrue}{New @endpe handling}%
145 \def\@endpefalse{\global\let\if@endpe\iffalse}
146 \def\@endpetrue {%
147   \global\let\if@endpe\iftrue

```

If we are inside a simple or a semi-simple group then propagate to the outside, for all other group types do nothing. Normally, we would start out in the group opened by `\begin` (type 14). When we migrate out of that we are either on top-level (type 0) or in another semi-simple group (type 14) or in some other group. Thus, the best order of tests is to first test for 14, then for 0 and finally for 1 (simple group).

```

148   \ifnum\currentgrouptype =14      % semi-simple group
149     \aftergroup\propagate@doendpe
150   \else
151     \ifnum\currentgrouptype =\z@    % no group: top-level
152     \else
153       \ifnum\currentgrouptype =\@ne % simple group
154       \aftergroup\propagate@doendpe
155     \fi
156   \fi
157 \fi
158 }

```

If `\if@endpe` is still true after the group ends, we run `\@doendpe` that in turn runs another `\@endpetrue` (besides other things), thus propagating further if necessary. However, if the endpe situation got resolved and `\if@endpe` is false then nothing further happens.

```

159 \def\propagate@doendpe{\if@endpe \@doendpe \fi}
160 </2ekernel | latexrelease>
161 <latexrelease>\EndIncludeInRelease
162 <latexrelease>\IncludeInRelease{0000/00/00}%
163 <latexrelease>          {\@endpetrue}{New @endpe handling}%
164 <latexrelease>
165 <latexrelease>
166 <latexrelease>\def\@endpefalse{\let\if@endpe\iffalse}
167 <latexrelease>\def\@endpetrue{\let\if@endpe\iftrue}
168 <latexrelease>
169 <latexrelease>\EndIncludeInRelease
170 <*2ekernel>

```

(End of definition for \if@endpe and others.)

`\@mklab`

```

171 \def\@mklab#1{\hfil #1}

```

(End of definition for \@mklab.)

`\item`

```

172 \def\item{%
173   \@inmatherr\item
174   \@ifnextchar [\@item{\@noitemargtrue \@item[\@itemlabel]}}

```

(End of definition for \item.)

`\@donoparitem`

```
175 \def\@donoparitem{%
176   \@noparitemfalse
177   \global\setbox\@labels\hbox{\hskip -\leftmargin
178                                 \unhbox\@labels
179                                 \hskip \leftmargin}%
180   \if@minipage\else
181     \@tempskipa\lastskip
182     \vskip -\lastskip
183     \advance\@tempskipa\@outerparskip
184     \advance\@tempskipa -\parskip
185     \vskip\@tempskipa
186   \fi}
```

(End of definition for \@donoparitem.)

`\@item`

```
187 \def\@item[#1]{%
188   \if@noparitem
189     \@donoparitem
190   \else
191     \if@inlabel
192       \indent \par
193     \fi
194     \ifhmode
195       \unskip\unskip \par
196     \fi
197     \if@newlist
198       \if@nobreak
199         \@nbitem
200       \else
201         \addpenalty\@beginparpenalty
202         \addvspace\@topsep
203         \addvspace{-\parskip}%
204       \fi
205     \else
206       \addpenalty\@itempenalty
207       \addvspace\itemsep
208     \fi
209     \global\@inlabeltrue
210   \fi
211   \everypar{%
212     \@minipagefalse
213     \global\@newlistfalse
```

This `\if@inlabel` check is needed in case an item starts of inside a group so that `\everypar` does not become empty outside that group.

```
214   \if@inlabel
215     \global\@inlabelfalse
```

The paragraph indent is now removed by using `\setbox...` since this makes `\noindent` a no-op here, as it should be. Thus the following comment is redundant but is left here for the sake of future historians: this next command was changed from an `hskip` to a kern to avoid a break point after the `parindent` box: the skip could cause a line-break if a very long label occurs in `raggedright` setting. If `\noindent` was used after `\item` want

to cancel the `\itemindent` skip. This case can be detected as the indentation box will be void.

```

216     {\setbox\z@\lastbox
217     \ifvoid\z@
218     \kern-\itemindent
219     \fi}%

220     \box\@labels
221     \penalty\z@
222     \fi

```

This code is intended to prevent a page break after the first line of an item that comes immediately after a section title. It may be sensible to always forbid a page break after one line of an item? As with all such settings of `\clubpenalty` it is local so will have no effect if the item starts in a group.

Only resetting `\@nobreak` when it is true is now essential since now it is sometimes set locally.

```

223     \if@nobreak
224     \@nobreakfalse
225     \clubpenalty \@M
226     \else
227     \clubpenalty \@clubpenalty
228     \everypar{}%
229     \fi}%

230 \if@noitemarg
231 \@noitemargfalse
232 \if@nmbrrlist

233     \refstepcounter\@listctr
234     \fi
235 \fi

```

We use `\sbox` to support colour commands.

```

236 \sbox\@tempboxa{\makelabel{#1}}%
237 \global\setbox\@labels\hbox{%
238 \unhbox\@labels
239 \hskip \itemindent
240 \hskip -\labelwidth
241 \hskip -\labelsep
242 \ifdim \wd\@tempboxa >\labelwidth
243 \box\@tempboxa

244 \else
245 \hbox to\labelwidth {\unhbox\@tempboxa}%
246 \fi
247 \hskip \labelsep}%
248 \ignorespaces}

```

(End of definition for \@item.)

`\makelabel`

```

249 \def\makelabel#1{%
250 \@latex@error{Lonely \string\item--perhaps a missing
251 list environment}\@ehc}

```

(End of definition for `\makeLabel`.)

`\@nbitem`

```
252 \def\@nbitem{%
253   \@tempskipa\@outerparskip
254   \advance\@tempskipa -\parskip
255   \addvspace\@tempskipa}
```

(End of definition for `\@nbitem`.)

`\usecounter`

```
256 \def\usecounter#1{\@nmblisttrue\def\@listctr{#1}\setcounter{#1}\z@}
```

(End of definition for `\usecounter`.)

1.6 Itemize and Enumerate

Enumeration is done with four counters: `enumi`, `enumii`, `enumiii` and `enumiv`, where `enumN` controls the numbering of the Nth level enumeration. The label is generated by the commands `\labelenumi` ... `\labelenumiv`, which should be defined by the document style. Note that `\p@enumN\theenumN` defines the output of a `\ref` command. A typical definition might be:

```
\def\theenumii{\alph{enumii}}
\def\p@enumii{\theenumi}
\def\labelenumii{\theenumii}
```

which will print the labels as ‘(a)’, ‘(b)’, ... and print a `\ref` as ‘3a’.

The item numbers are moved to the right of the label box, so they are always a distance of `\labelsep` from the item.

`\@enumdepth` holds the current enumeration nesting depth.

Itemization is controlled by four commands: `\labelitemi`, `\labelitemii`, `\labelitemiii`, and `\labelitemiv`. To cause the second-level list to be bulleted, you just define `\labelitemii` to be `•`. `\@itemspacing` and `\@itemdepth` are the analogs of `\@enumspacing` and `\@enumdepth`.

Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

```
\enumerate ==
BEGIN
  if \@enumdepth > 3
  then errormessage: “Too deeply nested”.
  else \@enumdepth :=L \@enumdepth + 1
        \@enumctr :=L eval(enum@\romannumeral\the\@enumdepth)
        \list{\label(\@enumctr)}
              {\usecounter{\@enumctr}
               \makeLabel{LABEL} == \hss \llap{LABEL}}
  fi
END
```

```
\endenumerate == \endlist
```

End of historical L^AT_EX 2.09 comments.

`\@enumdepth`

```
257 \newcount\@enumdepth \@enumdepth = 0
```

(End of definition for \@enumdepth.)

`\c@enumi`

`\c@enumii`

`\c@enumiii`

`\c@enumiv`

```
258 \@definecounter{enumi}
```

```
259 \@definecounter{enumii}
```

```
260 \@definecounter{enumiii}
```

```
261 \@definecounter{enumiv}
```

(End of definition for \c@enumi and others.)

`enumerate (env.)`

```
262 \def\enumerate{%
```

```
263   \ifnum \@enumdepth >\thr@@\toodeep\else
```

```
264     \advance\@enumdepth\@ne
```

```
265     \edef\@enumctr{enum\romannumeral\the\@enumdepth}%
```

```
266     \expandafter
```

```
267     \list
```

```
268       \csname label\@enumctr\endcsname
```

```
269       {\usecounter\@enumctr\def\makelabel##1{\hss\llap{##1}}}%
```

```
270   \fi}
```

```
271 \let\endenumerate =\endlist
```

Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

```
\itemize ==
```

```
  BEGIN
```

```
    if \@itemdepth > 3
```

```
      then errormessage: 'Too deeply nested'.
```

```
    else \@itemdepth :=L \@itemdepth + 1
```

```
      \@itemitem == eval(labelitem\romannumeral\the\@itemdepth)
```

```
      \list{\@nameuse{\@itemitem}}
```

```
        {\makelabel{LABEL} == \hss \llap{LABEL}}
```

```
    fi
```

```
  END
```

```
\enditemize == \endlist
```

End of historical L^AT_EX 2.09 comments.

`\@itemdepth`

```
272 \newcount\@itemdepth \@itemdepth = 0
```

(End of definition for \@itemdepth.)

`itemize (env.)`

```
273 \def\itemize{%
```

```
274   \ifnum \@itemdepth >\thr@@\toodeep\else
```

```
275     \advance\@itemdepth\@ne
```

```
276     \edef\@itemitem{labelitem\romannumeral\the\@itemdepth}%
```

```
277 \expandafter
278 \list
279 \csname@itemitem\endcsname
280 {\def\makeLabel##1{\hss\llap{##1}}}%
281 \fi}
282 \let\enditemize =\endlist
283 \endkernel)
```

File 40

ltboxes.dtx

1 L^AT_EX Box commands

- `\makebox` `\makebox[wid][pos]{obj}`
Puts *obj* in an `\hbox` of width *wid*, positioned by *pos*.
The possible *pos* are:
s stretched,
l flushleft,
r flushright,
c (default) centred.
If *wid* is missing, then *pos* is also missing and *obj* is put in an `\hbox` of its natural width.
- `\makebox(x,y)[pos]{obj}`
Puts *obj* in an `\hbox` of width $x * \unitlength$ and height $y * \unitlength$.
pos arguments are s, l, r or c (default) for stretched, flushleft, flushright or centred, and t or b for top, bottom – or combinations like tr or rb. Default for horizontal and vertical are centered. Note that in this picture mode version of `\makebox` a [b] aligns on the *bottom* of the text as documented. If you want to align on the *baseline* use `\makebox(,) [b]{\raisebox{0pt}{\height}[0pt]{xyz}}` or `\makebox(,) [b]{\smash{xyz}}`
- `\mbox` `\mbox{obj}` The same as `\makebox{obj}`, but is more efficient as no checking for optional arguments is done.
- `\newsavebox` `\newsavebox{cmd}` : If *cmd* is undefined, then defines it to be a T_EX box register.
- `\savebox` `\savebox{cmd} ...` : *cmd* is defined to be a T_EX box register, and the ‘...’ are any `\makebox` arguments. It is like `\makebox`, except it doesn’t produce text but saves the value in `\box cmd`.
- `\sbox` `\sbox{cmd}{obj}` is an efficient abbreviation for `\savebox{cmd}{obj}`.
- `lrbox (env.)` `\begin{lrbox}{cmd}{text}\end{lrbox}` is equivalent to `\sbox{cmd}{text}` except that any white space at the beginning and end of *text* is ignored.
- `\framebox` `\framebox ...` : like `\makebox`, except it puts a ‘frame’ around the box. The frame is made of lines of thickness `\fboxrule`, separated by space `\fboxsep` from the text – except for `\framebox(X,Y) ...`, where the thickness of the lines is as for the picture environment, and there is no separation added.
- `\fbox` `\fbox{obj}` is an abbreviation for `\framebox{obj}`.
- `\parbox` `\parbox[pos][height][inner-pos]{width}{text}` : Makes a box with `\hsize` *width*, positioned by *pos* as follows: c : `\vcenter` (placed in $\$. . \$$ if not in math mode) b : `\vbox` t : `\vtop` default value is c. Sets `\hsize := width` and calls `\@parboxrestore`, which does the following: Restores the original definitions of:
- `\par`
 - `\`
 - `\-`
 - `\’`
 - `\‘`
 - `\=`

Resets the following parameters:

<code>\parindent</code>	=	0pt	
<code>\parskip</code>	=	0pt	added 20 Jan 87
<code>\linewidth</code>	=	<code>\hsize</code>	
<code>\totalleftmargin</code>	=	0pt	
<code>\leftskip</code>	=	0pt	
<code>\rightskip</code>	=	0pt	
<code>\@rightskip</code>	=	0pt	
<code>\parfillskip</code>	=	0pt plus 1fil	
<code>\lineskip</code>	=	<code>\normallineskip</code>	
<code>\baselineskip</code>	=	<code>\normalbaselineskip</code>	

Calls `\sloppy`

`minipage (env.)` Note: `\@arrayparboxrestore` same as `\@parboxrestore` but it doesn't restore `\@textwidth == \columnwidth == box width` changes footnotes by redefining:

```
\@mpfn == mpfootnote
\thempfn == \thempfootnote
\@footnotetext == \@mpfootnotetext
  resets the following list environment parameters:
\@listdepth == \@mplistdepth
where \@mplistdepth is initialized to zero,
```

and executes `\@minipagerestore` to allow the document style to reset any other parameters it desires. It sets `@minipage` true, and resets `\everypar` to set it false. This switch keeps `\addvspace` from putting space at the top of a minipage.

Change added 24 May 89: `\minipage` sets `@minipage` globally; `\endminipage` resets it false.

```
\rule      \rule[raised]{width}{height} : Makes a width * height rule, raised
           (raised).
\underline \underline{text} : Makes an underlined hbox with text in it.
\raisebox  \raisebox{distance}[height][depth]{box} :
           Raises box up by distance length (down if distance negative). Makes TEX think
           that the new box extends height above the line and depth below, for a total vertical
           length of height+depth. Default values of height & depth = actual height and
           depth of box in new position.
           1 <2kernel
           2 \message{boxes,}
```

```
\makebox   \makebox User level command just looks for optional [ or (.
           3 </2kernel
           4 <latexrelease>\IncludeInRelease{2015/01/01}%
           5 <latexrelease>           {\makebox}{Make \makebox robust}%
           6 <*2kernel | latexrelease>
           7 \DeclareRobustCommand\makebox{%
           8   \leavevmode
           9   \@ifnextchar(%)
          10     \makepicbox
          11     {\@ifnextchar [\@makebox\mbox]}%
          12 </2kernel | latexrelease>
          13 <latexrelease>\EndIncludeInRelease
```

```

14 <latexrelease>\IncludeInRelease{0000/00/00}%
15 <latexrelease>          {\makebox}{Make \makebox robust}%
16 <latexrelease>\def\makebox{%
17 <latexrelease>  \leavevmode
18 <latexrelease>  \ifnextchar(%)
19 <latexrelease>    \makepicbox
20 <latexrelease>    {\ifnextchar[\@makebox\mbox]}%
21 <latexrelease>\expandafter\let\csname makebox \endcsname\@undefined
22 <latexrelease>\EndIncludeInRelease
23 <*2kernel>

```

(End of definition for \makebox.)

\mbox The basic horizontal box command for L^AT_EX.

```
24 \DeclareRobustCommand\mbox[1]{\leavevmode\hbox{#1}}
```

(End of definition for \mbox.)

\@makebox Look for a possible second optional argument (defaults to c).

```
25 \def\@makebox[#1]{%
26   \ifnextchar [{\@imakebox[#1]}{\@imakebox[#1][c]}%

```

(End of definition for \@makebox.)

\@begin@tempboxa Helper macro for supporting \height, \width etc. Grab #1 into \@tempboxa and measure it.

```
27 \long\def\@begin@tempboxa#1#2{%
28   \begingroup
29   \setbox\@tempboxa#1{\color@begingroup#2\color@endgroup}%
30   \def\width{\wd\@tempboxa}%
31   \def\height{\ht\@tempboxa}%
32   \def\depth{\dp\@tempboxa}%
33   \let\totalheight\@ovri
34   \totalheight\height
35   \advance\totalheight\depth}

```

(End of definition for \@begin@tempboxa.)

\@end@tempboxa End the group started by \@begin@tempboxa, so that the scope of \height only includes the ‘length’ argument to the user-command.

```
36 \let\@end@tempboxa\endgroup
```

(End of definition for \@end@tempboxa.)

\bm@c Set up spacing.

```

\bm@l 37 \def\bm@c{\hss\unhbox\@tempboxa\hss}
\bm@r 38 \def\bm@l{\unhbox\@tempboxa\hss}\let\bm@t\bm@l
\bm@s 39 \def\bm@r{\hss\unhbox\@tempboxa}\let\bm@b\bm@r
\bm@t 40 \def\bm@s{\unhbox\@tempboxa}

```

\bm@b *(End of definition for \bm@c and others.)*

`\@imakebox` Internal form of `\makebox`.

```
41 </2ekernel>
42 <latexrelease>\IncludeInRelease{2023/11/01}%
43 <latexrelease>          {\@imakebox}{Unknown alignment warning}%
44 <*2ekernel | latexrelease>
45 \long\def\@imakebox[#1][#2]#3{%
46   \@begin@tempboxa\hbox{#3}%
47   \setlength\@tempdima{#1}%      support calc
48   \hb@xt@\@tempdima{%
49     \expandafter\ifx\csname bm@#2\endcsname\relax
50     \bm@c
51     \@latex@warning{Unexpected alignment #2}%
52   \else
53     \csname bm@#2\endcsname
54   \fi}%
55   \@end@tempboxa}
56 </2ekernel | latexrelease>
57 <latexrelease>\EndIncludeInRelease
58 <latexrelease>\IncludeInRelease{0000/00/00}%
59 <latexrelease>          {\@imakebox}{Unknown alignment warning}%
60 <latexrelease>\long\def\@imakebox[#1][#2]#3{%
61 <latexrelease>  \@begin@tempboxa\hbox{#3}%
62 <latexrelease>  \setlength\@tempdima{#1}%      support calc
63 <latexrelease>  \hb@xt@\@tempdima{\csname bm@#2\endcsname}%
64 <latexrelease>  \@end@tempboxa}
65 <latexrelease>\EndIncludeInRelease
66 <*2ekernel>
```

(End of definition for \@imakebox.)

`\@makepicbox` Picture mode form of `\makebox`.

```
67 \def\@makepicbox(#1,#2){%
68   \@ifnextchar[{\@imakepicbox(#1,#2)}{\@imakepicbox(#1,#2) []}]}
```

(End of definition for \@makepicbox.)

`\@imakepicbox` picture mode version

```
69 </2ekernel>
70 <*2ekernel | latexrelease>
71 <latexrelease>\IncludeInRelease{2020/10/01}%
72 <latexrelease>          {\@imakepicbox}{default units}%
73 \long\def\@imakepicbox(#1,#2)[#3]#4{%
74   \@defaultunitsset\@tempdimc{#2}\unitlength
75   \vbox to\@tempdimc
76   {\let\mb@b\vss \let\mb@l\hss\let\mb@r\hss
77    \let\mb@t\vss
78    \@tfor\reserved@a :=#3\do{%
79      \if s\reserved@a
80        \let\mb@l\relax\let\mb@r\relax
81      \else
82        \expandafter\let\csname mb@\reserved@a\endcsname\relax
83      \fi}%
84   \mb@t
85   \@defaultunitsset\@tempdimc{#1}\unitlength
```

```

86     \hb@xt@\@tempdimc{\mb@l #4\mb@r}%
87     \mb@b

```

This kern ensures that a b option aligns on the bottom of the text rather than the baseline. this is the documented behaviour in the L^AT_EX Book. The kern is removed in compatibility mode.

```

88     \kern\z@}}
89 </2ekernel | latexrelease>
90 <latexrelease>\EndIncludeInRelease
91 <latexrelease>\IncludeInRelease{0000/00/00}%
92 <latexrelease>           {\@imakepicbox}{default units}%
93 <latexrelease>\long\def\@imakepicbox(#1,#2)[#3]#4{%
94 <latexrelease>  \vbox to#2\unitlength
95 <latexrelease>  {\let\mb@b\vss \let\mb@l\hss\let\mb@r\hss
96 <latexrelease>   \let\mb@t\vss
97 <latexrelease>   \@tfor\reserved@a :=#3\do{%
98 <latexrelease>     \if s\reserved@a
99 <latexrelease>       \let\mb@l\relax\let\mb@r\relax
100 <latexrelease>     \else
101 <latexrelease>       \expandafter\let\csname mb@\reserved@a\endcsname\relax
102 <latexrelease>     \fi}%
103 <latexrelease>   \mb@t
104 <latexrelease>   \hb@xt@ #1\unitlength{\mb@l #4\mb@r}%
105 <latexrelease>   \mb@b
106 <latexrelease>   \kern\z@}}
107 <latexrelease>\EndIncludeInRelease
108 <*2ekernel>

```

(End of definition for \@imakepicbox.)

`\set@color` This macro is initially a no-op, but the color package will redefine it to insert a `\special`.

```

109 \let\set@color\relax

```

(End of definition for \set@color.)

`\color@begingroup` `\color@setgroup` `\normalcolor` `\color@hbox` `\color@vbox` `\color@endbox` In the past these macros were initially no-ops, and the color package redefined them to be `\begingroup`, `\endgroup`, `\begingroup\set@color`, `\hbox\bgroup\color@begingroup`, `\color@endgroup\egroup`. and *<set to main document color>* respectively.

Nowadays we always set the group already in the kernel as this makes the coding simpler.

```

110 </2ekernel>
111 <*2ekernel | latexrelease>
112 <latexrelease>\IncludeInRelease{2021/06/01}%
113 <latexrelease>           {\color@begingroup}{color group settings}%
114 \let\color@begingroup\begingroup
115 \def\color@setgroup{\color@begingroup}           % changed further in color package
116 \let\normalcolor\relax                           % remains untouched; only changed in a color pa
117 \def\color@hbox{\hbox\bgroup\color@begingroup}
118 \def\color@vbox{\vbox\bgroup\color@begingroup}
119 \def\color@endbox{\color@endgroup\egroup}
120 </2ekernel | latexrelease>
121 <latexrelease>\EndIncludeInRelease

```

```

122 <latexrelease>\IncludeInRelease{0000/00/00}%
123 <latexrelease>          {\color@begingroup}{color group settings}%
124 <latexrelease>
125 <latexrelease>\let\color@begingroup\relax
126 <latexrelease>\let\color@setgroup\relax
127 <latexrelease>\let\normalcolor\relax
128 <latexrelease>\let\color@hbox\relax
129 <latexrelease>\let\color@vbox\relax
130 <latexrelease>\let\color@endbox\relax
131 <latexrelease>
132 <latexrelease>\EndIncludeInRelease
133 <*2ekernel>

```

(End of definition for \color@begingroup and others.)

`\color@endgroup` This macro is separated out because it received an update in 2024, so requires its own rollback.

```

134 </2ekernel>
135 <*2ekernel | latexrelease>
136 <latexrelease>\IncludeInRelease{2024/11/01}%
137 <latexrelease>          {\color@endgroup}{color group settings}%

```

Beside `\endgraf` for handling vertical boxes we also reset `\if@endpe` as we are leaving the context.

```

138 \def\color@endgroup{\endgraf\@endpefalse\endgroup}
139 <latexrelease>
140 </2ekernel | latexrelease>
141 <latexrelease>\EndIncludeInRelease

142 <latexrelease>\IncludeInRelease{2021/06/01}%
143 <latexrelease>          {\color@endgroup}{color group settings}%
144 <latexrelease>\def\color@endgroup{\endgraf\endgroup}
145 <latexrelease>
146 <latexrelease>\EndIncludeInRelease

147 <latexrelease>\IncludeInRelease{0000/00/00}%
148 <latexrelease>          {\color@endgroup}{color group settings}%
149 <latexrelease>
150 <latexrelease>\let\color@endgroup\relax
151 <latexrelease>
152 <latexrelease>\EndIncludeInRelease
153 <*2ekernel>

```

(End of definition for \color@endgroup.)

`\newsavebox` Allocate a new ‘savebox’.

```

154 \def\newsavebox#1{\@ifdefinable{#1}{\newbox#1}}

```

(End of definition for \newsavebox.)

`\savebox` Save #1 in a box register.

```

155 </2ekernel>
156 <latexrelease>\IncludeInRelease{2015/01/01}%
157 <latexrelease>          {\savebox}{Make \savebox robust}%
158 <*2ekernel | latexrelease>
159 \DeclareRobustCommand\savebox[1]{%

```

```

160 \@ifnextchar(%)
161   {\@savepicbox#1}{\@ifnextchar[{\@savebox#1}{\sbox#1}}}%
162 </2ekernel | latexrelease)
163 <latexrelease)\EndIncludeInRelease
164 <latexrelease)\IncludeInRelease{0000/00/00}%
165 <latexrelease)      {\savebox}{Make \savebox robust}%
166 <latexrelease)\def\savebox#1{%
167 <latexrelease)  \@ifnextchar(%)
168 <latexrelease)    {\@savepicbox#1}{\@ifnextchar[{\@savebox#1}{\sbox#1}}}%
169 <latexrelease)\expandafter\let\csname savebox \endcsname\undefined
170 <latexrelease)\EndIncludeInRelease
171 <*2ekernel)

```

(End of definition for \savebox.)

\sbox Save #1 in a box register.

```

172 \DeclareRobustCommand\sbox[2]{\setbox#1\hbox{%
173   \color@setgroup#2\color@endgroup}}

```

(End of definition for \sbox.)

\@savebox Look for second optional argument.

```

174 \def\@savebox#1[#2]{%
175   \@ifnextchar [{\@isavebox#1[#2]}{\@isavebox#1[#2][c]}}

```

(End of definition for \@savebox.)

\@isavebox

```

176 \long\def\@isavebox#1[#2][#3]#4{%
177   \sbox#1{\@imakebox[#2][#3]{#4}}

```

(End of definition for \@isavebox.)

\@savepicbox Picture mode version of \savebox.

```

178 \def\@savepicbox#1(#2,#3){%
179   \@ifnextchar[%]
180     {\@isavepicbox#1(#2,#3)}{\@isavepicbox#1(#2,#3) []}}

```

(End of definition for \@savepicbox.)

\@isavepicbox Picture mode version of \savebox.

```

181 \long\def\@isavepicbox#1(#2,#3)[#4]#5{%
182   \sbox#1{\@imakepicbox(#2,#3)[#4]{#5}}

```

(End of definition for \@isavepicbox.)

\lrbox **lrbox**: the new environment form of \sbox. Use \aftergroup tricks to enable a *local* assignment to be made to the box, in a way that it still has an effect *outside* the **lrbox** environment.

```

183 \def\lrbox#1{%
184   \edef\reserved@a{%
185     \endgroup
186     \setbox#1\hbox{%
187       \begingroup\aftergroup}%
188     \def\noexpand\@currenvir{\@currenvir}%

```

```

189     \def\noexpand\@currentvline{\on@line}}%
190 \reserved@a
191 \endpefalse
192 \color@setgroup
193 \ignorespaces}

(End of definition for \lrbox.)

\endlrbox End the lrbox environment.
194 \def\endlrbox{\unskip\color@endgroup}

(End of definition for \endlrbox.)

\usebox unchanged
195 \DeclareRobustCommand\usebox[1]{\leavevmode\copy #1\relax}

(End of definition for \usebox.)

\frame The following definition of \frame was written by Pavel Curtis (Extra space removed 14
Jan 88) RmS 92/08/24: Replaced occurrence of \@halfwidth by \@wholewidth
196 \DeclareRobustCommand\frame[1]{%
197 \leavevmode
198 \hbox{%
199 \hskip-\@wholewidth
200 \vbox{%
201 \vskip-\@wholewidth
202 \hrule \@height\@wholewidth
203 \hbox{%
204 \vrule\@width\@wholewidth
205 #1%
206 \vrule\@width\@wholewidth}%
207 \hrule \@height\@wholewidth
208 \vskip-\@wholewidth}%
209 \hskip-\@wholewidth}}

(End of definition for \frame.)

\fbxrule user level parameters,
\fbxsep 210 \newdimen\fbxrule
211 \newdimen\fbxsep

(End of definition for \fbxrule and \fbxsep.)

\fbx Abbreviated framed box command.
212 \DeclareRobustCommand\fbx[1]{%
213 \leavevmode
214 \setbox\@tempboxa\hbox{%
215 \color@begingroup
216 \kern\fbxsep{#1}\kern\fbxsep
217 \color@endgroup}%
218 \@frameb@x\relax}

(End of definition for \fbx.)

```

`\framebox` Framed version of `\makebox`.

```
219 </2ekernel>
220 <latexrelease>\IncludeInRelease{2015/01/01}%
221 <latexrelease>          {\framebox}{Make \framebox robust}%
222 <*2ekernel | latexrelease>
223 \DeclareRobustCommand\framebox{%
224   \ifnextchar (%)
225     \framepicbox{\ifnextchar[\@framebox\fbx]}%
226 </2ekernel | latexrelease>
227 <latexrelease>\EndIncludeInRelease
228 <latexrelease>\IncludeInRelease{0000/00/00}%
229 <latexrelease>          {\framebox}{Make \framebox robust}%
230 <latexrelease>\def\framebox{%
231 <latexrelease>  \ifnextchar (%)
232 <latexrelease>    \framepicbox{\ifnextchar[\@framebox\fbx]}%
233 <latexrelease>\expandafter\let\csname framebox \endcsname \@undefined
234 <latexrelease>\EndIncludeInRelease
235 <*2ekernel>
```

(End of definition for \framebox.)

`\@framebox` Deal with optional arguments.

```
236 \def\@framebox[#1]{%
237   \ifnextchar [%]
238     {\@framebox[#1]}%
239     {\@framebox[#1][c]}}
```

(End of definition for \@framebox.)

`\@ifframebox` The handling the optional arguments. In order to set the whole box, including the frame to the specified dimension, we first determine that dimension from the natural size of the text, #3. calculated width.

```
240 </2ekernel>
241 <latexrelease>\IncludeInRelease{2023/11/01}%
242 <latexrelease>          {\@ifframebox}{Unknown alignment warning}%
243 <*2ekernel | latexrelease>
244 \long\def\@ifframebox[#1][#2]#3{%
245   \leavevmode
246   \@begin@tempboxa\hbox{#3}%
247   \setlength\@tempdima{#1}%
248   \setbox\@tempboxa\hb@xt@\@tempdima
249     {\kern\fbxsep
250     \expandafter\ifx\csname bm@#2\endcsname\relax
251     \bmc
252     \@latex@warning{Unexpected alignment #2}%
253     \else
254     \csname bm@#2\endcsname
255     \fi
256     \kern\fbxsep}%
257   \@frameboxx{\kern-\fbxrule}%
258   \@end@tempboxa}
259 </2ekernel | latexrelease>
260 <latexrelease>\EndIncludeInRelease
261 <latexrelease>\IncludeInRelease{0000/00/00}%
```



```

262 <latexrelease>                {\@iframebox}{Unknown alignment warning}%
263 <latexrelease>\long\def\@iframebox[#1][#2]#3{%
264 <latexrelease> \leavevmode
265 <latexrelease> \@begin@tempboxa\hbox{#3}%
266 <latexrelease> \setlength\@tempdima{#1}%
267 <latexrelease> \setbox\@tempboxa\hb@xt@\@tempdima
268 <latexrelease>     {\kern\fboxsep\csname bm@#2\endcsname\kern\fboxsep}%
269 <latexrelease> \@frameb@x{\kern-\fboxrule}%
270 <latexrelease> \@end@tempboxa}
271 <latexrelease>\EndIncludeInRelease
272 <*2ekernel)

```

(End of definition for \@iframebox.)

`\@frameb@x` Common part of `\framebox` and `\fbox`. #1 is a negative kern in the `\framebox` case so that the vertical rules do not add to the width of the box.

```

273 \def\@frameb@x#1{%
274   \@tempdima\fboxrule
275   \advance\@tempdima\fboxsep
276   \advance\@tempdima\dp\@tempboxa
277   \hbox{%
278     \lower\@tempdima\hbox{%
279       \vbox{%
280         \hrule\@height\fboxrule
281         \hbox{%
282           \vrule\@width\fboxrule
283           #1%
284           \vbox{%
285             \vskip\fboxsep
286             \box\@tempboxa
287             \vskip\fboxsep}%
288           #1%
289           \vrule\@width\fboxrule}%
290         \hrule\@height\fboxrule}%
291         }%
292   }%
293 }

```

(End of definition for \@frameb@x.)

`\@framepicbox` Picture mode version.

```

294 \def\@framepicbox(#1,#2){%
295   \ifnextchar[{\@iframepicbox(#1,#2)}{\@framepicbox(#1,#2) []}

```

(End of definition for \@framepicbox.)

`\@iframepicbox` Picture mode version.

```

296 \long\def\@iframepicbox(#1,#2)[#3]#4{%
297   \frame{\@imakepicbox(#1,#2)[#3]{#4}}

```

(End of definition for \@iframepicbox.)

```

\parbox The main vertical-box command for LATEX.
298 </2ekernel>
299 <latexrelease>\IncludeInRelease{2015/01/01}%
300 <latexrelease>          {\parbox}{Make \parbox robust}%
301 <*2ekernel | latexrelease>
302 \DeclareRobustCommand\parbox{%
303   \@ifnextchar [%]
304     \@iparbox
305     {\@iiiparbox c\relax[s]}}%
306 </2ekernel | latexrelease>
307 <latexrelease>\EndIncludeInRelease
308 <latexrelease>\IncludeInRelease{0000/00/00}%
309 <latexrelease>          {\parbox}{Make \parbox robust}%
310 <latexrelease>\def\parbox{%
311 <latexrelease>  \@ifnextchar [%]
312 <latexrelease>    \@iparbox
313 <latexrelease>      {\@iiiparbox c\relax[s]}}%
314 <latexrelease>\expandafter\let\csname parbox \endcsname\@undefined
315 <latexrelease>\EndIncludeInRelease
316 <*2ekernel>

(End of definition for \parbox.)

\@iparbox Optional argument handling.
317 \def\@iparbox[#1]{%
318   \@ifnextchar [%]
319     {\@iiiparbox[#1]}%
320     {\@iiiparbox{#1}\relax[s]}}

(End of definition for \@iparbox.)

\@iiiparbox Optional argument handling.
321 \def\@iiiparbox#1[#2]{%
322   \@ifnextchar [%]
323     {\@iiiparbox{#1}{#2}}%
324     {\@iiiparbox{#1}{#2}[#1]}}

(End of definition for \@iiiparbox.)

\@iiiparbox The internal version of \parbox.
\@parboxto
325 \let\@parboxto\@empty

326 </2ekernel>
327 <latexrelease>\IncludeInRelease{2023/11/01}%
328 <latexrelease>          {\@iiiparbox}{Unknown alignment warning}%
329 <*2ekernel | latexrelease>
330 \long\def\@iiiparbox#1#2[#3]#4#5{%
331   \leavevmode
332   \@pboxswfalse
333   \setlength\@tempdima{#4}%
334   \@begin@tempboxa\vbox{\hsize\@tempdima\@parboxrestore#5\@par}%
335     \ifx\relax#2\else
336       \setlength\@tempdimb{#2}%
337       \edef\@parboxto{to\the\@tempdimb}%
338       \fi

```

```

339 \if#1b\vbox
340 \else\if #1t\vtop
341 \else\ifmode\vcenter
342 \else\@pboxswtrue $\vcenter
343 \fi\fi\fi
344 \@parboxto{\let\hss\vss\let\unhbox\unvbox
345 \expandafter\ifx\csname bm@#3\endcsname\relax
346 \bm@c
347 \@latex@warning{Unexpected alignment #3}%
348 \else
349 \csname bm@#3\endcsname
350 \fi}%
351 \if@pboxsw \m@th$\fi
352 \@end@tempboxa}
353 </2ekernel | latexrelease>
354 <latexrelease>\EndIncludeInRelease
355 <latexrelease>\IncludeInRelease{0000/00/00}%
356 <latexrelease> \{\@iiiparbox}{Unknown alignment warning}%
357 <latexrelease>\long\def\@iiiparbox#1#2[#3]#4#5{%
358 <latexrelease> \leavevmode
359 <latexrelease> \@pboxswfalse
360 <latexrelease> \setlength\@tempdima{#4}%
361 <latexrelease> \begin@tempboxa\vbox{\hsize\@tempdima\@parboxrestore#5\@par}%
362 <latexrelease> \ifx\relax#2\else
363 <latexrelease> \setlength\@tempdimb{#2}%
364 <latexrelease> \edef\@parboxto{to\the\@tempdimb}%
365 <latexrelease> \fi
366 <latexrelease> \if#1b\vbox
367 <latexrelease> \else\if #1t\vtop
368 <latexrelease> \else\ifmode\vcenter
369 <latexrelease> \else\@pboxswtrue $\vcenter
370 <latexrelease> \fi\fi\fi
371 <latexrelease> \@parboxto{\let\hss\vss\let\unhbox\unvbox
372 <latexrelease> \csname bm@#3\endcsname}%
373 <latexrelease> \if@pboxsw \m@th$\fi
374 <latexrelease> \@end@tempboxa}
375 <latexrelease>\EndIncludeInRelease
376 <*2ekernel)

```

(End of definition for \@iiiparbox and \@parboxto.)

`\@arrayparboxrestore` Restore various paragraph parameters.

The rationale for allowing two normally global flags to be set locally here was stated originally by Donald Arseneau and extended by Chris Rowley. It is because these flags are only set globally to true by section commands, and these should never appear within boxes or, indeed, in any group; and they are only ever set globally to false when they are definitely true.

If anyone is unhappy with this argument then both flags should be treated as in `\@setnobreak`; otherwise this command will be redundant.

```

377 </2ekernel)
378 <latexrelease>\IncludeInRelease{2017-04-15}%
379 <latexrelease> \{\normallineskiplimit}
380 <latexrelease> {reset \lineskiplimit}%
381 <*2ekernel | latexrelease>

```

```

382 \def\@arrayparboxrestore{%
383   \let\if@nobreak\iffalse
384   \let\if@noskipsec\iffalse
385   \let\par\@@par
386   \let\-\@dischyph

```

Redefined accents to allow changes in font encoding

```

387   \let'\@acci\let'\@accii\let\=\@acciii
388   \parindent\z@ \parskip\z@skip
389   \everypar{}%
390   \linewidth\hsize
391   \@totalleftmargin\z@
392   \leftskip\z@skip \rightskip\z@skip \@rightskip\z@skip
393   \parfillskip\@flushglue
394   \lineskip\normallineskip

```

```

395   \lineskiplimit\normallineskiplimit

```

```

396   \baselineskip\normalbaselineskip

```

```

397   \sloppy}

```

```

398 (/2ekernel | latexrelease)

```

```

399 <latexrelease>\EndIncludeInRelease

```

```

400 <latexrelease>\IncludeInRelease{0000-00-00}%

```

```

401 <latexrelease>           {\normallineskiplimit}

```

```

402 <latexrelease>           {reset \lineskiplimit}%

```

```

403 <latexrelease>\def\@arrayparboxrestore{%

```

```

404 <latexrelease> \let\if@nobreak\iffalse

```

```

405 <latexrelease> \let\if@noskipsec\iffalse

```

```

406 <latexrelease> \let\par\@@par

```

```

407 <latexrelease> \let\-\@dischyph

```

```

408 <latexrelease> \let'\@acci\let'\@accii\let\=\@acciii

```

```

409 <latexrelease> \parindent\z@ \parskip\z@skip

```

```

410 <latexrelease> \everypar{}%

```

```

411 <latexrelease> \linewidth\hsize

```

```

412 <latexrelease> \@totalleftmargin\z@

```

```

413 <latexrelease> \leftskip\z@skip \rightskip\z@skip \@rightskip\z@skip

```

```

414 <latexrelease> \parfillskip\@flushglue \lineskip\normallineskip

```

```

415 <latexrelease> \baselineskip\normalbaselineskip

```

```

416 <latexrelease> \sloppy}

```

```

417 <latexrelease>\EndIncludeInRelease

```

```

418 (*2ekernel)

```

(End of definition for \@arrayparboxrestore.)

\@parboxrestore Restore various paragraph parameters, and also \.

```

419 \def\@parboxrestore{\@arrayparboxrestore\let\\\@normalcr}

```

(End of definition for \@parboxrestore.)

\if@minipage Switch that is true at the start of a minipage.

```

420 \def\@minipagefalse{\global\let\if@minipage\iffalse}

```

```

421 \def\@minipagetrue {\global\let\if@minipage\iftrue}

```

```

422 \@minipagefalse

```

(End of definition for \if@minipage.)

`\if@in@minipage@env`

```
423 \newif\if@in@minipage@env
```

(End of definition for \if@in@minipage@env.)

`\minipage` Essentially an environment form of `\parbox`.

```
424 \def\minipage{%
425   \ifnextchar[%
426     \@minipage
427     {\@iiminipage c\relax[s]}}
```

(End of definition for \minipage.)

`\@minipage` Optional argument handling.

```
428 \def\@minipage[#1]{%
429   \ifnextchar[%
430     {\@iiminipage{#1}}%
431     {\@iiminipage{#1}\relax[s]}}
```

(End of definition for \@minipage.)

`\@iiminipage` Optional argument handling.

```
432 \def\@iiminipage#1[#2]{%
433   \ifnextchar[%
434     {\@iiminipage{#1}{#2}}%
435     {\@iiminipage{#1}{#2}[#1]}}
```

(End of definition for \@iiminipage.)

`\@iiminipage` Internal form of minipage.

```
436 \def\@iiminipage#1#2[#3]#4{%
437   \leavevmode
438   \@pboxswfalse
439   \setlength\@tempdima{#4}%
440   \def\@mpargs{#1}{#2}{#3}{#4}%
441   \setbox\@tempboxa\vbox\bgroup
442     \color@begingroup
443     \hsize\@tempdima
444     \textwidth\hsize \columnwidth\hsize
```

We check for nested minipages inside the box so that there is always a group resetting the switch even if the code does not use `\begin` to start the minipage.

```
445   \if@in@minipage@env
```

We only issue a warning if the outer minipage contained footnotes because that is the problematical case.

```
446     \ifvoid\@mpfootins\else
447       \@latex@warning{Nested minipage:
448         footnotes may be misplaced}%
449     \fi
450   \else
451     \@in@minipage@envtrue
452   \fi
```

```

453 \parboxrestore
454 \def\@mpfn{mpfootnote}\def\thempfn{\thempfootnote}\c@mpfootnote\z@
455 \let\@footnotetext\@mpfootnotetext
456 \let\@listdepth\@mplistdepth \@mplistdepth\z@
457 \@minipagerestore
458 \setminipage}

```

(End of definition for \@iiiminipage.)

`\@minipagerestore` Hook so that other styles can reset other commands in a minipage.

```

459 \let\@minipagerestore=\relax

```

(End of definition for \@minipagerestore.)

`\endminipage`

```

460 \def\endminipage{%
461 \par
462 \unskip
463 \ifvoid\@mpfootins\else
464 \vskip\skip\@mpfootins
465 \normalcolor
466 \footnoterule
467 \unvbox\@mpfootins
468 \fi
469 \@minipagefalse %% added 24 May 89
470 \color@endgroup
471 \egroup
472 \expandafter\@iiiparbox\@mpargs{\unvbox\@tempboxa}}

```

(End of definition for \endminipage.)

`\@mplistdepth` Versions of `\@listdepth` and `\footins` local to minipage.

```

\@mpfootins
473 \newcount\@mplistdepth
474 \newinsert\@mpfootins

```

(End of definition for \@mplistdepth and \@mpfootins.)

`\@mpfootnotetext` Minipage version of `\@footnotetext`.

Final `\strut` added 27 Mar 89, on suggestion by Don Hosek

```

475 </2ekernel>
476 <*2ekernel | latexrelease>
477 <latexrelease>\IncludeInRelease{2021/11/15}%
478 <latexrelease> \{\@mpfootnotetext\}{footnotetext tagging}%
479 \long\def\@mpfootnotetext#1{%
480 \global\setbox\@mpfootins\vbox{%
481 \unvbox\@mpfootins
482 \reset@font\footnotesize
483 \hsize\columnwidth
484 \parboxrestore
485 \def\@currentcounter{mpfootnote}%
486 \protected@edef\@currentlabel
487 {\csname p@mpfootnote\endcsname\@thefnmark}%
488 \color@begingroup
489 \makefnstext{%
490 \rule\z@\footnotesep\ignorespaces#1\@finalstrut\strutbox}%

```

```

491     \par
492     \color@endgroup}}
493 </2ekernel | latexrelease)
494 <latexrelease>\EndIncludeInRelease
495 <latexrelease>\IncludeInRelease{2021/06/01}%
496 <latexrelease>           {\@mpfootnotetext}{footnotetext tagging}%
497 <latexrelease>\long\def\@mpfootnotetext#1{%
498 <latexrelease>  \global\setbox\@mpfootins\vbox{%
499 <latexrelease>    \unvbox\@mpfootins
500 <latexrelease>    \reset@font\footnotesize
501 <latexrelease>    \hsize\columnwidth
502 <latexrelease>    \@parboxrestore
503 <latexrelease>    \protected@edef\@currentlabel
504 <latexrelease>      {\csname p@mpfootnote\endcsname\@thefnmark}%
505 <latexrelease>    \color@begingroup
506 <latexrelease>      \@makefntext{%
507 <latexrelease>        \rule\z@\footnotesep\ignorespaces#1\@finalstrut\strutbox}%
508 <latexrelease>    \par
509 <latexrelease>    \color@endgroup}}
510 <latexrelease>\EndIncludeInRelease
511 <latexrelease>\IncludeInRelease{0000/00/00}%
512 <latexrelease>           {\@mpfootnotetext}{footnotetext tagging}%
513 <latexrelease>
514 <latexrelease>\long\def\@mpfootnotetext#1{%
515 <latexrelease>  \global\setbox\@mpfootins\vbox{%
516 <latexrelease>    \unvbox\@mpfootins
517 <latexrelease>    \reset@font\footnotesize
518 <latexrelease>    \hsize\columnwidth
519 <latexrelease>    \@parboxrestore
520 <latexrelease>    \protected@edef\@currentlabel
521 <latexrelease>      {\csname p@mpfootnote\endcsname\@thefnmark}%
522 <latexrelease>    \color@begingroup
523 <latexrelease>      \@makefntext{%
524 <latexrelease>        \rule\z@\footnotesep\ignorespaces#1\@finalstrut\strutbox}%
525 <latexrelease>    \color@endgroup}}
526 <latexrelease>
527 <latexrelease>\EndIncludeInRelease
528 <*2ekernel)

(End of definition for \@mpfootnotetext.)

529 \newif\if@pboxsw

```

\rule Draw a rule of the specified size.

```

530 </2ekernel)
531 <latexrelease>\IncludeInRelease{2015/01/01}%
532 <latexrelease>           {\rule}{Make \rule robust}%
533 <*2ekernel | latexrelease)
534 \DeclareRobustCommand\rule{\@ifnextchar[\@rule{\@rule[\z@]}}%
535 </2ekernel | latexrelease)
536 <latexrelease>\EndIncludeInRelease
537 <latexrelease>\IncludeInRelease{0000/00/00}%
538 <latexrelease>           {\rule}{Make \rule robust}%
539 <latexrelease>\def\rule{\@ifnextchar[\@rule{\@rule[\z@]}}%

```

```

540 <latexrelease>\expandafter\let\csname rule \endcsname\@undefined
541 <latexrelease>\EndIncludeInRelease
542 <*2ekernel>

```

(End of definition for \rule.)

\@rule Internal form of \rule.

```

543 \def\@rule[#1]#2#3{%
544 \leavevmode
545 \hbox{%
546 \setlength\@tempdima{#1}%
547 \setlength\@tempdimb{#2}%
548 \setlength\@tempdimc{#3}%
549 \advance\@tempdimc\@tempdima
550 \vrule\@width\@tempdimb\@height\@tempdimc\@depth-\@tempdima}}

```

(End of definition for \@rule.)

\@@underline Saved primitive \underline.

```

551 \let\@@underline\underline

```

(End of definition for \@@underline.)

\underline L^AT_EX version works outside math.

```

552 \DeclareRobustCommand\underline[1]{%
553 \relax
554 \ifmmode\@@underline{#1}%
555 \else $\@@underline{\hbox{#1}}\m@th$\relax\fi

```

(End of definition for \underline.)

\raisebox Raise a box, and change its vertical dimensions.

```

556 </2ekernel>
557 <latexrelease>\IncludeInRelease{2015/01/01}%
558 <latexrelease> \{\raisebox}\{Make \raisebox robust}%
559 <*2ekernel | latexrelease>
560 \DeclareRobustCommand\raisebox[1]{%
561 \leavevmode
562 \@ifnextchar[{\@rsbox{#1}}{\@irsbox{#1}[]}}
563 </2ekernel | latexrelease>
564 <latexrelease>\EndIncludeInRelease
565 <latexrelease>\IncludeInRelease{0000/00/00}%
566 <latexrelease> \{\raisebox}\{Make \raisebox robust}%
567 <latexrelease>\def\raisebox#1{%
568 <latexrelease> \leavevmode
569 <latexrelease> \@ifnextchar[{\@rsbox{#1}}{\@irsbox{#1}[]}}
570 <latexrelease>\expandafter\let\csname raisebox \endcsname\@undefined
571 <latexrelease>\EndIncludeInRelease
572 <*2ekernel>

```

(End of definition for \raisebox.)

\@rsbox Optional argument handling.

```

573 \def\@rsbox#1[#2]{%
574 \@ifnextchar[{\@iirsbox{#1}[#2]}{\@irsbox{#1}[#2]}}

```


(End of definition for `\@rsbox`.)

`\@argrsbox` ...

(End of definition for `\@argrsbox`.)

`\@irsbox` Internal version of `\raisebox` (less than two optional args).

```
575 \long\def\@irsbox#1[#2]#3{%
576   \@begin@tempboxa\hbox{#3}%
577   \setlength\@tempdima{#1}%
578   \ifx\#2\else\setlength\@tempdimb{#2}\fi
579   \setbox\@tempboxa\hbox{\raise\@tempdima\box\@tempboxa}%
580   \ifx\#2\else\ht\@tempboxa\@tempdimb\fi
581   \box\@tempboxa
582   \@end@tempboxa}
```

(End of definition for `\@irsbox`.)

`\@iirsbox` Internal version of `\raisebox` (two optional args).

```
583 \long\def\@iirsbox#1[#2][#3]#4{%
584   \@begin@tempboxa\hbox{#4}%
585   \setlength\@tempdima{#1}%
586   \setlength\@tempdimb{#2}%
587   \setlength\dimen@{#3}%
588   \setbox\@tempboxa\hbox{\raise\@tempdima\box\@tempboxa}%
589   \ht\@tempboxa\@tempdimb
590   \dp\@tempboxa\dimen@
591   \box\@tempboxa
592   \@end@tempboxa}
```

(End of definition for `\@iirsbox`.)

`\@finalstrut` This macro adds a special strut the *depth* of the box given as #1, and height and width Opt. It is used for ensuring that the last line of a paragraph has the correct depth in ‘p’ columns of tables and in footnotes. In vertical mode nothing is done, as adding the strut (as done in 2.09) would start a new paragraph. It would be possible to inspect `\prevdepth` to check the depth of the just-completed paragraph, but we do not do that here. Actually we do even less now, skip the vmode test as it broke tabular ‘p’ columns.

The `\nobreak` was added (1995/10/31) to allow hyphenation of the final word of the paragraph.

In 2024 we changed the macro to account for vertical mode. In that case we use a strut produced with `\hrule` to avoid starting a new paragraph (resulting in spurious extra line) and also account for the `\prevdepth` of the previous line.

```
593 </2kernel>
594 <*2kernel | latexrelease>
595 <latexrelease>\IncludeInRelease{2024/06/01}%
596 <latexrelease>           {\@finalstrut}{final strut correction}%
597 \def\@finalstrut#1{%
598   \unskip
599   \ifhmode \nobreak
600   \else
```

If we are in vmode we now back up by a baseline.

```
601 \vskip-\baselineskip
602 \fi
```

Finally we unconditionally use `\vrule`.

```
603 \vrule\@width\z@\@height\z@\@depth\dp#1}
604 </2ekernel | latexrelease>
605 <latexrelease>\EndIncludeInRelease
606 <latexrelease>\IncludeInRelease{0000/00/00}%
607 <latexrelease> \@finalstrut}{final strut correction}%
608 <latexrelease>\def\@finalstrut#1{%
609 <latexrelease> \unskip\ifhmode\nobreak\fi
610 <latexrelease> \vrule\@width\z@\@height\z@\@depth\dp#1}
611 <latexrelease>
612 <latexrelease>\EndIncludeInRelease
613 <*2ekernel>
```

(End of definition for \@finalstrut.)

1.1 Some low-level constructs

The following commands are basically inherited from plain T_EX.

```
\leftline These macros place text on a full line either centred or left or right adjusted.
\rightline
\centerline 614 \def\@@line{\hb@xt@\hsize}
\@@line 615 \DeclareRobustCommand\leftline[1]{\@@line{#1\hss}}
616 \DeclareRobustCommand\rightline[1]{\@@line{\hss#1}}
617 \DeclareRobustCommand\centerline[1]{\@@line{\hss#1\hss}}
```

(End of definition for \leftline and others.)

```
\rlap These macros place text to the left or right of the current reference point without taking
\llap up space.
\clap 618 \DeclareRobustCommand\rlap[1]{\hb@xt@\z@{#1\hss}}
619 \DeclareRobustCommand\llap[1]{\hb@xt@\z@{\hss#1}}
```

And here is the version that centers, it was initially introduced by `mathtools`.

```
620 \DeclareRobustCommand\clap[1]{\hb@xt@\z@{\hss#1\hss}}
```

(End of definition for \rlap, \llap, and \clap.)

```
621 </2ekernel>
```

File 41

lfttab.dtx

1 Tabbing, Tabular and Array Environments

This section deals with ‘Lining It Up in Columns’. First the `tabbing` environment is defined, and then in second part, `tabular` together with its variants, `tabular*` and `array`.

Note that the `tabular` defined here is essentially the original L^AT_EX 2.09 version, not the extended version described in *The L^AT_EX Companion*. Use the `array` package to obtain the extended version.

1.1 tabbing

Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

`\dimen(\@firsttab + i)` = distance of tab stop `i` from left margin
0 <= `i` <= 15 (?).

`\dimen\@firsttab` is initialized to `\@totalleftmargin`, so it starts at the prevailing left margin.

`\@maxtab` = number of highest defined tab register
probably = `\@firsttab + 12`

`\@nxttabmar` = tab stop number of next line’s left margin

`\@curtabmar` = tab stop number of current line’s left margin

`\@curtab` = number of the current tab. At start of line,
it equals `\@curtabmar`

`\@hightab` = largest tab number currently defined.

`\@tabpush` = depth of `\pushtab`’s

`\box\@curline` = contents of current line, excluding left margin
skip, and excluding contents of current field

`\box\@curfield` = contents of current field

`@rjfield` = switch: T iff the last field of the line should
be right-justified at the right margin.

`\tabbingsep` = distance left by the `\` command between the
current position and the field that is
“left-shifted”.

UTILITY MACROS

`\@stopfield` : closes the current field

`\@addfield` : adds the current field to the current line.

`\@contfield` : continues the current field

`\@startfield` : begins the next field

`\@stopline` : closes the current line and outputs it

```

\@startline : starts the next line
\@ifatmargin : an \if that is true iff the current line.
                has width zero

\@startline ==
BEGIN
  \@curtabmar :=G \@nxttabmar
  \@curtab :=G \@curtabmar
  \box\@curline :=G null
  \@startfield
  \strut
END

\@stopline ==
BEGIN
  \unskip
  \@stopfield
  if @rjfield = T
    then @rjfield :=G F
      \@tempdima := \@totalleftmargin + \linewidth
      \hb@xt@ \@tempdima{\@itemfudge
                          \hskip \dimen\@curtabmar
                          \box\@curline
                          \hfil
                          \box\@curfield}
    else \@addfield
      \hbox {\@itemfudge
            \hskip \dimen\@curtabmar
            \box\@curline}
    fi
  END

\@startfield ==
BEGIN
  \box\@curfield :=G \hbox {
  END

\@stopfield ==
BEGIN
  }
  END

\@contfield ==
BEGIN
  \box\@curfield :=G \hbox { \unhbox\@currfield %%} brace matching
  END
\@addfield ==
BEGIN
  \box\@curline :=G \unbox\@curline * \unbox\@curfield
  END

```

```

\@ifatmargin ==
BEGIN
  if dim of box\@curline = 0pt then
  END

\tabbing ==
BEGIN
  \lineskip :=L 0pt
  \> == \@rtab
  \< == \@ltab
  \= == \@settab
  \+ == \@tabplus
  \- == \@tabminus
  \‘ == \@tabrj
  \’ == \@tablab
  \\ == BEGIN \@stopline \@startline END
  \\[DIST] == BEGIN
    \@stopline \vskip DIST \@startline\ignorespaces END
  \\* == BEGIN \@stopline \penalty 10000 \@startline END
  \\*[DIST] == BEGIN \@stopline \penalty 10000 \vskip DIST
    \@startline\ignorespaces END
  \@hightab := \@nxttabmar :=G \@firsttab
  \@tabpush :=G 0
  \dimen\@firsttab := \@totalleftmargin
  @rjfield :=G F
  \trivlist \item\relax
  if @minipage = F then \vskip \parskip fi
  \box\@tabfbox = \rlap{\indent\the\everypar}
    % note: \the\everypar sets @inlabel :=G F
  \@itemfudge == BEGIN \box\@tabfbox END
  \@startline
  \ignorespaces
  END

\@endtabbing ==
BEGIN
  \@stopline
  if \@tabpush > 0 then error message: "unmatched \poptabs" fi
  \endtrivlist
  END

\@rtab ==
BEGIN
  \@stopfield
  \@addfield
  if \@curtab < \@hightab
    then \@curtab :=G \@curtab + 1
    else error message "Undefined Tab" fi

```

```

\@tempdima := \dimen\@curtab - \dimen\@curtabmar
              - width of box \@curline
\box\@curline :=G \hbox{\unhbox\@curline + \hskip\@tempdima}
\@startfield
END

\@settab ==
BEGIN
  \@stopfield
  \@addfield
  if \@curtab < \@maxtab
    then \@curtab :=G \@curtab+1
    else error message: "Too many tabs"    fi
  if \@curtab > \@hightab
    then \@hightab :=L \@curtab    fi
  \dimen\@curtab :=L \dimen\@curtabmar + width of \box\@curline
  \@startfield
END

\@ltab ==
BEGIN
  \@ifatmargin
  then if \@curtabmar > \@firsttab
    then \@curtab :=G \@curtab - 1
        \@curtabmar :=G \@curtabmar - 1
    else error message "Too many untab"    fi
  else error message "Left tab in middle of line"
  fi
END

\@tabplus ==
BEGIN
  if \@nxttabmar < \@hightab
    then \@nxttabmar :=G \@nxttabmar+1
    else error message "Undefined tab"
  fi
END

\@tabminus ==
BEGIN
  if \@nxttabmar > \@firsttab
    then \@nxttabmar :=G \@nxttabmar-1
    else error message "Too many untab"
  fi
END

\@tabrj ==
BEGIN \@stopfield
  \@addfield
  @rjfield :=G T

```

```

        \@startfield
    END

\@tablab ==
    BEGIN \@stopfield
        \box\@curline G:= \hbox{\box\@curline %% 'G' added 17 Jun 86
                                \hskip - width of \box\@curline
                                \hskip -\tabbingsep
                                \box\@curline
                                \hskip \tabbingsep }

        \@startfield
    END

\pushtabs ==
    BEGIN
        \@stopfield
        \@tabpush :=G \@tabpush + 1
        \begingroup
        \@contfield
    END

\poptabs ==
    BEGIN
        \@stopfield
        if \@tabpush > 0
            then \endgroup
                \@tabpush :=G \@tabpush - 1
            else error message: "Too many \poptabs"
        fi
        \@contfield
    END

```

End of historical L^AT_EX 2.09 comments.

\a The accents \‘, \’ , and \= that have been redefined inside a tabbing environment can be called by typing \a‘, \a’ , and \a=. The macro \a is defined in ltoutenc.dtx.

(End of definition for \a.)

The ‘2ekernel’ code ensures that a \usepackage{autotabg} is essentially ignored if a ‘full’ format is being used that has picture mode already in the format.

```
1 <2ekernel>\expandafter\let\csname ver@autotabg.sty\endcsname\fmtversion
```

```

\@firsttab
  \@maxtab 2 <*2ekernel>
3 \newdimen\@gtempa
4 \chardef\@firsttab=\the\allocationnumber
5 \newdimen\@gtempa\newdimen\@gtempa\newdimen\@gtempa\newdimen\@gtempa
6 \newdimen\@gtempa\newdimen\@gtempa\newdimen\@gtempa\newdimen\@gtempa
7 \newdimen\@gtempa\newdimen\@gtempa\newdimen\@gtempa\newdimen\@gtempa
8 \newdimen\@gtempa
9 \chardef\@maxtab=\the\allocationnumber
10 \dimen\@firsttab=0pt

```

(End of definition for \@firsttab and \@maxtab.)

```
\@nxttabmar
\@curtabmar 11 \newcount\@nxttabmar
  \@curtab 12 \newcount\@curtabmar
  \@hightab 13 \newcount\@curtab
  \@tabpush 14 \newcount\@hightab
  15 \newcount\@tabpush
```

(End of definition for \@nxttabmar and others.)

```
\@curline
\@curfield 16 \newbox\@curline
\@tabfbox 17 \newbox\@curfield
  18 \newbox\@tabfbox
```

(End of definition for \@curline, \@curfield, and \@tabfbox.)

```
\if@rjfield
  19 \newif\if@rjfield
```

(End of definition for \if@rjfield.)

`\@startline` It is, in some sense, an error if the current margin tab setting is higher than the value of `\@hightab` (which is a local variable). That this is allowed is a fundamental design flaw which is not going to be corrected now.

```
20 \def\@startline{%
21   \ifnum \@nxttabmar >\@hightab
22     \@badtab
23     \global\@nxttabmar \@hightab
24     \fi
25     \global\@curtabmar \@nxttabmar
26     \global\@curtab \@curtabmar
27     \global\setbox\@curline \hbox {}%
28     \@startfield
29     \strut}
```

(End of definition for \@startline.)

```
\@stopline
30 \def\@stopline{%
31   \unskip
32   \@stopfield
33   \if@rjfield
34     \global\@rjfieldfalse
35     \@tempdima\@totalleftmargin
36     \advance\@tempdima\linewidth
37     \hb@xt@\@tempdima{%
38       \@itemfudge\hskip\dimen\@curtabmar
39       \box\@curline
40       \hfil
41       \box\@curfield}%
42   \else
43     \@addfield
44     \hbox{\@itemfudge\hskip\dimen\@curtabmar\box\@curline}%
45   \fi}
```


(End of definition for \@stopline.)

\@startfield

```
46 \def\@startfield{%
47   \global\setbox\@curfield\hbox\bgroup\color@begingroup}
```

(End of definition for \@startfield.)

\@stopfield

```
48 \def\@stopfield{%
49   \color@endgroup\egroup}
```

(End of definition for \@stopfield.)

\@contfield

```
50 \def\@contfield{%
51   \global\setbox\@curfield\hbox\bgroup\color@begingroup
52   \unhbox\@curfield}
```

(End of definition for \@contfield.)

\@addfield

```
53 \def\@addfield{\global\setbox\@curline\hbox{\unhbox
54   \@curline\unhbox\@curfield}}
```

(End of definition for \@addfield.)

\@ifatmargin

```
55 \def\@ifatmargin{\ifdim \wd\@curline =\z@}
```

(End of definition for \@ifatmargin.)

\@tabcr

```
56 \protected\def\@tabcr{\@stopline \@ifstar{\penalty \@M \@xtabcr}\@xtabcr}
```

(End of definition for \@tabcr.)

\@xtabcr

```
57 \def\@xtabcr{\@ifnextchar[\@itabcr{\@startline\ignorespaces}}
```

(End of definition for \@xtabcr.)

\@itabcr

```
58 </2ekernel>
59 < *2ekernel | latexrelease >
60 < latexrelease >\IncludeInRelease{2020/10/01}%
61 < latexrelease >           {\@itabcr}{Tabbing calc syntax}%
62 \def\@itabcr[#1]{\@vspace@calcify{#1}\@startline\ignorespaces}
63 </2ekernel | latexrelease >
64 < latexrelease >\EndIncludeInRelease
65 < latexrelease >\IncludeInRelease{0000/00/00}%
66 < latexrelease >           {\@itabcr}{Tabbing calc syntax}%
67 < latexrelease >
68 < latexrelease >\def\@itabcr[#1]{\vskip #1\@startline\ignorespaces}
69 < latexrelease >\EndIncludeInRelease
70 < *2ekernel >
```

tabbing (env.) We use \relax to prevent \item from scanning too far.

```
\tabbing 71 \def\tabbing{\lineskip \z@skip\let\>\@rtab\let\<\@ltab\let\=\@settab
72 \let\+\@tabplus\let\-\@tabminus\let\'\@tabrj\let\'\@tablab
73 \let\=\@tabcr
74 \@hightab\@firsttab
75 \global\@nxttabmar\@firsttab
76 \dimen\@firsttab\@totalleftmargin
77 \global\@tabpush\z@ \global\@rjfieldfalse
78 \trivlist \item\relax
79 \if@minipage\else\vskip\parskip\fi
80 \setbox\@tabfbox\hbox{%
81 \rlap{\hskip\@totalleftmargin\indent\the\everypar}}%
82 \def\@itemfudge{\box\@tabfbox}%
83 \@startline\ignorespaces}
```

```
\endtabbing 84 \def\endtabbing{%
85 \@stopline\ifnum\@tabpush >\z@ \@badpoptabs \fi\endtrivlist}
```

Omitted \global added to \@rtab 17 Jun 86

```
\@rtab 86 \def\@rtab{\@stopfield\@addfield\ifnum \@curtab<\@hightab
87 \global\advance\@curtab \@ne \else\@badtab\fi
88 \@tempdima\dimen\@curtab
89 \advance\@tempdima -\dimen\@curtabmar
90 \advance\@tempdima -\wd\@curline
91 \global\setbox\@curline\hbox{\unhbox\@curline\hskip\@tempdima}%
92 \@startfield\ignorespaces}
```

```
\@settab 93 \def\@settab{\@stopfield\@addfield
94 \ifnum \@curtab <\@maxtab
95 \ifnum\@curtab =\@hightab
96 \advance\@hightab \@ne
97 \fi
98 \global\advance\@curtab \@ne
99 \else
100 \@latexerror{Tab overflow}\@ehd
101 \fi
102 \dimen\@curtab \dimen\@curtabmar
103 \advance\dimen\@curtab \wd\@curline
104 \@startfield
105 \ignorespaces}
```

```
\@ltab 106 \def\@ltab{\@ifatmargin\ifnum\@curtabmar >\@firsttab
107 \global\advance\@curtab \m@ne \global\advance\@curtabmar\m@ne\else
108 \@badtab\fi\else
109 \@latexerror{\string\<\space in mid line}\@ehd\fi\ignorespaces}
```

```
\@tabplus 110 \def\@tabplus{%
111 \ifnum\@nxttabmar<\@hightab
```

```

112     \global\advance\@nxttabmar\@ne
113     \else
114     \@badtab
115     \fi
116     \ignorespaces}

\@tabminus 117 \def\@tabminus{%
118     \ifnum\@nxttabmar>\@firsttab
119     \global\advance\@nxttabmar\m@ne
120     \else
121     \@badtab
122     \fi
123     \ignorespaces}

\@tabrj 124 \def\@tabrj{%
125     \@stopfield\@addfield\global\@rjfieldtrue\@startfield\ignorespaces}

\setbox\@curline made \global in \@tablab. 17 Jun 86

\@tablab 126 \def\@tablab{%
127     \@stopfield
128     \global\setbox\@curline\hbox{%
129     \box\@curline
130     \hskip-\wd\@curfield \hskip-\tabbingsep
131     \box\@curfield
132     \hskip\tabbingsep}%
133     \@startfield
134     \ignorespaces}

135 </2ekernel>
136 <*2ekernel | latexrelease>
137 <latexrelease>\IncludeInRelease{2019/10/01}%
138 <latexrelease>          {\pushtabs}{Make commands robust}%

\pushtabs 139 \DeclareRobustCommand\pushtabs{%
140     \@stopfield\@addfield\global\advance\@tabpush \@ne \begingroup
141     \@contfield}

\poptabs  It is, in some sense, an error if, after the endgroup, the current tab setting is higher
          than the new value of \@hightab (which is a local variable). That this is allowed is a
          fundamental design flaw which is not going to be corrected now.

142 \DeclareRobustCommand\poptabs{\@stopfield\@addfield
143     \ifnum \@tabpush >\z@
144     \endgroup
145     \global\advance\@tabpush \m@ne
146     \ifnum \@curtab >\@hightab
147     \global \@curtab \@hightab
148     \@badtab
149     \fi
150     \else
151     \@badpoptabs
152     \fi
153     \@contfield}

```

```

154 \DeclareRobustCommand\kill{\@stopfield\@startline\ignorespaces}
(End of definition for \@itabcr and others.)
155 </2ekernel | latexrelease>
156 <latexrelease>\EndIncludeInRelease
157 <latexrelease>\IncludeInRelease{0000/00/00}%
158 <latexrelease>          {\pushtabs}{Make commands robust}%
159 <latexrelease>
160 <latexrelease>\kernel@make@fragile\pushtabs
161 <latexrelease>\kernel@make@fragile\poptabs
162 <latexrelease>\kernel@make@fragile\kill
163 <latexrelease>
164 <latexrelease>\EndIncludeInRelease
165 <*2ekernel>

```

`\tabbingsep`

```

166 \newdimen\tabbingsep
(End of definition for \tabbingsep.)

```

1.2 array and tabular environments

Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

ARRAY PARAMETERS:

`\arraycolsep`
: half the width separating columns in an array environment

`\tabcolsep`
: half the width separating columns in a tabular environment

`\arrayrulewidth`
: width of rules

`\doublerulesep`
: space between adjacent rules in array or tabular

`\arraystretch`
: line spacing in array and tabular environments is done by placing a strut in every row of height and depth `\arraystretch` times the height and depth of the strut produced by an ordinary `\strut` command.

PREAMBLE:

The PREAMBLE argument of an array or tabular environment can contain the following:

`l,r,c` : indicate where entry is to be placed.

`|` : for vertical rule

`@{EXP}` : inserts the text EXP in every column.
`\arraycolsep` or `\tabcolsep` spacing is suppressed.

`*{N}{PRE}` : equivalent to writing N copies of PRE in the preamble.
PRE may contain `*{N'}{EXP'}` expressions.

`p{LEN}` : makes entry in parbox of width LEN.

SPECIAL ARRAY COMMANDS:

`\multicolumn{N}{FORMAT}{ITEM}` : replaces the next N column items by ITEM, formatted according to FORMAT. FORMAT should contain at most one l,r or c. If it contains none, then ITEM is ignored.

`\vline` : draws a vertical line the height of the current row. May appear in an array element entry.

`\hline` : draws a horizontal line between rows. Must appear either before the first entry (to appear above the first row) or right after a `\\` command. If followed by another `\hline`, then adds a `\vskip` of `\doublerulesep`.

`\cline{i-j}` : draws horizontal lines between rows covering columns i through j, inclusive. Multiple commands may follow one another to provide lines covering several disjoint columns

`\extracolsep{WIDTH}` : for use inside an @ in the preamble. Causes a WIDTH space to be added between columns for the rest of the columns. This is in addition to the ordinary intercolumn space.

```

\array ==
  BEGIN
    \@acol    == \@arrayacol
    \@classz  == \@arrayclassz
    \@classiv == \@arrayclassiv
    \\        == \@arraycr
    \@halignto == NULL
    \@tabarray
  END

\endarray{NAME} == BEGIN \crrc }} END

\tabular ==
  BEGIN
    \@halignto == NULL
    \@tabular
  END

\tabular*{WIDTH} ==
  BEGIN
    \@halignto == to WIDTH
    \@tabular
  END

\@tabular ==
  BEGIN
    \leavevmode
    \hbox { $
      \@acol    == \@tabacol

```

```

        \@classz == \@tabclassz
        \@classiv == \@tabclassiv
        \\\ == \@tabularcr
        \@tabarray
    END

\endtabular == BEGIN \crrc}} $} END

\@tabarray == if next char = [ then \@array else \@array[c] fi

\@array[POS]{PREAMBLE} ==
    BEGIN
        define \@arstrutbox to make \@arstrut produce strut of height
            and depth \arraystretch times the height and
            depth of a normal strut.
        \@mkpream{PREAMBLE}
        \@preamble == \halign \@halignto {\tabskip=0pt\@arstrut
            eval{\@preamble}\tabskip = 0pt\cr %%}
        \@startpbox == \@@startpbox
        \@endpbox == \@@endpbox
        if POS = t then \vtop
            else if POS = b then \vbox
                else \center
            fi
        fi
        {
            \par ==L {} % changed 92/09/18
            \@sharp == #
            \protect == \relax
            \lineskip :=L 0pt
            \baselineskip :=L 0pt
            \@preamble
        }
    END

\@arraycr ==
    BEGIN
        $ %% Prevents extra space at end of row's last entry.
        if next char = [
            then \@argarraycr
            else $ \cr %% Needed to balance $
        }
    END

\@argarraycr[LENGTH] ==
    BEGIN
        $ %% Needed to balance $ of \@arraycr
        if LENGTH > 0
            then \@tempdima := depth of \@arstrutbox + LENGTH
                \vrule height 0pt width 0pt depth \@tempdima
                \cr
            else \cr \noalign{\vskip LENGTH}
        }
    END

```

END

`\@tabularcr` and `\@argtabularcr` same as `\@arraycr` and `\@argarraycr`
except without the extra `$`'s.

End of historical L^AT_EX 2.09 comments.

`\extracolsep` This command needs to expand during the tabular preamble construction so can't be robust.

```
167 \def\extracolsep#1{\tabskip #1\relax}
```

(End of definition for \extracolsep.)

`\array`~~`\array`~~

```
168 \def\array{\let\@acol\@arrayacol \let\@classz\@arrayclassz
```

```
169 \let\@classiv\@arrayclassiv
```

```
170 \let\\\@arraycr\let\@halignto\@empty\@tabarray}
```

(End of definition for \array.)

`\endarray`

```
\endtabular 171 \def\endarray{\crrc\egroup\egroup}
```

```
\endtabular* 172 \def\endtabular{\crrc\egroup\egroup $\egroup}
```

```
173 \expandafter \let \csname endtabular*\endcsname = \endtabular
```

(End of definition for \endarray, \endtabular, and \endtabular.)*

`\tabular`~~`\tabular`~~

```
174 \def\tabular{\let\@halignto\@empty\@tabular}
```

(End of definition for \tabular.)

`\tabular*` Note that the change to use `\setlength` slightly alters the timing of the expansion and use of the length in `#1` but this is very unlikely to have any practical effect.

```
175 \@namedef\tabular*#1{%
```

```
176 \setlength\dimen@{#1}%
```

```
177 \edef\@halignto{to\the\dimen@}\@tabular}
```

(End of definition for \tabular.)*

`\@tabular`

```
178 \def\@tabular{\leavevmode \hbox \bgroup $\let\@acol\@tabacol
```

```
179 \let\@classz\@tabclassz
```

```
180 \let\@classiv\@tabclassiv \let\\\@tabularcr\@tabarray}
```

(End of definition for \@tabular.)

`\@tabarray` RmS 91/11/04 added `\m@th`.

```
181 \def\@tabarray{\m@th\@ifnextchar[\@array{\@array[c]}}
```

(End of definition for \@tabarray.)

RmS 1993/11/03 changed `\halign` to `\ialign` and removed superfluous `\tabskip` assignment

`\@array`

```
182 \def\@array[#1]#2{%
```

```
183 \if #1t\vtop \else \if#1b\vbox \else \vcenter \fi\fi
```

184 `\bgroup`

This next bit of code sets up the strut and then builds the `halign` and its preamble according to the specification in the second argument.

This code has been moved inside the box. A side effect of this has been to expose what was a buglet in the previous version: since the `\@arstrut` below is expanded and contains an `\ifmode` then it could produce an unnecessary extra box in every row, thus wasting ‘lots of’ main memory.

```
185 \setbox\@arstrutbox\hbox{%
186   \vrule \@height\arraystretch\ht\strutbox
187         \@depth\arraystretch \dp\strutbox
188         \@width\z@}%
189 \@mkpream{#2}%
190 \edef\@preamble{%
191   \ialign \noexpand\@halignto
192     \bgroup \@arstrut \@preamble \tabskip\z@skip \cr}%
```

That is the end of setting up the preamble; now we reset things before executing the `halign` built-up in `\@preamble`. The restorations could be done by introducing an extra group, thus saving tokens.

```
193 \let\@startpbox\@startpbox \let\@endpbox\@endpbox
194 \let\tabularnewline\%
195 \let\par\@empty
196 \let\@sharp#\%
197 \set@typeset@protect
198 \lineskip\z@skip\baselineskip\z@skip
```

If the parsing of the preamble goes wrong there may be some characters left which $\text{T}_{\text{E}}\text{X}$ then tries to typeset, i.e., we would be in horizontal mode. That would produce an endless loop because the `\halign` expects vertical mode thus issues a `\par` but that is a no-op at this point. So we better test this case issue some error message and make a crude recovery by ending that horizontal mode with force. A better fix would be to ensure that we never pick up more than a single character token (not done).

```
199 \ifhmode \@preamerr\z@ \@par\fi
200 \@preamble}
```

(End of definition for \@array.)

`\@arraycr` Array version of `\%`.

```
201 \protected\def\@arraycr{%
202   ${\ifnum0='}\fi\@ifstar\@xarraycr\@arraycr}
```

(End of definition for \@arraycr.)

`\@arraycr`

```
203 \def\@xarraycr{\@ifnextchar[\@argarraycr{\ifnum0='{}\fi}\cr}}
```

(End of definition for \@arraycr.)

`\@argarraycr`

```
204 \def\@argarraycr[#1]{%
205   \ifnum0='{}\fi\ifdim #1>\z@ \@xargarraycr{#1}\else
206   \@yargarraycr{#1}\fi}
```

(End of definition for \@argarraycr.)


```

\tabularnewline Tabular version of \\.
207 \let\tabularnewline\relax

(End of definition for \tabularnewline.)

\@tabularcr
208 \protected\def\@tabularcr{%
209   {\ifnum0='}\fi\@ifstar\@xtabularcr\@xtabularcr}

(End of definition for \@tabularcr.)

\@xtabularcr
210 \def\@xtabularcr{\@ifnextchar[\@argtabularcr{\ifnum0='{}\fi}\cr}}

(End of definition for \@xtabularcr.)

\@argtabularcr
211 \def\@argtabularcr[#1]{%
212   \ifnum0='{}\fi}%
213   \ifdim #1>\z@
214     \unskip\@xargarraycr{#1}%
215   \else
216     \@yargarraycr{#1}%
217   \fi}

(End of definition for \@argtabularcr.)

\@xargarraycr
218 \def\@xargarraycr#1{\@tempdima #1\advance\@tempdima \dp \@arstrutbox
219   \vrule \@height\z@ \@depth\@tempdima \@width\z@ \cr}

(End of definition for \@xargarraycr.)

\@yargarraycr
220 \langle/2ekernel\rangle
221 \langle*2ekernel | latexrelease\rangle
222 \langlelatexrelease\rangle\IncludeInRelease{2020/10/01}%
223 \langlelatexrelease\rangle          {\@yargarraycr}{tabular support calc syntax}%
224 \def\@yargarraycr#1{\cr\noalign{\@vspace@calcify{#1}}\cr}
225 \langle/2ekernel | latexrelease\rangle
226 \langlelatexrelease\rangle\EndIncludeInRelease

227 \langlelatexrelease\rangle\IncludeInRelease{0000/00/00}%
228 \langlelatexrelease\rangle          {\@yargarraycr}{tabular support calc syntax}%
229 \langlelatexrelease\rangle
230 \langlelatexrelease\rangle\def\@yargarraycr#1{\cr\noalign{\vskip #1}}
231 \langlelatexrelease\rangle\EndIncludeInRelease
232 \langle*2ekernel\rangle

(End of definition for \@yargarraycr.)

```

`\multicolumn` *Historical L^AT_EX 2.09 comments (not necessarily accurate any more):*

```

\multicolumn{NUMBER}{FORMAT}{ITEM} ==
BEGIN
\multispan{NUMBER}
\begingroup
\@addamp == null
\@mkpream{FORMAT}
\@sharp == ITEM
\protect == \relax
\@startpbox == \@@startpbox
\@endpbox == \@@endpbox
\@arstrut
\@preamble
\endgroup
END

```

End of historical L^AT_EX 2.09 comments.

The command `\def\@addamp{}` was removed from `\multicolumn` on 6 Dec 86 because it caused embedded array environments not to work. I think that it was included originally to prevent an error message if the 2nd argument to the `\multicolumn` command had two column specifiers.

8 Feb 89 — `\hbox{}` added after `\@preamble` to correct bug that occurred if `\multicolumn` preceded `\[D]` with $D > 0$, caused by `\[]` command doing an `\unskip`, which removed `\tabcolsep` glue inserted by `\multicolumn`.

This has been made long so that, for example, a p-column can contain multiple paragraphs; maybe the arguments of @-expressions should also be able to contain multiple paragraphs.

```

233 \long\def\multicolumn#1#2#3{\multispan{#1}\begingroup
234   \@mkpream{#2}%
235   \def\@sharp{#3}\set@typeset@protect
236   \let\@startpbox\@@startpbox\let\@endpbox\@@endpbox
237   \@arstrut \@preamble\hbox{}\endgroup\ignorespaces}

```

(End of definition for \multicolumn.)

Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

Codes for classes and character numbers of array, tabular and multicolumn arguments.

Character	Class	Number
c	0	0
l	0	1
r	0	2
l	1	-
@	2	-
p	3	-
{@-exp}	4	-
{p-arg}	5	-

`\@testpach \foo` : expands `\foo`, which should be an array parameter

token, and sets `\@chclass` and `\@chnum` to its class and number. Uses `\@lastchclass` to distinguish 4 and 5

Preamble error codes

0: 'illegal character'
 1: 'Missing @-exp'
 2: 'Missing p-arg'

```
\@addamp ==
  BEGIN if \@firstamp = true then \@firstamp := false
        else &                                fi
  END
```

```
\@mkpream TOKENLIST ==
  BEGIN
    \@firstamp := T
    \@lastchclass := 6
    \@preamble == null
    \@sharp == \relax
    \@protect == BEGIN \noexpand\protect\noexpand END
    \@startpbox == \relax
    \@endpbox == \relax
    \@expast{TOKENLIST}
    for \@nextchar := expand(\reserved@a)
      do \@testpach{\@nextchar}
        case of \@chclass
          0 -> \@classz
          1 -> \@classi
          ...
          5 -> \@classv
        end case
        \@lastchclass := \@chclass
      od
    case of \@lastchclass
      0 -> \hskip \arraycolsep           % lrc
      1 ->                                % l
      2 -> \@preamerr1 % 'Missing @-exp' % @
      3 -> \@preamerr2 % 'Missing p-arg' % p
      4 ->                                % @-exp
      5 -> \hskip \arraycolsep           % p-exp
    end case
  END
```

```
\@arrayclassz ==
  BEGIN
    \@preamble := \@preamble *
    case of \@lastchclass
      0 -> \hskip \arraycolsep \@addamp \hskip \arraycolsep
      1 -> \@addamp \hskip \arraycolsep
      2 -> % impossible
```

```

        3 -> % impossible
        4 -> \@addamp
        5 -> \hskip \arraycolsep \@addamp \hskip \arraycolsep
        6 -> \@addamp \hskip \arraycolsep
    end case
* case of \@chnum
    0 -> \hfil$\relax\@sharp$\hfil
    1 -> $\relax\@sharp$\hfil
    2 -> \hfil$\relax\@sharp$
    end case
END

\@tabclassz == similar to \@arrayclassz

\@classi ==
BEGIN
    \@preamble := \@preamble *
        case of \@lastchclass
            0 -> \hskip \arraycolsep \@arrayrule
            1 -> \hskip \doublerulesep \@arrayrule
            2 -> % impossible
            3 -> % impossible
            4 -> \@arrayrule
            5 -> \hskip \arraycolsep \@arrayrule
            6 -> \@arrayrule
        end case
END

\@classii ==
BEGIN
    \@preamble := \@preamble *
        case of \@lastchclass
            0 ->
            1 -> \hskip .5\arrayrulewidth
            2 -> % impossible
            else ->
        end case
END

\@classiii ==
BEGIN
    \@preamble := \@preamble *
        case of \@lastchclass
            0 -> \hskip \arraycolsep \@addamp \hskip \arraycolsep
            1 -> \@addamp \hskip \arraycolsep
            2 -> % impossible
            3 -> % impossible
            4 -> \@addamp
            5 -> \hskip \arraycolsep \@addamp \hskip \arraycolsep
            6 -> \@addamp \hskip \arraycolsep

```

```

                                end case
END

\@arrayclassiv ==
    BEGIN \@preamble := \@preamble * $ \@nextchar$ END

\@tabclassiv == same as \@arrayclassiv except without the $ ... $

\@classv ==
    BEGIN
        \@preamble :=
            \@preamble * \@startpbox{\@nextchar}\ignorespaces\@sharp
                        \@endpbox
    END

\@expast{S}:
    Sets \reserved@a := S with all instances of *{N}{STRING}
    replaced by N copies of STRING, where N > 0. An *
    appearing inside braces is ignored, but *-expressions
    inside STRING are expanded, so nested *-expressions are
    handled properly.

\@expast{S} == BEGIN \@xexpast S *0x\@@ END

\@xexpast S1 *{N}{S2} S3 \@@ ==
    BEGIN
        \reserved@a := S1
        \@tempcnta := N
        if \@tempcnta > 0
            then while \@tempcnta > 0 do \reserved@a := \reserved@a S2
                \@tempcnta := \@tempcnta - 1 od
                \reserved@b == \@xexpast
            else \reserved@b == \@xexnoop
            fi
        \expandafter \reserved@b \reserved@a S3 \@@
    END
End of historical LATEX 2.09 comments.

\@xexnoop
238 \def\@xexnoop #1\@@{ }
(End of definition for \@xexnoop.)

\@expast
239 \def\@expast#1{\@xexpast #1*0x\@@}
(End of definition for \@expast.)

```

`\@xexpast`

```
240 \def\@xexpast#1*#2#3#4\@@{%
241   \edef\reserved@a{#1}%
242   \@tempcnta#2\relax
243   \ifnum\@tempcnta>\z@
244     \@whilenum\@tempcnta>\z@\do
245       {\edef\reserved@a{\reserved@a#3}\advance\@tempcnta \m@ne}%
246     \let\reserved@b\@xexpast
247   \else
248     \let\reserved@b\@xexnoop
249   \fi
250   \expandafter\reserved@b\reserved@a #4\@@}
```

(End of definition for \@xexpast.)

`\if@firstamp`

`\@addamp`

```
251 \newif\if@firstamp
252 \def\@addamp{%
253   \if@firstamp
254     \@firstampfalse
255   \else
256     \edef\@preamble{\@preamble &}%
257   \fi}
```

(End of definition for \if@firstamp and \@addamp.)

`\@arrayacol`

`\@tabacol`

`\@ampacol`

`\@acolampacol`

```
258 \def\@arrayacol{\edef\@preamble{\@preamble \hskip \arraycolsep}}
259 \def\@tabacol{\edef\@preamble{\@preamble \hskip \tabcolsep}}
260 \def\@ampacol{\@addamp \@acol}
261 \def\@acolampacol{\@acol\@addamp\@acol}
```

(End of definition for \@arrayacol and others.)

`\@mkpream`

```
262 \def\@mkpream#1{\@firstampttrue\@lastchclass6
263   \let\@preamble\@empty
264   \let\protect\@unexpandable@protect
265   \let\@sharp\relax
266   \let\@startpbox\relax\let\@endpbox\relax
267   \@expast{#1}%
268   \expandafter\@tfor \expandafter
269     \@nextchar \expandafter:\expandafter=\reserved@a\do
270     {\@testpach\@nextchar
271     \ifcase \@chclass \@classz \or \@classi \or \@classii \or \@classiii
272     \or \@classiv \or\@classv \fi\@lastchclass\@chclass}%
273   \ifcase \@lastchclass \@acol
274     \or \or \@preamerr \@ne\or \@preamerr \tw@\or \or \@acol \fi}
```

(End of definition for \@mkpream.)

`\@arrayclassz`

```
275 \def\@arrayclassz{\ifcase \@lastchclass \@acolampacol \or \@ampacol \or
276 \or \or \@addamp \or
277 \@acolampacol \or \@firstampfalse \@acol \fi
278 \edef\@preamble{\@preamble
279 \ifcase \@chnum
280 \hfil$\relax\@sharp$\hfil \or $\relax\@sharp$\hfil
281 \or \hfil$\relax\@sharp$\fi}}
```

(End of definition for \@arrayclassz.)

`\@tabclassz` RmS 91/08/14 inserted extra braces around entry for NFSS

```
282 \def\@tabclassz{%
283 \ifcase\@lastchclass
284 \@acolampacol
285 \or
286 \@ampacol
287 \or
288 \or
289 \or
290 \@addamp
291 \or
292 \@acolampacol
293 \or
294 \@firstampfalse\@acol
295 \fi
296 \edef\@preamble{%
297 \@preamble{%
298 \ifcase\@chnum
299 \hfil
300 \hskip1sp%
301 \ignorespaces\@sharp\unskip\hfil
302 \or
303 \hskip1sp\ignorespaces\@sharp\unskip\hfil
304 \or
305 \hfil\hskip1sp\ignorespaces\@sharp\unskip
306 \fi}}}
```

(End of definition for \@tabclassz.)

`\@classi`

```
307 \def\@classi{%
308 \ifcase\@lastchclass
309 \@acol\@arrayrule
310 \or
311 \@addtopreamble{\hskip \doublerulesep}\@arrayrule
312 \or
313 \or
314 \or
315 \@arrayrule
316 \or
317 \@acol\@arrayrule
318 \or
```

```

319     \@arrayrule
320     \fi}

(End of definition for \@classi.)

\@classii
321 \def\@classii{%
322   \ifcase\@lastchclass
323   \or
324   \@addtopreamble{\hskip .5\arrayrulewidth}%
325   \fi}

(End of definition for \@classii.)

\@classiii
326 \def\@classiii{\ifcase \@lastchclass \@acolampacol \or
327   \@addamp\@acol \or
328   \or \or \@addamp \or
329   \@acolampacol \or \@ampacol \fi}

(End of definition for \@classiii.)

\@tabclassiv
330 \def\@tabclassiv{\@addtopreamble\@nextchar}

(End of definition for \@tabclassiv.)

\@arrayclassiv
331 \def\@arrayclassiv{\@addtopreamble{\@nextchar$}}

(End of definition for \@arrayclassiv.)

\@classv
332 \def\@classv{\@addtopreamble{\@startpbox{\@nextchar}\ignorespaces
333   \@sharp\@endpbox}}

(End of definition for \@classv.)

\@addtopreamble
334 \def\@addtopreamble#1{\edef\@preamble{\@preamble #1}}

(End of definition for \@addtopreamble.)

\@chclass
\@lastchclass
\@chnum
335 \newcount\@chclass
336 \newcount\@lastchclass
337 \newcount\@chnum

(End of definition for \@chclass, \@lastchclass, and \@chnum.)

\arraycolsep
\@tabcolsep
\arrayrulewidth
\doublerulesep
338 \newdimen\arraycolsep
339 \newdimen\@tabcolsep
340 \newdimen\arrayrulewidth
341 \newdimen\doublerulesep

```


(End of definition for `\arraycolsep` and others.)

`\arraystretch`

```
342 \def\arraystretch{1} % Default value.
```

(End of definition for `\arraystretch`.)

`\@arstrutbox`

`\@arstrut`

```
343 \newbox\@arstrutbox
```

```
344 \def\@arstrut{%
```

```
345 \relax\ifmode\copy\@arstrutbox\else\unhcopy\@arstrutbox\fi}
```

(End of definition for `\@arstrutbox` and `\@arstrut`.)

`\@arrayrule`

```
346 \def\@arrayrule{\@addtopreamble{\hskip -.5\arrayrulewidth
```

```
347 \vrule \@width \arrayrulewidth\hskip -.5\arrayrulewidth}}
```

(End of definition for `\@arrayrule`.)

`\@testpach`

```
348 \def\@testpach#1{\@chclass \ifnum \@lastchclass=\tw@ 4 \else
```

```
349 \ifnum \@lastchclass=3 5 \else
```

```
350 \z@ \if #1c\@chnum \z@ \else
```

```
351 \if #1l\@chnum \@ne \else
```

```
352 \if #1r\@chnum \tw@ \else
```

```
353 \@chclass \if #1|\@ne \else
```

```
354 \if #1@\tw@ \else
```

```
355 \if #1p3 \else \z@ \@preamerr 0\fi
```

```
356 \fi \fi \fi \fi \fi \fi
```

```
357 \fi}
```

(End of definition for `\@testpach`.)

`\hline`

```
358 \def\hline{%
```

```
359 \noalign{\ifnum0='}\fi\hrule \@height \arrayrulewidth \futurelet
```

```
360 \reserved@a\@xhline}
```

(End of definition for `\hline`.)

`\@xhline`

```
361 \def\@xhline{\ifx\reserved@a\hline
```

```
362 \vskip\doublerulesep
```

Measure from the middle of the rules.

```
363 \vskip-\arrayrulewidth
```

```
364 \fi
```

```
365 \ifnum0='{\fi}}
```

(End of definition for `\@xhline`.)

`\vline`

```
366 \def\vline{\vrule \@width \arrayrulewidth}
```

(End of definition for `\vline`.)

`\cline` The old L^AT_EX2.09 implementation of `\cline` used up quite a lot of memory and two precious count registers. This new (1995/09/14) implementation does not use any count registers. It is coded in a way that depends heavily on the definition of `\multispan` so that command has been moved here from the file `ltpplain.dtx`.

These counters are no longer declared.

```
\newcount\@cla
```

```
\newcount\@clb
```

```
367 \def\cline#1{\@cline#1\@nil}
```

```
368 \def\@cline#1-#2\@nil{%
```

```
369 \omit
```

Use the counter from `\multispan`.

```
370 \@multicnt#1%
```

```
371 \advance\@multispan\m@ne
```

```
372 \ifnum\@multicnt=\@ne\@firstofone{&\omit}\fi
```

```
373 \@multicnt#2%
```

```
374 \advance\@multicnt-#1%
```

```
375 \advance\@multispan\@ne
```

The original had `\unskip` at this point, but how could a skip get here ???

```
376 \leaders\hrule\@height\arrayrulewidth\hfill
```

```
377 \cr
```

This is back spacing is fairly horrible, but it is what happened in the old version... An alternative would be to make `\cline` look ahead for a following `\cline` as does `\hline`. This would alter the spacing in existing documents so keep the old version in the kernel. Perhaps a package should do this differently.

```
378 \noalign{\vskip-\arrayrulewidth}}
```

(End of definition for `\cline` and `\@cline`.)

`\mscount` The `\mscount` counter is no longer declared, saving a csname and a register. It is declared in compatibility mode.

(End of definition for `\mscount`.)

`\multispan` Modify `\multispan` slightly from its plain T_EX definition to allow more efficient code sharing with `\multicolumn`. Also share a count register with `\multipt`.

```
\@multispan \sp@n 379 \def\multispan{\omit\@multispan}
```

```
380 \def\@multispan#1{%
```

```
381 \@multicnt#1\relax
```

```
382 \loop\ifnum\@multicnt>\@ne \sp@n\repeat}
```

```
383 \def\sp@n{\span\omit\advance\@multicnt\m@ne}
```

(End of definition for `\multispan`, `\@multispan`, and `\sp@n`.)

`\@startpbox` Helper macros for ‘p’ columns.

```
\@endpbox \startpbox{<width>} text \egroup is essentially \parbox{<width>}{<text>}
```

`\@endpbox` is essentially `\unskip \strut \par \egroup\hfil` (Changed 14 Jan 89) (changed again 1994/05/13)

```
384 \def\@startpbox#1{\vtop\bgroup \setlength\hsize{#1}\@arrayparboxrestore}
```

```
385 \def\@endpbox{\@finalstrut\@arstrutbox\par\egroup\hfil}
```

14 Jan 89: Def of \@endpbox changed from
`\def\@endpbox{\par\vskip\dp\@arstrutbox\egroup\hfil}`
so vertical spacing works out right if the last line of a 'p' entry has a descender.

(End of definition for \@startpbox and \@endpbox.)

```
\@@startpbox
```

```
\@@endpbox
```

```
386 \let\@@startpbox=\@startpbox
```

```
387 \let\@@endpbox=\@endpbox
```

(End of definition for \@@startpbox and \@@endpbox.)

```
388 </2kernel>
```

File 42

ltpictur.dtx

1 Picture Mode

Picture mode commands. In addition to the commands available in L^AT_EX2.09, This section adds the new `\qbezier` command for drawing curves.

`\qbezier` `\qbezier[N]($\langle AX,AY \rangle$)($\langle BX,BY \rangle$)($\langle CX,CY \rangle$)` plots a quadratic Bezier curve from ($\langle AX,AY \rangle$) to ($\langle CX,CY \rangle$), with ($\langle BX,BY \rangle$) as the third Bezier point, using $N+1$ points equally spaced parametrically. If $N = 0$ (the default value), then a sufficient number of points are used to draw a connected curve—except that at most `\qbeziermax+1` points are drawn. A “point” is a square of side `\@wholewidth`.

`\bezier` In addition, to be compatible with the old `bezier` package, a variant of this command, `\bezier`, is defined, in which the first argument is not optional.

Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

```
\unitlength      = value of dimension argument
\@wholewidth     = current line width
\@halfwidth     = half of current line width
\@linefont      = font for drawing lines
\@circlefont    = font for drawing circles
```

```
\linethickness{DIM} : Sets the width of horizontal and vertical lines
in a picture to DIM. Does not change width of slanted lines
or circles. Width of all lines reset by \thinlines and
\thicklines
```

```
\picture(XSIZE,YSIZE)(XORG,YORG)
BEGIN
  \@picht :=L YSIZE * \unitlength
  box \@picbox :=
    \hb@xt@ XSIZE * \unitlength
    {\hskip -XORG * \unitlength
     \lower YORG * \unitlength
     \hbox{
       \ignorespaces      %% added 13 June 89
     }
  }
END
```

```
\endpicture ==
BEGIN
    } \hss }
  height of \@picbox := \@picht
  depth of \@picbox := 0
  \mbox{\box\@picbox}  %% change 26 Aug 91
END
```

```
\put(X, Y){OBJ} ==
BEGIN
```

```

\@killglue
\raise Y * \unitlength \hb@xt@ 0pt { \hskip X * \unitlength
                                OBJ \hss }

\ignorespaces
END

\multiput(X,Y)(DELX,DELY){N}{OBJ} ==
BEGIN
\@killglue
\@multicnt := N
\@xdim := X * \unitlength
\@ydim := Y * \unitlength
while \@multicnt > 0
do \raise \@ydim \hb@xt@ 0pt { \hskip \@xdim
                                OBJ \hss }

\@multicnt := \@multicnt - 1
\@xdim := \@xdim + DELX * \unitlength
\@ydim := \@ydim + DELY * \unitlength
od
\ignorespaces
END

```

`\shortstack[POS]{TEXT}` : Makes a `\vbox` containing TEXT stacked as a one-column array, positioned l, r or c as indicated by POS.

End of historical L^AT_EX 2.09 comments.

The ‘2ekernel’ code ensures that a `\usepackage{autopict}` is essentially ignored if a ‘full’ format is being used that has picture mode already in the format.

```
1 <2ekernel>\expandafter\let\csname ver@autopict.sty\endcsname\fmtversion
```

```

\@wholewidth
\@halfwidth 2 <*2ekernel>
3 \newdimen\@wholewidth
4 \newdimen\@halfwidth

```

(End of definition for \@wholewidth and \@halfwidth.)

```

\unitlength
5 \newdimen\unitlength \unitlength =1pt

```

(End of definition for \unitlength.)

```

\@picbox
\@picht 6 \newbox\@picbox
7 \newdimen\@picht

```

(End of definition for \@picbox and \@picht.)

`\@defaultunitsset` Set a length register, #1, accepting number or an etex length expression, #2, with default unit, #3.

The register name in #1 can be prefixed by `\advance` so that the register is incremented by the supplied value.

```
\@defaultunitsset{\advance\@vxx}{\textwidth-15pt}\unitlength
#3 can be a literal unit such as cm or a length register such as \unitlength.
```

This is used in all `picture` commands that take picture coordinates. So `\put(2,2)` as previously but now `\put(\textwidth-5cm,0.4\texteight)` Note that you can only use expressions with lengths, `\put(1+2,0)` is not supported.

```
8 </2ekernel>
9 < *2ekernel | latexrelease >
10 < latexrelease > \IncludeInRelease{2020/10/01}%
11 < latexrelease >           {\@defaultunitsset}{default units}%
12 \def\@defaultunitsset#1#2#3{%
13   \@defaultunits#1\dimexpr#2#3\relax\relax\@nnil}
14 </2ekernel | latexrelease >

15 < latexrelease > \EndIncludeInRelease
16 < latexrelease > \IncludeInRelease{0000/00/00}%
17 < latexrelease >           {\@defaultunitsset}{default units}%
18 < latexrelease > \let\@defaultunitsset\@undefined
19 < latexrelease > \EndIncludeInRelease
20 < *2ekernel >
```

(End of definition for \@defaultunitsset.)

`\picture` #1 should be white space.

#1 should be a ((eating any white space before the bracket),

```
\picture@ 21 \long\def\picture#1{\picture@#1}
22 \def\picture@(#1){%
23   \ifnextchar({\@picture(#1)}{\@picture(#1)(0,0)}}
```

(End of definition for \picture and \picture@.)

`\@picture`

```
24 </2ekernel>
25 < *2ekernel | latexrelease >
26 < latexrelease > \IncludeInRelease{2020/10/01}%
27 < latexrelease >           {\@picture}{default units}%
28 \def\@picture(#1,#2)(#3,#4){%
29   \@defaultunitsset\@picht{#2}\unitlength
30   \@defaultunitsset\@tempdimc{#1}\unitlength
31   \setbox\@picbox\hb@xt@\@tempdimc\bgroup
32     \@defaultunitsset\@tempdimc{#3}\unitlength
33     \hskip -\@tempdimc
34     \@defaultunitsset\@tempdimc{#4}\unitlength
35     \lower\@tempdimc\hbox\bgroup
36     \ignorespaces}
37 </2ekernel | latexrelease >
```

```

38 <latexrelease>\EndIncludeInRelease
39 <latexrelease>\IncludeInRelease{0000/00/00}%
40 <latexrelease>          {\@picture}{default units}%
41 <latexrelease>\def\@picture(#1,#2)(#3,#4){%
42 <latexrelease>  \@picht#2\unitlength
43 <latexrelease>  \setbox\@picbox\hb@xt@#1\unitlength\bgroup
44 <latexrelease>    \hskip -#3\unitlength
45 <latexrelease>    \lower #4\unitlength\hbox\bgroup
46 <latexrelease>    \ignorespaces}
47 <latexrelease>\EndIncludeInRelease
48 <*2ekernel)

```

(End of definition for \@picture.)

\endpicture

```

49 \def\endpicture{%
50   \egroup\hss\egroup
51   \ht\@picbox\@picht\dp\@picbox\z@
52   \mbox{\box\@picbox}}

```

(End of definition for \endpicture.)

In the definitions of \put and \multiput, \hskip was replaced by \kern just in case arg #3 = “plus”. (Bug detected by Don Knuth. changed 20 Jul 87).

```

53 </2ekernel)
54 <*2ekernel | latexrelease)
55 <latexrelease>\IncludeInRelease{2020/10/01}%
56 <latexrelease>          {\put}{default units}%
57 <latexrelease>\expandafter\let\csname put \endcsname\@undefind
58 \long\def\put(#1,#2)#3{%
59   \@killglue
60   \@defaultunitsset\@tempdimc{#2}\unitlength
61   \raise\@tempdimc
62   \hb@xt@\z@{%
63     \@defaultunitsset\@tempdimc{#1}\unitlength
64     \kern\@tempdimc
65     #3\hss}%
66   \ignorespaces}
67 </2ekernel | latexrelease)
68 <latexrelease>\EndIncludeInRelease
69 <latexrelease>\IncludeInRelease{0000/00/00}%
70 <latexrelease>          {\put}{default units}%
71 <latexrelease>\expandafter\let\csname put \endcsname\@undefind
72 <latexrelease>\long\def\put(#1,#2)#3{%
73 <latexrelease>  \@killglue\raise#2\unitlength
74 <latexrelease>  \hb@xt@\z@{\kern#1\unitlength #3\hss}%
75 <latexrelease>  \ignorespaces}
76 <latexrelease>\EndIncludeInRelease
77 <*2ekernel)

```

\multiput #3 had better be a (.)

```

78 </2ekernel)
79 <*2ekernel | latexrelease)
80 <latexrelease>\IncludeInRelease{2020/10/01}%

```

```

81 <latexrelease>                {\multiput}{default units}%
82 <latexrelease>\expandafter\let\csname multiput \endcsname \@undefind
83 \def\multiput(#1,#2)#3{%
84   \@defaultunitsset\@xdim{#1}\unitlength
85   \@defaultunitsset\@ydim{#2}\unitlength
86   \@multiput{ }
87 </2ekernel | latexrelease>

88 <latexrelease>\EndIncludeInRelease
89 <latexrelease>\IncludeInRelease{0000/00/00}%
90 <latexrelease>                {\multiput}{default units}%
91 <latexrelease>\expandafter\let\csname multiput \endcsname \@undefind
92 <latexrelease>\def\multiput(#1,#2)#3{%
93 <latexrelease>  \@xdim #1\unitlength
94 <latexrelease>  \@ydim #2\unitlength
95 <latexrelease>  \@multiput{ }
96 <latexrelease>\EndIncludeInRelease
97 <*2ekernel>

```

(End of definition for \multiput.)

`\@multiput`

```

98 </2ekernel>
99 <*2ekernel | latexrelease>
100 <latexrelease>\IncludeInRelease{2020/10/01}%
101 <latexrelease>                {\@multiput}{default units}%
102 \long\def\@multiput(#1,#2)#3#4{%
103   \@killglue\@multicnt #3\relax
104   \@whilenum \@multicnt >\z\do
105     {\raise\@ydim\hb@xt@{\z@{\kern\@xdim #4\hss}}%
106     \advance\@multicnt\m@ne
107     \@defaultunitsset{\advance\@xdim}{#1}\unitlength
108     \@defaultunitsset{\advance\@ydim}{#2}\unitlength}%
109   \ignorespaces}
110 </2ekernel | latexrelease>

111 <latexrelease>\EndIncludeInRelease
112 <latexrelease>\IncludeInRelease{0000/00/00}%
113 <latexrelease>                {\@multiput}{default units}%
114 <latexrelease>\long\def\@multiput(#1,#2)#3#4{%
115 <latexrelease>  \@killglue\@multicnt #3\relax
116 <latexrelease>  \@whilenum \@multicnt >\z\do
117 <latexrelease>    {\raise\@ydim\hb@xt@{\z@{\kern\@xdim #4\hss}}%
118 <latexrelease>    \advance\@multicnt\m@ne
119 <latexrelease>    \advance\@xdim#1\unitlength\advance\@ydim#2\unitlength}%
120 <latexrelease>  \ignorespaces}
121 <latexrelease>\EndIncludeInRelease
122 <*2ekernel>

```

(End of definition for \@multiput.)

`\@killglue`

```

123 \def\@killglue{\unskip\@whiledim \lastskip >\z\do{\unskip}}

```

(End of definition for \@killglue.)


```

\thinlines
\thicklines 124 \DeclareRobustCommand\thinlines{\let\@linefnt\tenln
125 \let\@circlefnt\tencirc
126 \@wholewidth\fontdimen8\tenln \@halfwidth .5\@wholewidth}
127 \DeclareRobustCommand\thicklines{\let\@linefnt\tenlnw
128 \let\@circlefnt\tencircw
129 \@wholewidth\fontdimen8\tenlnw \@halfwidth .5\@wholewidth}

(End of definition for \thinlines and \thicklines.)

\linethickness
130 \DeclareRobustCommand*\linethickness[1]
131 {\@wholewidth #1\relax \@halfwidth .5\@wholewidth \ignorespaces}

(End of definition for \linethickness.)

\ishortstack
132 \def\shortstack{\@ifnextchar[\@shortstack{\@shortstack[c]}}

(End of definition for \ishortstack.)

\@ishortstack
133 \def\@shortstack[#1]{%
134 \leavevmode
135 \vbox\bgroup
136 \baselineskip-\p@\lineskip 3\p@
137 \let\mb@l\hss\let\mb@r\hss
138 \expandafter\let\csname mb@#1\endcsname\relax
139 \let\\\@stackcr
140 \@ishortstack}

(End of definition for \@ishortstack.)

\@ishortstack
141 \def\@ishortstack#1{\ialign{\mb@l {##}\unskip\mb@r\cr #1\crr}\egroup}

(End of definition for \@ishortstack.)

\@stackcr
\@ixstackcr 142 \protected\def\@stackcr{\@ifstar\@ixstackcr\@ixstackcr}
143 \def\@ixstackcr{\@ifnextchar[\@istackcr{\cr\ignorespaces}}

(End of definition for \@stackcr and \@ixstackcr.)

\@istackcr
144 </2ekernel>
145 <*2ekernel | latexrelease>
146 <latexrelease>\IncludeInRelease{2020/10/01}%
147 <latexrelease> \@istackcr}{\shortstack calc support}%
148 \def\@istackcr[#1]{\cr\noalign{\@vspace@calcify{#1}}\ignorespaces}
149 </2ekernel | latexrelease>

```

```

150 <latexrelease>\EndIncludeInRelease
151 <latexrelease>\IncludeInRelease{0000/00/00}%
152 <latexrelease>          {\@istackcr}{\shortstack calc support}%
153 <latexrelease>
154 <latexrelease>\def\@istackcr[#1]{\cr\noalign{\vskip #1}\ignorespaces}
155 <latexrelease>\EndIncludeInRelease
156 <*2kernel>

(End of definition for \@istackcr.)
Historical LATEX 2.09 comments (not necessarily accurate any more):
\line(X,Y){LEN} ==
BEGIN
  \@xarg    := X
  \@yarg    := Y
  \@linelen := LEN * \unitlength
  if \@xarg = 0
    then \@vline
    else if \@yarg = 0
      then \@hline
      else \@sline
        if
          if
            END

\@sline ==
BEGIN
  if \@xarg < 0
    then @negarg := T
           \@xarg := -\@xarg
           \@yyarg := -\@yarg
    else @negarg := F
           \@yyarg := \@yarg
  fi
  \@tempcnta := |\@yyarg|
  if \@tempcnta > 6
    then error: 'LATEX ERROR: Illegal \line or \vector argument.'
           \@tempcnta := 0
  fi
  \box\@linechar := \hbox{\@linefnt \@getlinechar(\@xarg,\@yyarg) }
  if \@yarg > 0 then \@upordown = \raise
                    \@clnht := 0
                    else \@upordown = \lower
                    \@clnht := height of \box\@linechar
  fi
  \@clnwd := width of \box\@linechar
  if @negarg
    then \hskip - width of \box\@linechar
           \reserved@a == \hskip - 2* width of box \@linechar
    else \reserved@a == \relax
  fi
  %% Put out integral number of line segments

```

```

while \@clnwd < \@linelen
do \@upordown \@clnht \copy\@linechar
\reserved@a
\@clnht := \@clnht + ht of \box\@linechar
\@clnwd := \@clnwd + width of \box\@linechar
od

% Put out last segment
\@clnht := \@clnht - height of \box\@linechar
\@clnwd := \@clnwd - width of \box\@linechar
\@tempdima := \@linelen - \@clnwd
\@tempdimb := \@tempdima - width of \box\@linechar
if @negarg then \hskip -\@tempdimb
else \hskip \@tempdimb
fi
\@tempdima := 1000 * \@tempdima
\@tempcnta := \@tempdima / width of \box\@linechar
\@tempdima := (\@tempcnta * ht of \box\@linechar)/1000
\@clnht := \@clnht + \@tempdima
if \@linelen < width of \box\@linechar
then \hskip width of \box\@linechar
else \hbox{\@upordown \@clnht \copy\@linechar}
fi
END

\@hline ==
BEGIN
if \@xarg < 0 then \hskip -\@linelen \fi
\vrule height \@halfwidth depth \@halfwidth width \@linelen
if \@xarg < 0 then \hskip -\@linelen \fi
END

\@vline == if \@yarg < 0 \@downline else \@upline fi

\@getlinechar(X,Y) ==
BEGIN
\@tempcnta := 8*X - 9
if Y > 0
then \@tempcnta := \@tempcnta + Y
else \@tempcnta := \@tempcnta - Y + 64
fi
\char\@tempcnta
END

\vector(X,Y){LEN} ==
BEGIN
\@xarg := X
\@yarg := Y
\@linelen := LEN * \unitlength

```

```

if \@xarg = 0
  then \@vvector
  else if \@yarg = 0
    then \@hvector
    else \@svector
  if
if
END

\@hvector ==
BEGIN
  \@hline
  {\@linefnt if \@xarg < 0 then \@getlarrow(1,0)
    else \@getrarrow(1,0)
  fi}
END

\@vvector == if \@yarg < 0 \@downvector else \@upvector fi

\@svector ==
BEGIN
  \@sline
  \@tempcnta := |\@yarg|
  if \@tempcnta < 5
    then \hskip - width of \box\@linechar
      \@upordown \@clnht \hbox
        {\@linefnt
          if @negarg then \@getlarrow(\@xarg,\@yyarg)
            else \@getrarrow(\@xarg,\@yyarg)
          fi }
    else error: 'LATEX ERROR: Illegal \line or \vector argument.'
  fi
END

\@getlarrow(X,Y) ==
BEGIN
  if Y = 0
    then \@tempcnta := '33
    else \@tempcnta := 16 * X - 9
      \@tempcntb := 2 * Y
      if \@tempcntb > 0
        then \@tempcnta := \@tempcnta + \@tempcntb
        else \@tempcnta := \@tempcnta - \@tempcntb + 64
      fi
  fi
  \char\@tempcnta
END

\@getrarrow(X,Y) ==
BEGIN

```

```

\@tempcntb := |Y|
case of \@tempcntb
  0 : \@tempcnta := '55
  1 : if X < 3
      then \@tempcnta := 24*X - 6
      else if X = 3
          then \@tempcnta := 49
          else \@tempcnta := 58 fi
      fi
  2 : if X < 3
      then \@tempcnta := 24*X - 3
      else \@tempcnta := 51      % X must = 3
      fi
  3 : \@tempcnta := 16*X - 2
  4 : \@tempcnta := 16*X + 7
endcase
if Y < 0
  then \@tempcnta := \@tempcnta + 64
  fi
\char\@tempcnta
END

```

End of historical L^AT_EX 2.09 comments.

`\if@negarg`

```

157 \newif\if@negarg
(End of definition for \if@negarg.)

```

`\line`

```

158 </2kernel>
159 < *2kernel | latexrelease >
160 < latexrelease > \IncludeInRelease{2020/10/01}%
161 < latexrelease >          {\line}{default units}%
162 < latexrelease > \expandafter\let\csname line \endcsname\@undefined
163 \def\line(#1,#2)#3{\@xarg #1\relax \@yarg #2\relax
164   \@defaultunitsset\@linelen{#3}\unitlength
165   \ifdim\@linelen<\z@\@badlinearg\else
166     \ifnum\@xarg =\z@ \@vline
167       \else \ifnum\@yarg =\z@ \@hline \else \@sline\fi
168     \fi
169   \fi}
170 </2kernel | latexrelease >

171 < latexrelease > \EndIncludeInRelease
172 < latexrelease > \IncludeInRelease{0000/00/00}%
173 < latexrelease >          {\line}{default units}%
174 < latexrelease > \expandafter\let\csname line \endcsname\@undefined
175 < latexrelease > \def\line(#1,#2)#3{\@xarg #1\relax \@yarg #2\relax
176 < latexrelease >   \@linelen #3\unitlength
177 < latexrelease >   \ifdim\@linelen<\z@\@badlinearg\else
178 < latexrelease >     \ifnum\@xarg =\z@ \@vline
179 < latexrelease >       \else \ifnum\@yarg =\z@ \@hline \else \@sline\fi
180 < latexrelease >     \fi

```

```

181 <latexrelease> \fi}
182 <latexrelease>\EndIncludeInRelease
183 <*2kernel>

```

(End of definition for \line.)

\@sline

```

184 \def\@sline{%
185   \ifnum\@xarg<\z@ \@negargtrue \@xarg -\@xarg \@yyarg -\@yarg
186   \else \@negargfalse \@yyarg \@yarg \fi
187   \ifnum \@yyarg >\z@ \@tempcnta\@yyarg \else \@tempcnta -\@yyarg \fi
188   \ifnum\@tempcnta>6 \@badlinearg\@tempcnta\z@ \fi
189   \ifnum\@xarg>6 \@badlinearg\@xarg \@ne \fi
190   \setbox\@linechar\hbox{\@linefnt\@getlinechar(\@xarg,\@yyarg)}%

```

If we have something like `\line(5,5){30}` the `\@linechar` will not contain a char and later on we will end in an infinite loop. So we check the width of the box and put in something as an emergency fix if necessary.

```

191 \ifdim\wd\@linechar=\z@
192   \setbox\@linechar\hbox{.}%
193   \@badlinearg
194 \fi
195 \ifnum \@yarg >\z@ \let\@upordown\raise \@clnht\z@
196   \else\let\@upordown\lower \@clnht \ht\@linechar\fi
197 \@clnwd \wd\@linechar
198 \if@negarg
199   \hskip -\wd\@linechar \def\reserved@a{\hskip -2\wd\@linechar}%
200 \else
201   \let\reserved@a\relax
202 \fi
203 \@whiledim \@clnwd <\@linelen \do
204   {\@upordown\@clnht\copy\@linechar
205     \reserved@a
206     \advance\@clnht \ht\@linechar
207     \advance\@clnwd \wd\@linechar}%
208 \advance\@clnht -\ht\@linechar
209 \advance\@clnwd -\wd\@linechar
210 \@tempdima\@linelen\advance\@tempdima -\@clnwd
211 \@tempdimb\@tempdima\advance\@tempdimb -\wd\@linechar
212 \if@negarg \hskip -\@tempdimb \else \hskip \@tempdimb \fi
213 \multiply\@tempdima \@m
214 \@tempcnta \@tempdima
215 \@tempdima \wd\@linechar \divide\@tempcnta \@tempdima
216 \@tempdima \ht\@linechar \multiply\@tempdima \@tempcnta
217 \divide\@tempdima \@m
218 \advance\@clnht \@tempdima
219 \ifdim \@linelen <\wd\@linechar
220   \hskip \wd\@linechar

```

Warn if line gets so short that it can't be printed. But don't warn if it is exactly zero since that was probably deliberate (e.g., to get a vector head only).

```

221 \ifdim \@linelen = \z@
222 \else
223   \@picture@warn
224 \fi

```

```
225 \else\@upordown\@clnht\copy\@linechar\fi}
```

(End of definition for \@sline.)

\@hline

```
226 \def\@hline{\ifnum \@xarg <\z@ \hskip -\@linelen \fi
227 \vrule \@height \@halfwidth \@depth \@halfwidth \@width \@linelen
228 \ifnum \@xarg <\z@ \hskip -\@linelen \fi}
```

(End of definition for \@hline.)

\@getlinechar

```
229 \def\@getlinechar(#1,#2){\@tempcnta#1\relax\multiply\@tempcnta 8%
230 \advance\@tempcnta -9\ifnum #2>\z@ \advance\@tempcnta #2\relax\else
231 \advance\@tempcnta -#2\relax\advance\@tempcnta 64 \fi
232 \char\@tempcnta}
```

(End of definition for \@getlinechar.)

\vector

```
233 \</2ekernel>
234 \*2ekernel | latexrelease>
235 \<latexrelease>\IncludeInRelease{2020/10/01}%
236 \<latexrelease> \{\vector\}{default units}%
237 \<latexrelease>\expandafter\let\csname vector \endcsname\@undefind
238 \def\vector(#1,#2)#3{\@xarg #1\relax \@yarg #2\relax
239 \@tempcnta \ifnum\@xarg<\z@ -\@xarg\else\@xarg\fi
240 \ifnum\@tempcnta<5\relax
241 \@defaultunitsset\@linelen{#3}\unitlength
242 \ifdim\@linelen<\z@\@badlinearg\else
243 \ifnum\@xarg =\z@ \@vvector
244 \else \ifnum\@yarg =\z@ \@hvector \else \@svector\fi
245 \fi
246 \fi
247 \else\@badlinearg\fi}
248 \</2ekernel | latexrelease>
249 \<latexrelease>\EndIncludeInRelease
250 \<latexrelease>\IncludeInRelease{0000/00/00}%
251 \<latexrelease> \{\vector\}{default units}%
252 \<latexrelease>\expandafter\let\csname vector \endcsname\@undefind
253 \<latexrelease>\def\vector(#1,#2)#3{\@xarg #1\relax \@yarg #2\relax
254 \<latexrelease> \@tempcnta \ifnum\@xarg<\z@ -\@xarg\else\@xarg\fi
255 \<latexrelease> \ifnum\@tempcnta<5\relax
256 \<latexrelease> \@linelen #3\unitlength
257 \<latexrelease> \ifdim\@linelen<\z@\@badlinearg\else
258 \<latexrelease> \ifnum\@xarg =\z@ \@vvector
259 \<latexrelease> \else \ifnum\@yarg =\z@ \@hvector \else \@svector\fi
260 \<latexrelease> \fi
261 \<latexrelease> \fi
262 \<latexrelease> \else\@badlinearg\fi}
263 \<latexrelease>\EndIncludeInRelease
264 \*2ekernel)
```

(End of definition for \vector.)

`\@hvector`

```
265 \def\@hvector{\@hline\hb@xt@z@{\@linefnt
266 \ifnum \@xarg <z@ \@getlarrow(1,0)\hss\else
267 \hss\@getrarrow(1,0)\fi}}
```

(End of definition for \@hvector.)

`\@vvector`

```
268 \def\@vvector{\ifnum \@yarg <z@ \@downvector \else \@upvector \fi}
```

(End of definition for \@vvector.)

`\@svector`

```
269 \def\@svector{\@sline
270 \@tempcnta\@yarg \ifnum\@tempcnta <z@ \@tempcnta -\@tempcnta\fi
271 \ifnum\@tempcnta <5%
272 \hskip -\wd\@linechar
273 \@upordown\@clnht \hbox{\@linefnt \if@negarg
274 \@getlarrow(\@xarg,\@yyarg)\else \@getrarrow(\@xarg,\@yyarg)\fi}%
275 \else\@badlinearg\fi}
```

(End of definition for \@svector.)

`\@getlarrow`

```
276 \def\@getlarrow(#1,#2){\ifnum #2=z@ \@tempcnta 27 % '33
277 \else
278 \@tempcnta #1\relax\multiply\@tempcnta \sixt@@n
279 \advance\@tempcnta -9 \@tempcntb #2\relax\multiply\@tempcntb \tw@
280 \ifnum \@tempcntb >z@ \advance\@tempcnta \@tempcntb
281 \else\advance\@tempcnta -\@tempcntb\advance\@tempcnta 64
282 \fi\fi\char\@tempcnta}
```

(End of definition for \@getlarrow.)

`\@getrarrow`

```
283 \def\@getrarrow(#1,#2){\@tempcntb #2\relax
284 \ifnum\@tempcntb <z@ \@tempcntb -\@tempcntb\relax\fi
285 \ifcase \@tempcntb\relax \@tempcnta 45 % '55
286 \or
287 \ifnum #1<\thr@@ \@tempcnta #1\relax\multiply\@tempcnta
288 24\advance\@tempcnta -6 \else \ifnum #1=\thr@@ \@tempcnta 49
289 \else\@tempcnta 58 \fi\fi\or
290 \ifnum #1<\thr@@ \@tempcnta=#1\relax\multiply\@tempcnta
291 24\advance\@tempcnta -\thr@@ \else \@tempcnta 51 \fi\or
292 \@tempcnta #1\relax\multiply\@tempcnta
293 \sixt@@n \advance\@tempcnta -\tw@ \else
294 \@tempcnta #1\relax\multiply\@tempcnta
295 \sixt@@n \advance\@tempcnta 7 \fi\ifnum #2<z@ \advance\@tempcnta 64 \fi
296 \char\@tempcnta}
```

(End of definition for \@getrarrow.)

`\@vline`

```
297 \def\@vline{\ifnum \@yarg <z@ \@downline \else \@upline\fi}
```


(End of definition for \vline.)

\@upline

```
298 \def\@upline{%
299   \hb@xt@z@{\hskip -\@halfwidth \vrule \@width \@wholewidth
300   \@height \@linelen \@depth z@\hss}}
```

(End of definition for \@upline.)

\@downline

```
301 \def\@downline{%
302   \hb@xt@z@{\hskip -\@halfwidth \vrule \@width \@wholewidth
303   \@height z@ \@depth \@linelen \hss}}
```

(End of definition for \@downline.)

\@upvector

```
304 \def\@upvector{\@upline\setbox\@tempboxa\hbox{\@linefnt\char 54}% '66
305   \raise \@linelen \hb@xt@z@{\lower \ht\@tempboxa\box\@tempboxa\hss}}
```

(End of definition for \@upvector.)

\@downvector

```
306 \def\@downvector{\@downline\lower \@linelen
307   \hb@xt@z@{\@linefnt\char 63 % '77
308   \hss}}
```

(End of definition for \@downvector.)

Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

```
\dashbox{D}(X,Y) ==
BEGIN
leave vertical mode
\hb@xt@ Opt {
  \baselineskip := Opt
  \lineskip := Opt
%% HORIZONTAL DASHES
  \@dashdim := X * \unitlength
  \@dashcnt := \@dashdim + 200 % to prevent roundoff error
  \@dashdim := D * \unitlength
  \@dashcnt := \@dashcnt / \@dashdim
  if \@dashcnt is odd
  then \@dashdim := Opt
      \@dashcnt := (\@dashcnt + 1) / 2
  else \@dashdim := \@dashdim / 2
      \@dashcnt := \@dashcnt / 2 - 1
      \box\@dashbox := \hbox{\vrule height \@halfwidth
      depth \@halfwidth width \@dashdim}
      \put(0,0){\copy\@dashbox}
      \put(0,Y){\copy\@dashbox}
      \put(X,0){\hskip -\@dashdim\copy\@dashbox}
      \put(X,Y){\hskip -\@dashdim\box\@dashbox}
  \@dashdim := 3 * \@dashdim
fi
```

```

\box\@dashbox := \hbox{\vrule height \@halfwidth
                        depth \@halfwidth width D * \unitlength
                        \hskip D * \unitlength}

\@tempcnta := 0
\put(0,0){\hskip \@dashdim
          while \@tempcnta < \@dashcnt
            do \copy\@dashbox
              \@tempcnta := \@tempcnta + 1
            od
          }
\@tempcnta := 0
put(0,Y){\hskip \@dashdim
         while \@tempcnta < \@dashcnt
           do \copy\@dashbox
             \@tempcnta := \@tempcnta + 1
           od
         }

%% vertical dashes
\@dashdim := Y * \unitlength
\@dashcnt := \@dashdim + 200 % to prevent roundoff error
\@dashdim := D * \unitlength
\@dashcnt := \@dashcnt / \@dashdim
if \@dashcnt is odd
then \@dashdim := 0pt
    \@dashcnt := (\@dashcnt + 1) / 2
else \@dashdim := \@dashdim / 2
    \@dashcnt := \@dashcnt / 2 - 1
\box\@dashbox := \hbox{\hskip -\@halfwidth
                        \vrule width \@wholewidth
                        height \@dashdim }

\put(0,0){\copy\@dashbox}
\put(X,0){\copy\@dashbox}
\put(0,Y){\lower\@dashdim\copy\@dashbox}
\put(X,Y){\lower\@dashdim\copy\@dashbox}
\@dashdim := 3 * \@dashdim
fi
\box\@dashbox := \hbox{\vrule width \@wholewidth
                        height D * \unitlength }

\@tempcnta := 0
put(0,0){\hskip -\@halfwidth
        \vbox{while \@tempcnta < \@dashcnt
              do \vskip D*\unitlength
                \copy\@dashbox
                \@tempcnta := \@tempcnta + 1
              od
              \vskip \@dashdim
            } }
\@tempcnta := 0
put(X,0){\hskip -\@halfwidth

```

```

        \vbox{while \@tempcnta < \@dashcnt
            do \vskip D*\unitlength
                \copy\@dashbox
                \@tempcnta := \@tempcnta + 1
            od
            \vskip \@dashdim
        }
    } % END DASHES

```

```
\@makepicbox(X,Y)
```

```
END
```

End of historical L^AT_EX 2.09 comments.

```
\dashbox
```

```

309 </2kernel>
310 <*2kernel | latexrelease>
311 <latexrelease>\IncludeInRelease{2020/10/01}%
312 <latexrelease>                {\dashbox}{default units}%
313 <latexrelease>\expandafter\let\csname dashbox \endcsname\undefind
314 \def\dashbox#1(#2,#3){\leavevmode\hb@xt@\z@{\baselineskip \z@skip
315 \lineskip \z@skip
316 \@defaultunitsset\@dashdim{#2}\unitlength
317 \@dashcnt \@dashdim \advance\@dashcnt 200
318 \@defaultunitsset\@dashdim{#1}\unitlength
319 \divide\@dashcnt \@dashdim
320 \ifodd\@dashcnt\@dashdim \z@
321 \advance\@dashcnt \@one \divide\@dashcnt \tw@
322 \else \divide\@dashdim \tw@ \divide\@dashcnt \tw@
323 \advance\@dashcnt \m@ne
324 \setbox\@dashbox \hbox{\vrule \@height \@halfwidth \@depth \@halfwidth
325 \@width \@dashdim}\put(0,0){\copy\@dashbox}%
326 \put(0,#3){\copy\@dashbox}%
327 \put(#2,0){\hskip-\@dashdim\copy\@dashbox}%
328 \put(#2,#3){\hskip-\@dashdim\box\@dashbox}%
329 \multiply\@dashdim \thr@@
330 \fi
331 \setbox\@dashbox \hbox{%
332   \@defaultunitsset\@tempdimc{#1}\unitlength
333   \vrule \@height \@halfwidth \@depth \@halfwidth \@width \@tempdimc
334   \hskip\@tempdimc}%
335 \@tempcnta\z@
336 \put(0,0){\hskip\@dashdim \@whilenum \@tempcnta <\@dashcnt
337 \do{\copy\@dashbox\advance\@tempcnta \@one }}\@tempcnta\z@
338 \put(0,#3){\hskip\@dashdim \@whilenum \@tempcnta <\@dashcnt
339 \do{\copy\@dashbox\advance\@tempcnta \@one }}%
340 \@defaultunitsset\@dashdim{#3}\unitlength
341 \@dashcnt \@dashdim \advance\@dashcnt 200
342 \@defaultunitsset\@dashdim{#1}\unitlength
343 \divide\@dashcnt \@dashdim
344 \ifodd\@dashcnt \@dashdim \z@
345 \advance\@dashcnt \@one \divide\@dashcnt \tw@
346 \else

```

```

347 \divide\@dashdim \tw@ \divide\@dashcnt \tw@
348 \advance\@dashcnt \m@ne
349 \setbox\@dashbox\hbox{\hskip -\@halfwidth
350 \vrule \@width \@wholewidth
351 \@height \@dashdim}\put(0,0){\copy\@dashbox}%
352 \put(#2,0){\copy\@dashbox}%
353 \put(0,#3){\lower\@dashdim\copy\@dashbox}%
354 \put(#2,#3){\lower\@dashdim\copy\@dashbox}%
355 \multiply\@dashdim \thr@@
356 \fi
357 \@defaultunitsset\@tempdimb{#1}\unitlength
358 \setbox\@dashbox\hbox{%
359 \vrule \@width \@wholewidth \@height\@tempdimb}%
360 \@tempcnta\z@
361 \put(0,0){\hskip -\@halfwidth \vbox{\@whilenum \@tempcnta <\@dashcnt
362 \do{\vskip\@tempdimb\copy\@dashbox\advance\@tempcnta \@ne }%
363 \vskip\@dashdim}}\@tempcnta\z@
364 \put(#2,0){\hskip -\@halfwidth \vbox{\@whilenum \@tempcnta<\@dashcnt
365 \do{\vskip\@tempdimb\copy\@dashbox\advance\@tempcnta \@ne }%
366 \vskip\@dashdim}}\@makepicbox(#2,#3)}
367 /2kernel | latexrelease)

368 <latexrelease>\EndIncludeInRelease
369 <latexrelease>\IncludeInRelease{0000/00/00}%
370 <latexrelease> \dashbox{default units}%
371 <latexrelease>\expandafter\let\csname dashbox \endcsname\@undefind
372 <latexrelease>\def\dashbox#1(#2,#3){%
373 <latexrelease>\leavevmode\hb@xt@z@{\baselineskip \z@skip
374 <latexrelease>\lineskip \z@skip
375 <latexrelease>\@dashdim #2\unitlength
376 <latexrelease>\@dashcnt \@dashdim \advance\@dashcnt 200
377 <latexrelease>\@dashdim #1\unitlength\divide\@dashcnt \@dashdim
378 <latexrelease>\ifodd\@dashcnt\@dashdim \z@
379 <latexrelease>\advance\@dashcnt \@ne \divide\@dashcnt \tw@
380 <latexrelease>\else \divide\@dashdim \tw@ \divide\@dashcnt \tw@
381 <latexrelease>\advance\@dashcnt \m@ne
382 <latexrelease>\setbox\@dashbox \hbox{%
383 <latexrelease> \vrule \@height \@halfwidth \@depth \@halfwidth
384 <latexrelease> \@width \@dashdim}\put(0,0){\copy\@dashbox}%
385 <latexrelease>\put(0,#3){\copy\@dashbox}%
386 <latexrelease>\put(#2,0){\hskip-\@dashdim\copy\@dashbox}%
387 <latexrelease>\put(#2,#3){\hskip-\@dashdim\box\@dashbox}%
388 <latexrelease>\multiply\@dashdim \thr@@
389 <latexrelease>\fi
390 <latexrelease>\setbox\@dashbox \hbox{%
391 <latexrelease> \vrule \@height \@halfwidth \@depth \@halfwidth
392 <latexrelease> \@width #1\unitlength\hskip #1\unitlength}\@tempcnta\z@
393 <latexrelease>\put(0,0){\hskip\@dashdim \@whilenum \@tempcnta <\@dashcnt
394 <latexrelease>\do{\copy\@dashbox\advance\@tempcnta \@ne }}\@tempcnta\z@
395 <latexrelease>\put(0,#3){\hskip\@dashdim \@whilenum \@tempcnta <\@dashcnt
396 <latexrelease>\do{\copy\@dashbox\advance\@tempcnta \@ne }}%
397 <latexrelease>\@dashdim #3\unitlength
398 <latexrelease>\@dashcnt \@dashdim \advance\@dashcnt 200
399 <latexrelease>\@dashdim #1\unitlength\divide\@dashcnt \@dashdim
400 <latexrelease>\ifodd\@dashcnt \@dashdim \z@

```

```

401 <latexrelease>\advance\@dashcnt \@ne \divide\@dashcnt \tw@
402 <latexrelease>\else
403 <latexrelease>\divide\@dashdim \tw@ \divide\@dashcnt \tw@
404 <latexrelease>\advance\@dashcnt \m@ne
405 <latexrelease>\setbox\@dashbox\hbox{\hskip -\@halfwidth
406 <latexrelease>\vrule \@width \@wholewidth
407 <latexrelease>\@height \@dashdim}\put(0,0){\copy\@dashbox}%
408 <latexrelease>\put(#2,0){\copy\@dashbox}%
409 <latexrelease>\put(0,#3){\lower\@dashdim\copy\@dashbox}%
410 <latexrelease>\put(#2,#3){\lower\@dashdim\copy\@dashbox}%
411 <latexrelease>\multiply\@dashdim \thr@@
412 <latexrelease>\fi
413 <latexrelease>\setbox\@dashbox\hbox{\vrule \@width \@wholewidth
414 <latexrelease>\@height #1\unitlength}\@tempcnta\z@
415 <latexrelease>\put(0,0){%
416 <latexrelease> \hskip -\@halfwidth \vbox{\@whilenum \@tempcnta <\@dashcnt
417 <latexrelease> \do{\vskip #1\unitlength\copy\@dashbox
418 <latexrelease> \advance\@tempcnta\@ne }%
419 <latexrelease> \vskip\@dashdim}}\@tempcnta\z@
420 <latexrelease>\put(#2,0){%
421 <latexrelease> \hskip -\@halfwidth \vbox{\@whilenum \@tempcnta<\@dashcnt
422 <latexrelease> \do{\vskip #1\unitlength\copy\@dashbox
423 <latexrelease> \advance\@tempcnta \@ne }%
424 <latexrelease> \vskip\@dashdim}}\@makepicbox(#2,#3)}
425 <latexrelease>\EndIncludeInRelease
426 <*2kernel)

```

(End of definition for \dashbox.)

Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

CIRCLES AND OVALS

USER COMMANDS:

`\circle{D}` : Produces the circle with the diameter as close as possible to $D * \text{\unitlength}$. `\put(X,Y){\circle{D}}` puts the circle with its center at (X,Y).

`\oval(X,Y)` : Makes an oval as round as possible that fits in the rectangle of width $X * \text{\unitlength}$ and height $Y * \text{\unitlength}$. The reference point is the center.

`\oval(X,Y)[POS]` : Same as `\oval(X,Y)` except it draws only the half or quadrant of the oval indicated by POS. E.G., `\oval(X,Y)[t]` draws just the top half and `\oval(X,Y)[br]` draws just the bottom right quadrant. In all cases, the reference point is the same as the unqualified `\oval(X,Y)` command.

`\@ovvert {DELTA1} {DELTA2}` : Makes a vbox containing either the left side or the right side of the oval being constructed. The baseline will coincide with the outside bottom edge of the oval; the left side of the box will coincide with the left edge of the vertical

rule. The width of the box will be `\@tempdima`.
`DELTA1` and `DELTA2` are added to the character number in `\@tempcnta`
to get the characters for the top and bottom quarter circle pieces.

`\@ovhorz` : Makes an hbox containing the straight rule for either the
top or the bottom of the oval being constructed. The baseline
will coincide with bottom edge of the rule; the left side of
the box will coincide with the left side of the oval.
The width of the box will be `\@ovxx`.

`\@getcirc {DIAM}` : Sets `\@tempcnta` to the character number
of the top-right quarter circle with the largest
diameter less than or equal to `DIAM`.
Sets `\@tempboxa` to an hbox containing that character.
Sets `\@tempdima` to `\wd \@tempboxa`, which is the distance
from the circle's left outside edge to its right
inside edge.
(These characters are like those described in the
TeXbook, pp. 389-90.)

```
\@getcirc {DIAM} ==
BEGIN
  \@tempcnta      := integer coercion of (DIAM + 2pt)
                  + 2pt added 1 Nov 88
  \@tempcnta      := \@tempcnta / integer coercion of 4pt
  if \@tempcnta > 10
    then \@tempcnta := 10 fi
  if \@tempcnta > 0
    then \@tempcnta := \@tempcnta-1
    else LaTeX Warning: Oval too small.
  fi
  \@tempcnta      := 4 * \@tempcnta
  \@tempboxa      := \hbox{\@circlefont \char \@tempcnta}
  \@tempdima      := \wd \@tempboxa
END
```

```
\@put{X}{Y}{OBJ} ==
BEGIN
  \raise Y \hbext@ 0pt{\hskip X OBJ \hss}
END
```

```
\@oval(X,Y)[POS] ==
BEGIN
  \begingroup
  \boxmaxdepth := \maxdimen
  @ovt := @ovb := @ovl := @ovr := true
  for all E in POS
    do @ovE := false od
  \@ovxx      := X * \unitlength
  \@ovyy      := Y * \unitlength
```

```

\@tempdimb := min(\@ovxx,\@ovyy)
\@getcirc{\@tempdimb-2pt} %% "-2pt" added 7 Dec 89
\@ovro := \ht \@tempboxa
\@ovri := \dp \@tempboxa
\@ovdx := \@ovxx - \@tempdima
\@ovdx := \@ovdx/2
\@ovdy := \@ovyy - \@tempdima
\@ovdy := \@ovdy/2
\@circlefnt
\@tempboxa :=
  \hbox{
    if @ovr
      then \@ovvert{3}{2} \kern -\@tempdima
    fi
    if @ovl
      then \kern \@ovxx \@ovvert{0}{1} \kern -\@tempdima
        \kern -\@ovxx
      fi
    if @ovt
      then \@ovhorz \kern -\@ovxx
    fi
    if @ovb
      then \raise \@ovyy \@ovhorz
    fi
  }
\@ovdx := \@ovdx + \@ovro
\@ovdy := \@ovdy + \@ovro
\ht\@tempboxa := \dp\@tempboxa := 0
\@put{-\@ovdx}{-\@ovdy}{\box\@tempboxa}
\endgroup
END

```

```

\@ovvert {DELTA1} {DELTA2} ==
BEGIN
  \vbox to \@ovyy {
    if @ovb
      then \@tempcntb := \@tempcnta + DELTA1
        \kern -\@ovro
        \hbox { \char \@tempcntb }
        \nointerlineskip
      else \kern \@ovri \kern \@ovdy
    fi
    \leaders \vrule width \@wholewidth \vfil
    \nointerlineskip
    if @ovt
      then \@tempcntb := \@tempcnta + DELTA2
        \hbox { \char \@tempcntb }
      else \kern \@ovdy \kern \@ovro
    fi
  }

```

```

END

\@ovhorz ==
BEGIN
  \hb@xt@ \@ovxx{
    \kern \@ovro
    if @ovr
      then
        else \kern \@ovdx
      fi
    \leaders \hrule height \@wholewidth \hfil
    if @ovl
      then
        else \kern \@ovdx
      fi
    \kern \@ovri
  }
END

\circle{DIAM} ==
BEGIN
  \begingroup
  \boxmaxdepth := maxdimen
  \@tempdimb := DIAM *\unitlength
  if \@tempdimb > 15.5pt
    then \@getcirc{\@tempdimb}
      \@ovro := \ht \@tempboxa
      \@tempboxa := \hbox{
        \@circlefnt
        \@tempcnta := \@tempcnta + 2
        \char \@tempcnta
        \@tempcnta := \@tempcnta - 1
        \char \@tempcnta
        \kern -2\@tempdima
        \@tempcnta := \@tempcnta + 2
        \raise \@tempdima \hbox { \char \@tempcnta }
        \raise \@tempdima \box\@tempboxa
      }
      \ht\@tempboxa := \dp\@tempboxa := 0
      \@put{-\@ovro}{-\@ovro}{\@tempboxa}
    else
      \@circ{\@tempdimb}{96}
    fi
  \endgroup
END

\circle*{DIAM} == \@dot{DIAM} == \@circ{DIAM*\unitlength}{112}

\@circ{DIAM}{CHAR} ==
BEGIN

```



```

    \@tempcnta := integer coercion of (DIAM + .5pt)/1pt.
    if \@tempcnta > 15 then \@tempcnta := 15 fi
    if \@tempcnta > 1 then \@tempcnta := \@tempcnta - 1 fi
    \@tempcnta := \@tempcnta + CHAR
    \@circlefnt
    \char \@tempcnta
  END

```

End of historical L^AT_EX 2.09 comments.

`\if@ovt` If producing the Top Bottom Left or Right of an oval.

```

\if@ovb 427 \newif\if@ovt
\if@ovl 428 \newif\if@ovb
\if@ovr 429 \newif\if@ovl
        430 \newif\if@ovr

```

(End of definition for \if@ovt and others.)

```

\@ovxx
\@ovyy 431 \newdimen\@ovxx
\@ovdx 432 \newdimen\@ovyy
\@ovdy 433 \newdimen\@ovdx
\@ovro 434 \newdimen\@ovdy
\@ovri 435 \newdimen\@ovro
        436 \newdimen\@ovri

```

(End of definition for \@ovxx and others.)

`\advance\@tempdima 2pt\relax` added 1 Nov 88 to fix bug in which size of drawn circle not monotonic function of argument of `\circle`, caused by different rounding for dimensions of large and small circles.

`\@getcirc`

```

437 \def\@getcirc#1{\@tempdima #1\relax \advance\@tempdima 2\p@
438   \@tempcnta\@tempdima
439   \@tempdima 4\p@ \divide\@tempcnta\@tempdima
440   \ifnum \@tempcnta >10\relax
441     \@picture@warn
442     \@tempcnta 10\relax
443   \fi
444   \ifnum \@tempcnta >\z@ \advance\@tempcnta\m@ne
Warn if requirements for oval or circle can't be met.
445   \else \@picture@warn \fi
446   \multiply\@tempcnta 4\relax
447   \setbox \@tempboxa \hbox{\@circlefnt
448   \char \@tempcnta}\@tempdima \wd \@tempboxa}

```

(End of definition for \@getcirc.)

`\@picture@warn` Generic warning for lines, vectors (used in `\@sline`) and oval or circle (used in `\@getcirc`) are not available at right size.

```

449 \def\@picture@warn{\@latex@warning{%
450   \string\oval, \string\circle, or \string\line\space
451   size unavailable}}

```

```

(End of definition for \@picture@warn.)

\@put
452 \def\@put#1#2#3{\raise #2\hb@xt@z@{\hskip #1#3\hss}}
(End of definition for \@put.)

\oval
453 \def\oval(#1,#2){\ifnextchar[{\@oval(#1,#2)}{\@oval(#1,#2) []}}
(End of definition for \oval.)

454 \endkernel
455 \langle latexrelease \rangle \IncludeInRelease{2016/03/31}%
456 \langle latexrelease \rangle \{\@ovhlinetrue\}%
457 \langle latexrelease \rangle \{Avoid almost zero length leaders\}%
458 \endkernel | latexrelease

\if@ovvline Tests whether horizontal or vertical lines are needed.
\if@ovhline
459 \newif\if@ovvline \@ovvlinetrue
460 \newif\if@ovhline \@ovhlinetrue
461 \endkernel | latexrelease
462 \langle latexrelease \rangle \EndIncludeInRelease
463 \langle latexrelease \rangle \IncludeInRelease{0000/00/00}%
464 \langle latexrelease \rangle \{\@ovhlinetrue\}%
465 \langle latexrelease \rangle \{Avoid almost zero length leaders\}%
466 \langle latexrelease \rangle \let\if@ovvline\@undefined
467 \langle latexrelease \rangle \let\if@ovhline\@undefined
468 \langle latexrelease \rangle \EndIncludeInRelease
469 \endkernel

(End of definition for \if@ovvline and \if@ovhline.)

\@oval
470 \endkernel
471 \endkernel | latexrelease
472 \langle latexrelease \rangle \IncludeInRelease{2020/10/01}%
473 \langle latexrelease \rangle \{\@oval\}{default units}%
474 \def\@oval(#1,#2)[#3]{\begingroup\boxmaxdepth \maxdimen
475 \@ovttrue \@ovbtrue \@ovltrue \@ovrtrue

476 \@ovvlinefalse \@ovhlinefalse

477 \@tfor\reserved@a :=#3\do{%
478 \csname @ov\reserved@a false\endcsname}%
479 \@defaultunitsset\@ovxx{#1}\unitlength
480 \@defaultunitsset\@ovyy{#2}\unitlength

481 \@tempdimb \ifdim \@ovyy >\@ovxx \@ovxx \@ovxx \@ovvlinetrue
482 \else \@ovyy \ifdim \@ovyy =\@ovxx \else \@ovhlinetrue \fi\fi

483 \advance \@tempdimb -2\p@
484 \@getcirc \@tempdimb
485 \@ovro \ht\@tempboxa \ovri \dp\@tempboxa
486 \@ovdx\@ovxx \advance\@ovdx -\@tempdima \divide\@ovdx \tw@
487 \@ovdy\@ovyy \advance\@ovdy -\@tempdima \divide\@ovdy \tw@

```

```

488 \ifdim \@ovdx >\z@ \ovhlinetrue \fi
489 \ifdim \@ovdy >\z@ \ovvlinetrue \fi

490 \circlefnt \setbox\@tempboxa
491 \hbox{\if@ovr \ovvert32\kern -\@tempdima \fi
492 \if@ovl \kern \ovxx \ovvert01\kern -\@tempdima \kern -\ovxx \fi
493 \if@ovt \ovhorz \kern -\ovxx \fi
494 \if@ovb \raise \ovvy \ovhorz \fi}\advance\@ovdx\@ovro
495 \advance\@ovdy\@ovro \ht\@tempboxa\z@ \dp\@tempboxa\z@
496 \put{-\@ovdx}{-\@ovdy}{\box\@tempboxa}%
497 \endgroup}
498 (/2ekernel | latexrelease)

499 <latexrelease>\EndIncludeInRelease
500 <latexrelease>\IncludeInRelease{2016/03/31}%
501 <latexrelease> \{\@oval\}{default units}%
502 <latexrelease>\def\@oval(#1,#2)[#3]{\begingroup\boxmaxdepth \maxdimen
503 <latexrelease> \ovttrue \ovbtrue \ovltrue \ovrtrue
504 <latexrelease> \ovvlinefalse \ovhlinefalse
505 <latexrelease> \tfor\reserved@a :=#3\do{%
506 <latexrelease> \csname @ov\reserved@a false\endcsname}%
507 <latexrelease> \ovxx #1\unitlength
508 <latexrelease> \ovyy #2\unitlength
509 <latexrelease> \tempdimb \ifdim \ovyy >\ovxx \ovxx \ovvlinetrue
510 <latexrelease> \else \ovyy \ifdim \ovyy =\ovxx \else \ovhlinetrue
511 <latexrelease> \fi\fi
512 <latexrelease> \advance \tempdimb -2\p@
513 <latexrelease> \getcirc \tempdimb
514 <latexrelease> \ovro \ht\@tempboxa \ovri \dp\@tempboxa
515 <latexrelease> \ovdx\ovxx \advance\@ovdx -\@tempdima \divide\@ovdx \tw@
516 <latexrelease> \ovdy\ovvy \advance\@ovdy -\@tempdima \divide\@ovdy \tw@
517 <latexrelease> \ifdim \@ovdx >\z@ \ovhlinetrue \fi
518 <latexrelease> \ifdim \@ovdy >\z@ \ovvlinetrue \fi
519 <latexrelease> \circlefnt \setbox\@tempboxa
520 <latexrelease> \hbox{\if@ovr \ovvert32\kern -\@tempdima \fi
521 <latexrelease> \if@ovl
522 <latexrelease> \kern \ovxx \ovvert01\kern -\@tempdima \kern -\ovxx
523 <latexrelease> \fi
524 <latexrelease> \if@ovt \ovhorz \kern -\ovxx \fi
525 <latexrelease> \if@ovb \raise \ovvy \ovhorz \fi}\advance\@ovdx\@ovro
526 <latexrelease> \advance\@ovdy\@ovro \ht\@tempboxa\z@ \dp\@tempboxa\z@
527 <latexrelease> \put{-\@ovdx}{-\@ovdy}{\box\@tempboxa}%
528 <latexrelease> \endgroup}
529 <latexrelease>\EndIncludeInRelease

530 <latexrelease>\IncludeInRelease{0000/00/00}%
531 <latexrelease> \{\@oval\}{default units}%
532 <latexrelease>\def\@oval(#1,#2)[#3]{\begingroup\boxmaxdepth \maxdimen
533 <latexrelease> \ovttrue \ovbtrue \ovltrue \ovrtrue
534 <latexrelease> \tfor\reserved@a :=#3\do
535 <latexrelease> {\csname @ov\reserved@a false\endcsname}%
536 <latexrelease> \ovxx #1\unitlength
537 <latexrelease> \ovyy #2\unitlength
538 <latexrelease> \tempdimb \ifdim \ovyy >\ovxx \ovxx\else \ovyy \fi
539 <latexrelease> \advance \tempdimb -2\p@
540 <latexrelease> \getcirc \tempdimb

```

```

541 <latexrelease> \@ovro \ht\@tempboxa \@ovri \dp\@tempboxa
542 <latexrelease> \@ovdx\@ovxx \advance\@ovdx -\@tempdima \divide\@ovdx \tw@
543 <latexrelease> \@ovdy\@ovyy \advance\@ovdy -\@tempdima \divide\@ovdy \tw@
544 <latexrelease> \@circlefnt \setbox\@tempboxa
545 <latexrelease> \hbox{\if@ovr \@ovvert32\kern -\@tempdima \fi
546 <latexrelease> \if@ovl
547 <latexrelease> \kern \@ovxx \@ovvert01\kern -\@tempdima \kern -\@ovxx
548 <latexrelease> \fi
549 <latexrelease> \if@ovt \@ovhorz \kern -\@ovxx \fi
550 <latexrelease> \if@ovb \raise \@ovyy \@ovhorz \fi\advance\@ovdx\@ovro
551 <latexrelease> \advance\@ovdy\@ovro \ht\@tempboxa\z@ \dp\@tempboxa\z@
552 <latexrelease> \@put{-\@ovdx}{-\@ovdy}{\box\@tempboxa}%
553 <latexrelease> \endgroup}
554 <latexrelease> \EndIncludeInRelease
555 (*2ekernel)

```

(End of definition for \@oval.)

\@ovvert

```

556 </2ekernel>
557 <latexrelease> \IncludeInRelease{2016/03/31}%
558 <latexrelease> {\@ovvert}{Avoid almost zero length leaders}%
559 (*2ekernel | latexrelease)
560 \def\@ovvert#1#2{\vbox to\@ovyy{%
561 \if@ovb \@tempcntb \@tempcnta \advance \@tempcntb #1\relax
562 \kern -\@ovro \hbox{\char \@tempcntb}\nointerlineskip
563 \else \kern \@ovri \kern \@ovdy \fi
564
565 \if@ovvline \leaders\vrule \@width \@wholewidth \fi
566
567 \vfil \nointerlineskip
568 \if@ovt \@tempcntb \@tempcnta \advance \@tempcntb #2\relax
569 \hbox{\char \@tempcntb}%
570 \else \kern \@ovdy \kern \@ovro \fi}}
571 </2ekernel | latexrelease>
572 <latexrelease> \EndIncludeInRelease
573 <latexrelease> \IncludeInRelease{0000/00/00}%
574 <latexrelease> {\@ovvert}{Avoid almost zero length leaders}%
575 <latexrelease> \def\@ovvert#1#2{\vbox to\@ovyy{%
576 <latexrelease> \if@ovb \@tempcntb \@tempcnta \advance \@tempcntb #1\relax
577 <latexrelease> \kern -\@ovro \hbox{\char \@tempcntb}\nointerlineskip
578 <latexrelease> \else \kern \@ovri \kern \@ovdy \fi
579 <latexrelease> \leaders\vrule \@width \@wholewidth\vfil \nointerlineskip
580 <latexrelease> \if@ovt \@tempcntb \@tempcnta \advance \@tempcntb #2\relax
581 <latexrelease> \hbox{\char \@tempcntb}%
582 <latexrelease> \else \kern \@ovdy \kern \@ovro \fi}}
583 <latexrelease> \EndIncludeInRelease
584 (*2ekernel)

```

(End of definition for \@ovvert.)

\@ovhorz

```

583 </2ekernel>
584 <latexrelease> \IncludeInRelease{2016/03/31}%
585 <latexrelease> {\@ovhorz}{Avoid almost zero length leaders}%

```

```

586 <*2ekernel | latexrelease>
587 \def\@ovhorz{\hb@xt@\@ovxx{\kern \@ovro
588   \if@ovr \else \kern \@ovdx \fi
589   \if@ovhline \leaders \hrule \@height \@wholewidth \fi
590   \hfil
591   \if@ovl \else \kern \@ovdx \fi
592   \kern \@ovri}}
593 </2ekernel | latexrelease>
594 <latexrelease>\EndIncludeInRelease
595 <latexrelease>\IncludeInRelease{0000/00/00}%
596 <latexrelease>          {\@ovhorz}{Avoid almost zero length leaders}%
597 <latexrelease>\def\@ovhorz{\hb@xt@\@ovxx{\kern \@ovro
598 <latexrelease>   \if@ovr \else \kern \@ovdx \fi
599 <latexrelease>   \leaders \hrule \@height \@wholewidth \hfil
600 <latexrelease>   \if@ovl \else \kern \@ovdx \fi
601 <latexrelease>   \kern \@ovri}}
602 <latexrelease>\EndIncludeInRelease
603 <*2ekernel>

```

(End of definition for \@ovhorz.)

\circle

```
604 \def\circle{\@inmatherr\circle\@ifstar\@dot\@circle}
```

(End of definition for \circle.)

\@circle

```

605 </2ekernel>
606 <*2ekernel | latexrelease>
607 <latexrelease>\IncludeInRelease{2020/10/01}%
608 <latexrelease>          {\@circle}{default units}%
609 \def\@circle#1{%
610   \begingroup \boxmaxdepth \maxdimen
611   \@defaultunitsset\@tempdimb{#1}\unitlength
612   \ifdim \@tempdimb >15.5\p@ \@getcirc\@tempdimb
613     \@ovro\ht\@tempboxa
614     \setbox\@tempboxa\hbox{\@circlefnt
615       \advance\@tempcnta\tw@ \char \@tempcnta
616       \advance\@tempcnta\m@ne \char \@tempcnta \kern -2\@tempdima
617       \advance\@tempcnta\tw@
618       \raise \@tempdima \hbox{\char\@tempcnta}\raise \@tempdima
619       \box\@tempboxa}\ht\@tempboxa\z@ \dp\@tempboxa\z@
620       \@put{-\@ovro}{-\@ovro}{\box\@tempboxa}%
621     \else \@circ\@tempdimb{96}\fi\endgroup}
622 </2ekernel | latexrelease>
623 <latexrelease>\EndIncludeInRelease
624 <latexrelease>\IncludeInRelease{0000/00/00}%
625 <latexrelease>          {\@circle}{default units}%
626 <latexrelease>\def\@circle#1{%
627 <latexrelease>   \begingroup \boxmaxdepth \maxdimen \@tempdimb #1\unitlength
628 <latexrelease>   \ifdim \@tempdimb >15.5\p@ \@getcirc\@tempdimb
629 <latexrelease>   \ovro\ht\@tempboxa

```

```

630 <latexrelease> \setbox\@tempboxa\hbox{\@circlefnt
631 <latexrelease> \advance\@tempcnta\tw@ \char \@tempcnta
632 <latexrelease> \advance\@tempcnta\m@ne \char \@tempcnta
633 <latexrelease> \kern -2\@tempdima
634 <latexrelease> \advance\@tempcnta\tw@
635 <latexrelease> \raise \@tempdima \hbox{\char\@tempcnta}%
636 <latexrelease> \raise \@tempdima
637 <latexrelease> \box\@tempboxa\ht\@tempboxa\z@ \dp\@tempboxa\z@
638 <latexrelease> \put{-\@ovro}{-\@ovro}{\box\@tempboxa}%
639 <latexrelease> \else \@circ\@tempdimb{96}\fi\endgroup}
640 <latexrelease>\EndIncludeInRelease
641 <*2kernel>

```

(End of definition for \@circle.)

\@dot Internal form of \@circle*.

```

642 </2kernel>
643 <*2kernel | latexrelease>
644 <latexrelease>\IncludeInRelease{2020/10/01}%
645 <latexrelease> \{\@dot\}{default units}%
646 \def\@dot#1{%
647 \@defaultunitsset\@tempdimb{#1}\unitlength
648 \@circ\@tempdimb{112}}
649 </2kernel | latexrelease>
650 <latexrelease>\EndIncludeInRelease
651 <latexrelease>\IncludeInRelease{0000/00/00}%
652 <latexrelease> \{\@dot\}{default units}%
653 <latexrelease>\def\@dot#1{\@tempdimb #1\unitlength \@circ\@tempdimb{112}}
654 <latexrelease>\EndIncludeInRelease
655 <*2kernel>

```

(End of definition for \@dot.)

\@circ

```

656 \def\@circ#1#2{\@tempdima #1\relax \advance\@tempdima .5\p@
657 \@tempcnta\@tempdima \@tempdima \p@
658 \divide\@tempcnta\@tempdima
659 \ifnum\@tempcnta >15\relax \@tempcnta 15\relax \fi
660 \ifnum \@tempcnta >\z@ \advance\@tempcnta\m@ne\fi
661 \advance\@tempcnta #2\relax
662 \@circlefnt \char\@tempcnta}

```

(End of definition for \@circ.)

\@xarg Counters used for manipulating the ‘slope’ arguments.

```

\@yarg 663 \newcount\@xarg
\@yyarg 664 \newcount\@yarg
665 \newcount\@yyarg

```

(End of definition for \@xarg, \@yarg, and \@yyarg.)

\@multicnt Counter used in \multiput, and also \multicolumn.

```

666 \newcount\@multicnt

```

(End of definition for \@multicnt.)

```

\@xdim Length registers.
\@ydim 667 \newdimen\@xdim
        668 \newdimen\@ydim

(End of definition for \@xdim and \@ydim.)

\@linechar Box for holding a line segment character, for sloping lines.
        669 \newbox\@linechar

(End of definition for \@linechar.)

\@linelen Length of the line currently being built.
        670 \newdimen\@linelen

(End of definition for \@linelen.)

\@clnwd Height and width of current line segment.
\@clnht 671 \newdimen\@clnwd
        672 \newdimen\@clnht

(End of definition for \@clnwd and \@clnht.)

\@dashdim \dashbox internal registers.
\@dashbox 673 \newdimen\@dashdim
\@dashcnt 674 \newbox\@dashbox
        675 \newcount\@dashcnt

(End of definition for \@dashdim, \@dashbox, and \@dashcnt.)
Initialization: “\thinlines”
676 \let\@linefnt\tenln
677 \let\@circlefnt\tencirc
678 \@wholewidth\fontdimen8\tenln
679 \@halfwidth .5\@wholewidth

```

1.1 Curves

The new `\qbezier` command, based on the old `\bezier` defined in `bezier.sty`.
Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

```

\qbezier[N] == \bezier{N}

\bezier{N}(AX,AY)(BX,BY)(CX,CY) ==
BEGIN
  IF N = 0
  THEN \@xdima := |BX - AX|
       \@xb := |CX - BX|
       \@xa := Max(\@xa, \@xb)
       \@ya := |BY - AY|
       \@yb := |CY - BY|
       \@ya := Max(\@ya, \@yb)
       @sc := Max(\@xa, \@ya)
       %% The coefficient .5 below is the degree of overlap of
       %% successive points, where 1 is no overlap and 0 is

```

```

%% complete overlap. A coefficient of C multiplies
%% the number of points plotted by 1/C.
%%
\@xa := .5 * \@halfwidth
\@sc := \@sc / \@halfwidth
\@sc := Max(\@sc, qbeziermax)
ELSE \@sc := N
\@scp := \@sc+1
\@xb := 2 * (BX - AX) * \unitlength
\@xa := ((CX-AX)*\unitlength - \@xb)/\@sc
\@yb := 2 * (BY - AY) * \unitlength
\@ya := ((CY-AY)*\unitlength - \@yb)/\@sc
\@pictdot := square rule of width \@wholewidth
\count@ := 0
WHILE \count@ < \@scp
DO \@xdim := ((\count@*\@xa + \@xb) / \@sc) * \count@
\@ydim := ((\count@*\@ya + \@yb) / \@sc) * \count@
plot pt with relative coords (\@xdim,\@ydim)
\count@ := \count@+1
OD

```

End of historical L^AT_EX 2.09 comments.

\qbeziermax The maximum number of points to plot.

```
680 \def\qbeziermax{500}
```

(End of definition for \qbeziermax.)

In the code below, to save registers \@a ... are not used. Instead other registers are reused.

```

\newcounter{\@sc} -> \c@multicnt
\newcounter{\@scp} -> \@tempcnta
\newdimen\@xa -> \@ovxx
\newdimen\@xb -> \@ovdx
\newdimen\@ya -> \@ovyy
\newdimen\@yb -> \@ovdy
\newsavebox{\@pictdot} -> \@tempboxa

```

\qbezier Main user-level command to plot quadratic bezier curves. #2 should be (.

```
681 \newcommand\qbezier[2][0]{\bezier{#1}#2}
```

(End of definition for \qbezier.)

\bezier Form of \bezier compatible with 2.09 bezier.sty, but modified to ignore spaces between its arguments. #2 should be white space, and #4 should be (.

```
682 \def\bezier#1)#2(#3)#4({\@bezier#1)(#3){}
```



```

\@bezier 683 </2kernel>
684 <*2kernel | latexrelease>
685 <latexrelease>\IncludeInRelease{2020/10/01}%
686 <latexrelease>          {\@bezier}{default units}%
687 \def\@bezier#1(#2,#3)(#4,#5)(#6,#7){%
688   \ifnum #1=\z@
689     \@defaultunitsset\@ovxx{#4}\unitlength
690     \@defaultunitsset{\advance\@ovxx}{-#2}\unitlength
691     \ifdim \@ovxx<\z@ \@ovxx -\@ovxx \fi
692     \@defaultunitsset\@ovdx{#6}\unitlength
693     \@defaultunitsset{\advance\@ovdx}{-#4}\unitlength
694     \ifdim \@ovdx<\z@ \@ovdx -\@ovdx \fi
695     \ifdim \@ovxx<\@ovdx \@ovxx \@ovdx \fi
696     \@defaultunitsset\@ovyy{#5}\unitlength
697     \@defaultunitsset{\advance\@ovyy}{-#3}\unitlength
698     \ifdim \@ovyy<\z@ \@ovyy -\@ovyy \fi
699     \@defaultunitsset\@ovdy{#7}\unitlength
700     \@defaultunitsset{\advance\@ovdy}{-#5}\unitlength
701     \ifdim \@ovdy<\z@ \@ovdy -\@ovdy \fi
702     \ifdim \@ovyy<\@ovdy \@ovyy \@ovdy \fi
703     \@multicnt
704     \ifdim \@ovxx>\@ovyy \@ovxx \else \@ovyy \fi
705     \@ovxx .5\@halfwidth \divide\@multicnt\@ovxx
706     \ifnum \qbeziermax<\@multicnt
707       \@multicnt\qbeziermax\relax
708     \fi
709   \else \@multicnt#1\relax \fi
710   \@tempcnta\@multicnt \advance\@tempcnta\@ne
711   \@defaultunitsset\@ovdx{#4}\unitlength
712   \@defaultunitsset{\advance\@ovdx}{-#2}\unitlength
713   \multiply\@ovdx \tw@
714   \@defaultunitsset\@ovxx{#6}\unitlength
715   \@defaultunitsset{\advance\@ovxx}{-#2}\unitlength
716   \advance\@ovxx -\@ovdx \divide\@ovxx\@multicnt
717   \@defaultunitsset\@ovdy{#5}\unitlength
718   \@defaultunitsset{\advance\@ovdy}{-#3}\unitlength
719   \multiply\@ovdy \tw@
720   \@defaultunitsset\@ovyy{#7}\unitlength
721   \@defaultunitsset{\advance\@ovyy}{-#3}\unitlength
722   \advance\@ovyy -\@ovdy \divide\@ovyy\@multicnt

723   \setbox\@tempboxa\hbox{%
724     \hskip -\@halfwidth
725     \vrule \@height\@halfwidth
726     \@depth \@halfwidth
727     \@width \@wholewidth}%
728   \put(#2,#3){%
729     \count@\z@
730     \@whilenum{\count@<\@tempcnta}\do
731       {\@xdim\count@\@ovxx
732         \advance\@xdim\@ovdx
733         \divide\@xdim\@multicnt
734         \multiply\@xdim\count@

```

```

735     \ydim\count@\@ovvy
736     \advance\ydim\@ovdy
737     \divide\ydim\@multicnt
738     \multiply\ydim\count@
739     \raise \ydim
740     \hb@xt@\z@{\kern\@xdim
741         \unhcopy\@tempboxa\hss}%
742     \advance\count@\@ne}}
743 /2ekernel | latexrelease)

744 <latexrelease>\EndIncludeInRelease
745 <latexrelease>\IncludeInRelease{0000/00/00}%
746 <latexrelease>     {\@bezier}{default units}%
747 <latexrelease>\def\@bezier#1(#2,#3)(#4,#5)(#6,#7){%
748 <latexrelease>  \ifnum #1=\z@
749 <latexrelease>     \@ovxx #4\unitlength
750 <latexrelease>     \advance\@ovxx -#2\unitlength
751 <latexrelease>     \ifdim \@ovxx<\z@ \@ovxx -\@ovxx \fi
752 <latexrelease>     \@ovdx #6\unitlength
753 <latexrelease>     \advance\@ovdx -#4\unitlength
754 <latexrelease>     \ifdim \@ovdx<\z@ \@ovdx -\@ovdx \fi
755 <latexrelease>     \ifdim \@ovxx<\@ovdx \@ovxx \@ovdx \fi
756 <latexrelease>     \@ovyy #5\unitlength
757 <latexrelease>     \advance\@ovyy -#3\unitlength
758 <latexrelease>     \ifdim \@ovyy<\z@ \@ovyy -\@ovyy \fi
759 <latexrelease>     \@ovdy #7\unitlength
760 <latexrelease>     \advance\@ovdy -#5\unitlength
761 <latexrelease>     \ifdim \@ovdy<\z@ \@ovdy -\@ovdy \fi
762 <latexrelease>     \ifdim \@ovyy<\@ovdy \@ovyy \@ovdy \fi
763 <latexrelease>     \@multicnt
764 <latexrelease>     \ifdim \@ovxx>\@ovyy \@ovxx \else \@ovyy \fi
765 <latexrelease>     \@ovxx .5\@halfwidth \divide\@multicnt\@ovxx
766 <latexrelease>     \ifnum
767 <latexrelease>     \qbeziermax<\@multicnt \@multicnt\qbeziermax\relax
768 <latexrelease>     \fi
769 <latexrelease> \else \@multicnt#1\relax \fi
770 <latexrelease> \@tempcnta\@multicnt \advance\@tempcnta\@ne
771 <latexrelease> \@ovdx #4\unitlength \advance\@ovdx -#2\unitlength
772 <latexrelease>     \multiply\@ovdx \tw@
773 <latexrelease> \@ovxx #6\unitlength \advance\@ovxx -#2\unitlength
774 <latexrelease>     \advance\@ovxx -\@ovdx \divide\@ovxx\@multicnt
775 <latexrelease> \@ovdy #5\unitlength \advance\@ovdy -#3\unitlength
776 <latexrelease>     \multiply\@ovdy \tw@
777 <latexrelease> \@ovyy #7\unitlength \advance\@ovyy -#3\unitlength
778 <latexrelease>     \advance\@ovyy -\@ovdy \divide\@ovyy\@multicnt
779 <latexrelease> \setbox\@tempboxa\hbox{%
780 <latexrelease>     \hskip -\@halfwidth
781 <latexrelease>     \vrule \@height\@halfwidth
782 <latexrelease>     \@depth \@halfwidth
783 <latexrelease>     \@width \@wholewidth}%
784 <latexrelease> \put(#2,#3){%
785 <latexrelease>     \count@\z@
786 <latexrelease>     \@whilenum{\count@<\@tempcnta}\do
787 <latexrelease>     {\@xdim\count@\@ovxx
788 <latexrelease>     \advance\@xdim\@ovdx

```

```

789 <latexrelease>          \divide\@xdim\@multicnt
790 <latexrelease>          \multiply\@xdim\count@
791 <latexrelease>          \@ydim\count@\@ovvy
792 <latexrelease>          \advance\@ydim\@ovdy
793 <latexrelease>          \divide\@ydim\@multicnt
794 <latexrelease>          \multiply\@ydim\count@
795 <latexrelease>          \raise \@ydim
796 <latexrelease>          \hb@xt@\z@{\kern\@xdim
797 <latexrelease>          \unhcopy\@tempboxa\hss}%
798 <latexrelease>          \advance\count@\@ne}}
799 <latexrelease>\EndIncludeInRelease
800 <*2ekernel>

```

(End of definition for \bezier and \@bezier.)

As the commands above all use “picture” interface we couldn’t define them with \DeclareRobustCommand so we do that now.

```

801 </2ekernel>
802 <*2ekernel | latexrelease>
803 <latexrelease>\IncludeInRelease{2019/10/01}%
804 <latexrelease>          {\bezier}{Make commands robust}%
805 \MakeRobust\bezier
806 \MakeRobust\circle
807 \MakeRobust\dashbox
808 \MakeRobust\line
809 \MakeRobust\linethickness
810 \MakeRobust\multiput
811 \MakeRobust\oval
812 \MakeRobust\put
813 \MakeRobust\qbezier
814 \MakeRobust\shortstack
815 \MakeRobust\thinlines
816 \MakeRobust\vector
817 </2ekernel | latexrelease>
818 <latexrelease>\EndIncludeInRelease
819 <latexrelease>\IncludeInRelease{0000/00/00}%
820 <latexrelease>          {\bezier}{Make commands robust}%
821 <latexrelease>
822 <latexrelease>\kernel@make@fragile\bezier
823 <latexrelease>\kernel@make@fragile\circle
824 <latexrelease>\kernel@make@fragile\dashbox
825 <latexrelease>\kernel@make@fragile\line
826 <latexrelease>\kernel@make@fragile\linethickness
827 <latexrelease>\kernel@make@fragile\multiput
828 <latexrelease>\kernel@make@fragile\oval
829 <latexrelease>\kernel@make@fragile\put
830 <latexrelease>\kernel@make@fragile\qbezier
831 <latexrelease>\kernel@make@fragile\shortstack
832 <latexrelease>\kernel@make@fragile\thinlines
833 <latexrelease>\kernel@make@fragile\vector
834 <latexrelease>
835 <latexrelease>\EndIncludeInRelease
836 <*2ekernel>
837 </2ekernel>

```

File 43

ltthm.dtx

1 Theorem Environments

The user creates his own theorem-like environments with the command

```
\newtheorem{<name>}{<text>}[<counter>] or
```

```
\newtheorem{<name>}[<oldname>]{<text>}
```

This defines the environment $\langle name \rangle$ to be just as one would expect a theorem environment to be, except that it prints $\langle text \rangle$ instead of “Theorem”.

If $\langle oldname \rangle$ is given, then environments $\langle name \rangle$ and $\langle oldname \rangle$ use the same counter, so using a $\langle name \rangle$ environment advances the number of the next $\langle name \rangle$ environment, and vice-versa.

If $\langle counter \rangle$ is given, then environment $\langle name \rangle$ is numbered within $\langle counter \rangle$.

E.g., if $\langle counter \rangle = \text{subsection}$, then the first $\langle name \rangle$ in subsection 7.2 is numbered $\langle text \rangle$ 7.2.1.

The way $\langle name \rangle$ environments are numbered can be changed by redefining $\backslash the\langle name \rangle$.
Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

DOCUMENT STYLE PARAMETERS

$\backslash thmcounter\{COUNTER\}$: A command such that

```
\edef\theCOUNTER{\@thmcounter{COUNTER}}
```

defines $\backslash theCOUNTER$ to produce a number for a theorem environment.

The default is:

```
BEGIN \noexpand\arabic{COUNTER} END
```

$\backslash thmcountersep$: A separator placed between a theorem number and the number of the counter within which it is numbered.

E.g., to make the third theorem of section 7.2 be numbered 7.2-3, $\backslash thmcountersep$ should be $\backslash def$ 'ed to ‘-’. Its default is ‘.’.

$\backslash @begintheorem\{NAME\}\{NUMBER\}$: A command that begins a theorem environment for a ‘theorem’ named ‘NAME NUMBER’ –

e.g., $\backslash @begintheorem\{Lemma\}\{3.7\}$ starts Lemma 3.7.

$\backslash @opargbegintheorem\{NAME\}\{NUMBER\}\{OPARG\}$:

A command that begins a theorem environment for a ‘theorem’ named ‘NAME NUMBER’ with optional argument OPARG – e.g., $\backslash @begintheorem\{Lemma\}\{3.7\}\{Jones\}$ starts ‘Lemma 3.7 (Jones)’.

$\backslash @endtheorem$: A command that ends a theorem environment.

```
\newtheorem{NAME}{TEXT}[COUNTER] ==
```

```
BEGIN
```

```
if \NAME is definable
```

```

then \@definecounter{NAME}
  if COUNTER present
    then \@newctr{NAME}[COUNTER] fi
    \theNAME == BEGIN \theCOUNTER \@thmcountersep
                        eval\@thmcounter{NAME} END
    else \theNAME == BEGIN eval\@thmcounter{NAME} END
    \NAME == \@thm{NAME}{TEXT}
    \endNAME == \@endtheorem
  else error
fi
END

\newtheorem{NAME}[OLDNAME]{TEXT}==
BEGIN
  if counter OLDNAME nonexistent
  then ERROR
  else
    if \NAME is definable
    then BEGIN
      \theNAME == \theOLDNAME
      \NAME == \@thm{OLDNAME}{TEXT}
      \endNAME == \@endtheorem
      END
    else error
    fi
  fi
END

\@thm{NAME}{TEXT} ==
BEGIN
  \refstepcounter{NAME}
  if next char = [
    then \@ythm{NAME}{TEXT}
    else \@xthm{NAME}{TEXT}
  fi
END

\@xthm{NAME}{TEXT} ==
BEGIN
  \@begintheorem{TEXT}{\theNAME}
  \ignorespaces
END

\@ythm{NAME}{TEXT}[OPARG] ==
BEGIN
  \@opargbegintheorem{TEXT}{\theNAME}{OPARG}
  \ignorespaces
END

```

End of historical L^AT_EX 2.09 comments.

`\newtheorem` `\newtheorem` ought really be allowed only in the preamble Which would be good document style, and allow some main memory to be saved by declaring these commands to be `\@onlypreamble`. Unfortunately the L^AT_EX book indicates that `\newtheorem` may be used anywhere in the document...

```

1 <*2ekernel>
2 \def\newtheorem#1{%
3   \ifnextchar[{\@othm{#1}}{\@nthm{#1}}}
```

(End of definition for `\newtheorem`.)

`\@nthm`

```

4 \def\@nthm#1#2{%
5   \ifnextchar[{\@xnthm{#1}{#2}}{\@ynthm{#1}{#2}}}
```

(End of definition for `\@nthm`.)

`\@xnthm` 92/09/18 RmS: Changed `\@addtoreset` to `\@newctr` to produce error message if counter #3 does not exist (to be consistent with behaviour of `\newcounter`)

```

6 \def\@xnthm#1#2[#3]{%
7   \expandafter\@ifdefinable\csname #1\endcsname
8     {\@definecounter{#1}\@newctr{#1}[#3]%
9     \expandafter\xdef\csname the#1\endcsname{%
10      \expandafter\noexpand\csname the#3\endcsname \@thmcountersep
11      \@thmcounter{#1}}}%
12   \global\@namedef{#1}{\@thm{#1}{#2}}%
13   \global\@namedef{end#1}{\@endtheorem}}}
```

(End of definition for `\@xnthm`.)

`\@ynthm`

```

14 \def\@ynthm#1#2{%
15   \expandafter\@ifdefinable\csname #1\endcsname
16     {\@definecounter{#1}%
17     \expandafter\xdef\csname the#1\endcsname{\@thmcounter{#1}}%
18     \global\@namedef{#1}{\@thm{#1}{#2}}%
19     \global\@namedef{end#1}{\@endtheorem}}}
```

(End of definition for `\@ynthm`.)

`\@othm`

```

20 \def\@othm#1[#2]#3{%
21   \ifundefined{c@#2}{\@nocounterr{#2}}%
22   {\expandafter\@ifdefinable\csname #1\endcsname
23     {\global\@namedef{the#1}{\@nameuse{the#2}}%
24     \global\@namedef{#1}{\@thm{#2}{#3}}%
25     \global\@namedef{end#1}{\@endtheorem}}}
```

(End of definition for `\@othm`.)

`\@thm`

```

26 </2ekernel>
27 <*2ekernel | latexrelease>
28 <latexrelease>\IncludeInRelease{2024/03/18}%
29 <latexrelease>           {\@thm}{no link target}%
30 \def\@thm#1#2{%
```

```

31 \kernel@refstepcounter{#1}%
32 \ifnextchar[{\@ythm{#1}{#2}}{\@xthm{#1}{#2}}]{}
33 \latexrelease\EndIncludeInRelease
34 \latexrelease\IncludeInRelease{0000/00/00}%
35 \latexrelease{\@thm}{no link target}%
36 \latexrelease\def\@thm#1#2{%
37 \latexrelease \refstepcounter{#1}%
38 \latexrelease \ifnextchar[{\@ythm{#1}{#2}}{\@xthm{#1}{#2}}]{}
39 \latexrelease\EndIncludeInRelease
40 \kernel | latexrelease

```

(End of definition for \@thm.)

```

\@xthm
\@ythm
41 \kernel
42 \def\@xthm#1#2{%
43 \begintheorem{#2}{\csname the#1\endcsname}\ignorespaces}
44 \def\@ythm#1#2[#3]{%
45 \opargbegintheorem{#2}{\csname the#1\endcsname}{#3}\ignorespaces}

```

(End of definition for \@xthm and \@ythm.)

Default values

```

\@thmcounter
\@thmcountersep
46 \def\@thmcounter#1{\noexpand\arabic{#1}}
47 \def\@thmcountersep{.}

```

(End of definition for \@thmcounter and \@thmcountersep.)

```

\@begintheorem Providing theorem defaults.
\@opargbegintheorem
\@endtheorem
48 \kernel
49 \kernel | latexrelease
50 \latexrelease\IncludeInRelease{2024/03/18}%
51 \latexrelease{\@begintheorem}{add link targets}%
52 \def\@begintheorem#1#2{\trivlist
53 \item[\MakeLinkTarget{\@currentcounter}\hskip \labelsep{\bfseries #1\ #2}]\itshape}
54 \def\@opargbegintheorem#1#2#3{\trivlist
55 \item[\MakeLinkTarget{\@currentcounter}\hskip \labelsep{\bfseries #1\ #2\ (#3)}]\itshape}
56 \latexrelease\EndIncludeInRelease
57 \latexrelease\IncludeInRelease{0000/00/00}%
58 \latexrelease{\@begintheorem}{add link targets}%
59 \latexrelease\def\@begintheorem#1#2{\trivlist
60 \latexrelease \item[\hskip \labelsep{\bfseries #1\ #2}]\itshape}
61 \latexrelease\def\@opargbegintheorem#1#2#3{\trivlist
62 \latexrelease \item[\hskip \labelsep{\bfseries #1\ #2\ (#3)}]\itshape}
63 \latexrelease\EndIncludeInRelease
64 \kernel | latexrelease
65 \kernel
66 \def\@endtheorem{\endtrivlist}
67 \kernel

```

(End of definition for \@begintheorem, \@opargbegintheorem, and \@endtheorem.)

File 44

ltsect.dtx

1 Sectioning Commands

This file defines the declarations such as `\author` which are used by `\maketitle`. `\maketitle` itself is defined by each class, not in the L^AT_EX kernel.

The second part of the file defines the generic commands used for defining sectioning commands such as `\chapter`. Again the actual document level commands are defined in the class files, in terms of these commands.

```
1 <*2ekernel>
2 \message{title,}
```

1.1 The Title

`\title` The user defines the title and author by the declarations `\title{<name>}`, `\author{<name>}`
`\author` Similarly the date is declared with `\date{<date>}`.
`\date` Inside these, the `\thanks{<footnote text>}` command may be used to make acknowl-
`\thanks` edgements, notice of address, etc. in a footnote. If there are multiple authors, they have
`\and` to be separated with the `\and` command.
`\maketitle` And finally, the `\maketitle` command produces the actual title, using the informa-
tion previously saved with the other commands.

```
3 </2ekernel>
4 <*2ekernel | latexrelease>
5 <latexrelease>\IncludeInRelease{2019/10/01}%
6 <latexrelease>          {\title}{Make commands robust}%
```

`\title` `\title` for use in `\maketitle`. If not given `\maketitle` will produce an error message.

```
7 \DeclareRobustCommand\title[1]{\gdef\@title{#1}}
```

(End of definition for \title. This function is documented on page 810.)

`\author` `\author` for use in `\maketitle`. If not given `\maketitle` will produce a warning message.

```
8 \DeclareRobustCommand*\author[1]{\gdef\@author{#1}}
```

(End of definition for \author.)

`\date` `\date` for use in `\maketitle`. If not given `\maketitle` will produce `\today` as the default.

```
9 \DeclareRobustCommand*\date[1]{\gdef\@date{#1}}
```

(End of definition for \date.)

`\thanks`

```
10 \DeclareRobustCommand\thanks[1]{\footnotemark
11     \protected@xdef\@thanks{\@thanks
12         \protect\footnotetext[\the\c@footnote]{#1}}%
13 }
```

(End of definition for \thanks.)

`\and`

```
14 \DeclareRobustCommand\and{% % \begin{tabular}
15 \end{tabular}}%
16 \hskip 1em \@plus.17fil%
17 \begin{tabular}[t]{c}}% % \end{tabular}
```

(End of definition for \and.)

```
18 </2ekernel | latexrelease>
19 <latexrelease>\EndIncludeInRelease
20 <latexrelease>\IncludeInRelease{0000/00/00}%
21 <latexrelease> %\title}{Make commands robust}%
22 <latexrelease>
23 <latexrelease>\kernel@make@fragile\title
24 <latexrelease>\kernel@make@fragile\author
25 <latexrelease>\kernel@make@fragile\date
26 <latexrelease>\kernel@make@fragile\thanks
27 <latexrelease>\kernel@make@fragile\and
28 <latexrelease>
29 <latexrelease>\EndIncludeInRelease
30 <*2ekernel>
```

`\@title`

```
31 \def\@title{\@latex@error{No \noexpand\title given}\@ehc}
```

(End of definition for \@title.)

`\@author`

```
32 \def\@author{\@latex@warning@no@line{No \noexpand\author given}}
```

(End of definition for \@author.)

`\@date`

```
33 \gdef\@date{\today}
```

(End of definition for \@date.)

`\@thanks`

```
34 \let\@thanks\@empty
```

(End of definition for \@thanks.)

```
35 \message{sectioning,}
```

1.2 Sectioning

`\@secpenalty`

```
36 \newcount\@secpenalty
```

```
37 \@secpenalty = -300
```

(End of definition for \@secpenalty.)

`\if@noskipsec` Way back in 1991 (08/26) FMI & RmS set the `\@noskipsec` switch to true for the
`\@noskipsectrue` preamble and to false in `\document`. This was done to trap lists and related text in the preamble but it does not catch everything.

```
38 \newif\if@noskipsec \@noskipsectrue
```

(End of definition for `\ifnoskipsec` and `\noskipsectrue`.)

`\@startsection` The `\@startsection{<name>}{<level>}{<indent>}{<beforeskip>}{<afterskip>}{<style>}`*`[<altheading>]` `{<heading>}` command is the mother of all the user level sectioning commands. The part after the *, including the * is optional.

name: e.g., 'subsection'

level: a number, denoting depth of section – e.g., chapter = 0, section = 1, etc.

indent: Indentation of heading from left margin

beforeskip: Absolute value = skip to leave above the heading. If negative, then paragraph indent of text following heading is suppressed.

afterskip: if positive, then skip to leave below heading, else negative of skip to leave to right of run-in heading.

style: Commands to set style. Since June 1996 release the *last* command in this argument may be a command such as `\MakeUppercase` or `\fbox` that takes an argument. The section heading will be supplied as the argument to this command. So setting #6 to, say, `\bfseries\MakeUppercase` would produce bold, uppercase headings.

If '*' is missing, then increment the counter. If it is present, then there should be no `[<altheading>]` argument. The command uses the counter 'secnumdepth'. It contains a pointer to the highest section level that is to be numbered.

Warning: The `\@startsection` command should be at the same or higher grouping level as the text that follows it. For example, you should *not* do something like

```
\def\foo{ \begingroup ...
           \paragraph{...}
           \endgroup}
```

Pseudocode for the `\@startsection` command *Historical L^AT_EX 2.09 comments* (not necessarily accurate any more):

```
\@startsection
{NAME}{LEVEL}{INDENT}{BEFORESKIP}{AFTERSKIP}{STYLE} ==
BEGIN
  IF @noskipsec = T THEN \leavevmode FI
                                % true if previous section had no body.

  \par
  \@tempkipa := BEFORESKIP
  @afterindent := T
  IF \@tempkipa < 0 THEN \@tempkipa := -\@tempkipa
                        @afterindent := F
  FI
  IF @nobreak = true
    THEN \everypar == null
    ELSE \addpenalty{\@secpenalty}
         \addvspace{\@tempkipa}
  FI
  IF * next
```

```

        THEN \@ssect{INDENT}{BEFORESKIP}{AFTERSKIP}{STYLE}
        ELSE \@dblarg{\@sect
                {NAME}{LEVEL}{INDENT}
                {BEFORESKIP}{AFTERSKIP}{STYLE}}
    FI
END

```

End of historical L^AT_EX 2.09 comments.

```

39 \def\@startsection#1#2#3#4#5#6{%
40   \if@noskipsec \leavevmode \fi
41   \par
42   \@tempskipa #4\relax
43   \@afterindenttrue
44   \ifdim \@tempskipa <\z@
45     \@tempskipa -\@tempskipa \@afterindentfalse
46   \fi
47   \if@nobreak
48     \everypar{}%
49   \else
50     \addpenalty\@secpenalty\addvspace\@tempskipa
51   \fi
52   \@ifstar
53     {\@ssect{#3}{#4}{#5}{#6}}%
54     {\@dblarg{\@sect{#1}{#2}{#3}{#4}{#5}{#6}}}

```

(End of definition for \@startsection.)

`\@sect` Pseudocode for the `\@sect` command *Historical L^AT_EX 2.09 comments (not necessarily accurate any more):*

```

\@sect{NAME}{LEVEL}
      {INDENT}{BEFORESKIP}{AFTERSKIP}
      {STYLE}[ARG1]{ARG2}
      ==
BEGIN
  IF LEVEL > \c@secnumdepth
    THEN \@svsec :=L null
    ELSE \refstepcounter{NAME}
         \@svsec :=L BEGIN \@secntformat{#1}\relax END
  FI
  IF AFTERSKIP > 0
    THEN \begingroup
         STYLE
         \@hangfrom{\hskip INDENT\@svsec}
         {\interlinepenalty 10000 ARG2\par}
         \endgroup
         \NAMEmark{ARG1}
         \addcontentsline{toc}{NAME}
         { IF LEVEL > \c@secnumdepth
           ELSE \protect\numberline{\theNAME} FI
           ARG1 }
    ELSE \@svsechd == BEGIN STYLE
         \hskip INDENT\@svsec

```

```

ARG2
\NAMEmark{ARG1}
\addcontentsline{toc}{NAME}
  { IF LEVEL > \c@secnumdepth
    ELSE
      \protect\numberline{\theNAME}
    FI
  ARG1 }

      END
    FI
  \@xsect{AFTERSKIP}
END
End of historical LATEX 2.09 comments.

55 \def\@sect#1#2#3#4#5#6[#7]#8{%
56   \ifnum #2>\c@secnumdepth
57     \let\@svsec\@empty
58   \else
59     \refstepcounter{#1}%

Since \@secntformat might end with an improper \hskip which is scanning forward
for plus or minus we end the definition of \@svsec with \relax as a precaution.

60     \protected@edef\@svsec{\@secntformat{#1}\relax}%
61   \fi
62   \@tempskipa #5\relax
63   \ifdim \@tempskipa>\z@
64     \begingroup

This { used to be after the argument to \@hangfrom but was moved here to allow com-
mands such as \MakeUppercase to be used at the end of #6.

65     #6{%
66       \@hangfrom{\hskip #3\relax\@svsec}%
67       \interlinepenalty \@M #8\@par}%
68   \endgroup
69   \csname #1mark\endcsname{#7}%
70   \addcontentsline{toc}{#1}{%
71     \ifnum #2>\c@secnumdepth \else
72       \protect\numberline{\csname the#1\endcsname}%
73     \fi
74     #7}%
75   \else

\relax added 2 May 90

76   \def\@svsechd{%
77     #6{\hskip #3\relax
78     \@svsec #8}%
79     \csname #1mark\endcsname{#7}%
80     \addcontentsline{toc}{#1}{%
81       \ifnum #2>\c@secnumdepth \else
82         \protect\numberline{\csname the#1\endcsname}%
83       \fi
84       #7}}%
85   \fi
86   \@xsect{#5}}

```

(End of definition for \@sect.)

`\@xsect` Pseudocode for the `\@xsect` command *Historical L^AT_EX 2.09 comments (not necessarily accurate any more)*:

```
\@xsect{AFTERSKIP} ==
BEGIN
  IF AFTERSKIP > 0
    THEN \par \nobreak
         \vskip AFTERSKIP
         \@afterheading
    ELSE @nobreak :=G F
         @noskipsec :=G T
         \everypar{ IF @noskipsec = T
                     THEN @noskipsec :=G F
                          \clubpenalty := 10000 % local
                          \hskip -\parindent
                          \begingroup
                          \@svsechd
                          \endgroup
                          \unskip
                          \hskip -AFTERSKIP \relax
                                  %% relax added 14 Jan 91
                     ELSE \clubpenalty := \@clubpenalty % local
                          \everypar := NULL
                     FI
                   }
  FI
END
```

End of historical L^AT_EX 2.09 comments.

```
87 \def\@xsect#1{%
88   \@tempkipa #1\relax
89   \ifdim \@tempkipa>\z@
```

Why not combine `\@sect` and `\@xsect` and save doing the same test twice? It is not possible to change this now as these have become hooks!

This `\par` seems unnecessary.

```
90   \par \nobreak
91   \vskip \@tempkipa
92   \@afterheading
93   \else
94   \@nobreakfalse
95   \global\@noskipsectrue
96   \everypar{%
97     \if@noskipsec
98       \global\@noskipsecfalse
99       {\setbox\z@\lastbox}%
100      \clubpenalty\@M
101      \begingroup \@svsechd \endgroup
102      \unskip
103      \@tempkipa #1\relax
```

```

104     \hskip -\@tempskipa
105     \else
106       \clubpenalty \@clubpenalty
107       \everypar{}%
108     \fi}%
109 \fi
110 \ignorespaces}

```

(End of definition for \@xsect.)

`\@secntformat` This command formats the section number including the space following it.

```

111 \def\@secntformat#1{\csname the#1\endcsname\quad}

```

(End of definition for \@secntformat.)

Pseudocode for the `\@ssect` command *Historical L^AT_EX 2.09 comments (not necessarily accurate any more)*:

```

\@ssect{INDENT}{BEFORESKIP}{AFTERSKIP}{STYLE}{ARG} ==
BEGIN
  IF AFTERSKIP > 0
    THEN \begingroup
         STYLE
         \@hangfrom{\hskip INDENT}
              {\interlinepenalty 10000 ARG\par}
         \endgroup
    ELSE \@svsechd == BEGIN STYLE
              \hskip INDENT
              ARG
            END
  FI
  \@xsect{AFTERSKIP}
END

```

End of historical L^AT_EX 2.09 comments.

Pseudocode for the `\@afterheading` command *Historical L^AT_EX 2.09 comments (not necessarily accurate any more)*:

```

\@afterheading ==
BEGIN
  @nobreak :=G true
  \everypar := BEGIN IF @nobreak = T
                  THEN @nobreak :=G false
                   \clubpenalty := 10000 % local
                   IF @afterindent = F
                     THEN remove \lastbox
                   FI
                  ELSE \clubpenalty := \@clubpenalty % local
                   \everypar := NULL
                  FI
                END
  END

```

End of historical L^AT_EX 2.09 comments.

`\@ssect`

```
112 \def\@ssect#1#2#3#4#5{%
113   \@tempkipa #3\relax
114   \ifdim \@tempkipa>\z@
115     \begingroup
116       #4{%
117         \@hangfrom{\hskip #1}%
118         \interlinepenalty \@M #5\@par}%
119     \endgroup
120   \else
121     \def\@svsechd{#4{\hskip #1\relax #5}}%
122     \fi
123   \@xsect{#3}}
```

(End of definition for \@ssect.)

`\if@afterindent`

```
\@afterindenttrue 124 \newif\if@afterindent \@afterindenttrue
```

(End of definition for \if@afterindent and \@afterindenttrue.)

`\@afterheading`

This hook is used in setting up custom-built headings in classes.dtx.

```
125 \def\@afterheading{%
126   \@nbreaktrue
127   \everypar{%
128     \if@nbreak
129       \@nbreakfalse
130       \clubpenalty \@M
131       \if@afterindent \else
132         {\setbox\z@\lastbox}%
133       \fi
134     \else
135       \clubpenalty \@clubpenalty
136     \everypar{}%
137   \fi}}
```

(End of definition for \@afterheading.)

`\@hangfrom`

`\@hangfrom{<text>}` : Puts <text> in a box, and makes a hanging indentation of the following material up to the first `\par`. Should be used in vertical mode.

```
138 \def\@hangfrom#1{\setbox\@tempboxa\hbox{#1}}%
139   \hangindent \wd\@tempboxa\noindent\box\@tempboxa}
```

(End of definition for \@hangfrom.)

`\c@secnumdepth`

`\c@tocdepth`

```
140 \newcount\c@secnumdepth
141 \newcount\c@tocdepth
```

(End of definition for \c@secnumdepth and \c@tocdepth.)

```
\secdef \secdef{<unstarcmds>}{<unstarcmds>}{<starcmds>}
```

When defining a `\chapter` or `\section` command without using `\@startsection`, you can use `\secdef` as follows:

1. `\def\chapter{ ... \secdef \langlestarcmd\rangle \langleunstarcmd\rangle }`
2. `\def\langlestarcmd\rangle[#1]#2{ ... } % Command to define \chapter[...]{...}`
3. `\def\langleunstarcmd\rangle#1{ ... } % Command to define \chapter*{...}`

```
142 \def\secdef#1#2{\@ifstar{#2}{\@dblarg{#1}}}
```

(End of definition for `\secdef`.)

1.2.1 Initializations

```
\sectionmark
\subsectionmark 143 \let\sectionmark@gobble
\subsubsectionmark 144 \let\subsectionmark@gobble
\paragraphmark 145 \let\subsubsectionmark@gobble
\subparagraphmark 146 \let\paragraphmark@gobble
147 \let\subparagraphmark@gobble
```

(End of definition for `\sectionmark` and others.)

```
148 \message{contents,}
```

1.3 Table of Contents etc.

1.3.1 Convention

`\tf@{foo}` = file number for output for table foo. The file is opened only if `@filesw = true`.

1.3.2 Commands

A `\l@{type}{<entry>}{<page>}` Macro needs to be defined by document style for making an entry of type `<type>` in a table of contents, etc. E.g., the document style should define `\l@chapter`, `\l@section`, etc.

Note: When the `\protect` command is used in the `<entry>` or `<text>` of one of the commands below, it causes the following control sequence to be written on the file without being expanded. The sequence will be expanded when the table of contents entry is processed.

Surprise: Inside an `\addcontentsline` or `\addtocontents` command argument, the commands: `\index`, `\glossary`, and `\label` are no-ops. This could cause a problem if the user puts an `\index` or `\label` into one of the commands he writes, or into the optional ‘short version’ argument of a `\section` or `\caption` command.

```
\@starttoc The \@starttoc{<ext>} command is used to define the commands:
\tableofcontents, \listoffigures, etc.
```

For example: `\@starttoc{lof}` is used in `\listoffigures`. This command reads the `.<ext>` file and sets up to write the new `.<ext>` file.

Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

```
\@starttoc{EXT} ==
```



```

BEGIN
  \begingroup
  \makeatletter
  read file \jobname.EXT
  IF @filesw = true
    THEN open \jobname.EXT as file \tf@EXT
  FI
  @nobreak :=G FALSE %% added 24 May 89
  \endgroup
END

```

End of historical L^AT_EX 2.09 comments.

```

149 \def\@starttoc#1{%
150   \begingroup
151   \makeatletter
152   \@input{\jobname.#1}%
153   \if@filesw
154     \expandafter\newwrite\csname tf@#1\endcsname
155     \immediate\openout \csname tf@#1\endcsname \jobname.#1\relax
156   \fi
157   \@nobreakfalse
158   \endgroup}

```

(End of definition for \@starttoc.)

`\addcontentsline` The `\addcontentsline{<table>}{<type>}{<entry>}` command allows the user to add his/her own entry to a table of contents, etc. The command adds the entry `\contentsline{<type>}{<entry>}{<page>}{}` to the `.<table>` file.

This macro is implemented as an application of `\addtocontents`. Note that `\thepage` is not expandable during `\protected@write` therefore one gets the page number at the time of the `\shipout`.

```

159 </2kernel>
160 <*2kernel | latexrelease>
161 <latexrelease>\IncludeInRelease{2020/10/01}%
162 <latexrelease>          {\addcontentsline}{fourth argument}%
163 \def\addcontentsline#1#2#3{%

```

We add an empty brace pair at the end of `\contentsline` so that the number of argument is identical in documents with and without hyperref.

```

164   \addtocontents{#1}{\protect\contentsline{#2}{#3}{\thepage}}}%
165   \protected@file@percent}}
166 </2kernel | latexrelease>
167 <latexrelease>\EndIncludeInRelease
168 <latexrelease>\IncludeInRelease{2018/12/01}%
169 <latexrelease>          {\addcontentsline}{Mask line endings}%
170 <latexrelease> \def\addcontentsline#1#2#3{%
171 <latexrelease> \addtocontents{#1}{\protect\contentsline{#2}{#3}{\thepage}}%

```

We add `\protected@file@percent` at the end which is turned inside `\@writefile` into a percent character to mask the newline after the closing argument brace.

```

172 <latexrelease>          \protected@file@percent}}
173 <latexrelease>\EndIncludeInRelease
174 <latexrelease>\IncludeInRelease{0000/00/00}%
175 <latexrelease>          {\addcontentsline}{Mask line endings}%

```

```

176 <latexrelease>\def\addcontentsline#1#2#3{%
177 <latexrelease> \addtocontents{#1}{\protect\contentsline{#2}{#3}{\thepage}}
178 <latexrelease>\EndIncludeInRelease
179 <*2ekernel>

```

(End of definition for \addcontentsline.)

`\@gobble@om` Each of these four commands accepts an optional and a mandatory argument, possibly
`\@gobble@som` preceded by a star. In all cases the expansion is empty or just manipulates the spaces
`\@gobble@with@sphack@om` around the command. Used to disable commands such as `\index` or `\label` in certain
`\@gobble@with@sphack@som` situations.

```

180 </2ekernel>
181 <*2ekernel | latexrelease>
182 <latexrelease>\IncludeInRelease{2025/06/01}%
183 <latexrelease> \@gobble@som{Extended index/label}%

```

Just getting rid of the input in an expandable way is done by these two.

```

184 \DeclareExpandableDocumentCommand\@gobble@om{+o+m}{-}
185 \DeclareExpandableDocumentCommand\@gobble@som{s+o+m}{-}

```

When something needs to be suppressed during typesetting, one often also wants to ensure that this doesn't end up with two spaces (in case there is one before and one after). This can't be done expandably.

```

186 \DeclareDocumentCommand\@gobble@with@sphack@om{+o+m}{-\@bsphack\@esphack}
187 \DeclareDocumentCommand\@gobble@with@sphack@som{s+o+m}{-\@bsphack\@esphack}

```

(End of definition for \@gobble@om and others.)

`\addtocontents` The `\addtocontents{<table>}{<text>}` command adds `<text>` to the `.<table>` file, with no page number.

```

188 \long\def\addtocontents#1#2{%
189 \protected@write\@auxout
190 \let\label\@gobble@om
191 \let\index\@gobble@som
192 \let\glossary\@gobble@om}%
193 {\string\@writefile{#1}{#2}}
194 </2ekernel | latexrelease>
195 <latexrelease>\EndIncludeInRelease

196 <latexrelease>\IncludeInRelease{0000/00/00}%
197 <latexrelease> \@gobble@som{Extended index/label}%
198 <latexrelease>\long\def\addtocontents#1#2{%
199 <latexrelease> \protected@write\@auxout
200 <latexrelease> \let\label\@gobble \let\index\@gobble \let\glossary\@gobble}%
201 <latexrelease> {\string\@writefile{#1}{#2}}
202 <latexrelease>
203 <latexrelease>\let\@gobble@om\@undefined
204 <latexrelease>\let\@gobble@som\@undefined
205 <latexrelease>\let\@gobble@with@sphack@om\@undefined
206 <latexrelease>\let\@gobble@with@sphack@som\@undefined
207 <latexrelease>\EndIncludeInRelease
208 <*2ekernel>

```

(End of definition for \addtocontents.)

`\contentsline` The `\contentsline{<type>}{<entry>}{<page>}{}` macro produces a `<type>` entry in a table of contents, etc. It will appear in the `.toc` or other file. For example, the entry for subsection 1.4.3 in the table of contents, might be produced by:

```
\contentsline{subsection}
  {\numberline{1.4.3}Gnats and Gnus}{22}{}
```

The `\protect` command causes command sequences to be written without expanding them.

```
209 </2ekernel>
210 <*2ekernel | latexrelease>
211 <latexrelease>\IncludeInRelease{2021/11/15}%
212 <latexrelease>          {\contentsline}{Four arguments}%
```

In the `toc` file `\contentsline` is followed by 4 arguments these days, but only the first 3 are used in the old interface. The fourth was by default empty and only used when `hyperref` was loaded. We now pick up all 4 arguments, save the last one away in `\@contentsline@destination` and then call the old interface. This is done to simplify the interface to `hyperref` and to prepare for future changes.

```
213 \def\contentsline#1#2#3#4{\gdef\@contentsline@destination{#4}%
214          \csname l@#1\endcsname{#2}{#3}}
```

Default definition.

```
215 \let\@contentsline@destination\@empty
216 </2ekernel | latexrelease>
217 <latexrelease>\EndIncludeInRelease
218 <latexrelease>\IncludeInRelease{0000/00/00}%
219 <latexrelease>          {\contentsline}{Four arguments}%
220 <latexrelease>
221 <latexrelease>\def\contentsline#1{\csname l@#1\endcsname}
222 <latexrelease>\let\@contentsline@destination\@undefined
223 <latexrelease>\EndIncludeInRelease
224 <*2ekernel>
```

(End of definition for \contentsline.)

`\@dottedtocline{<level>}{<indent>}{<numwidth> }{<title>}{<page>}`: Macro to produce a table of contents line with the following parameters:

level If `<level> > \c@tocdepth`, then no line produced.

indent Total indentation from the left margin.

numwidth Width of box for number if the `<title>` has a `\numberline` command. As of 25 Jan 1988, this is also the amount of extra indentation added to second and later lines of a multiple line entry.

title Contents of entry.

page Page number.

Uses the following parameters, which must be set by the document style. They should be defined with `\def`'s.

`pnumwidth` Width of box in which page number is set.

`tocrmarg` Right margin indentation for all but last line of multiple-line entries.

`dotsep` Separation between dots, in mu units. Should be `\def`'d to a number like 2 or 1.7

`\@dottedtocline`

```
225 </2ekernel>
226 <*2ekernel | latexrelease>
227 <latexrelease>\IncludeInRelease{2018/12/01}%
228 <latexrelease>          {\@dottedtocline}{Prevent protrusion}%
229 \def\@dottedtocline#1#2#3#4#5{%
230   \ifnum #1>\c@tocdepth \else
231     \vskip \z@ \@plus.2\p@
232     {\leftskip #2\relax \rightskip \@tocrmarg \parfillskip -\rightskip
233     \parindent #2\relax\@afterindenttrue
234     \interlinepenalty\@M
235     \leavevmode
236     \@tempdima #3\relax
237
238     \advance\leftskip \@tempdima \null\nobreak\hskip -\leftskip
239     {#4}\nobreak
240     \leaders\hbox{$\m@th
```

If a document uses fonts other than computer modern, the use of a dot from math can be very disturbing despite the fact that this might be the only place in a document that then uses computer modern. Therefore we surround the dot with an `\hbox` to escape to the surrounding text font.

```
240     \mkern \@dotsep mu\hbox{.}\mkern \@dotsep
241     mu$}\hfill
242     \nobreak
243     \hb@xt@\@pnumwidth{\hfil\normalfont \normalcolor #5%
```

We finish off by preventing any protrusion if that is enabled. If protrusion happens the number may shift to the right and as a result you may end up with an additional dot in the toc line in some situations.

```
244         \kern-\p@\kern\p@}%
245     \par}%
246 \fi}
```

(End of definition for \@dottedtocline.)

`\noprotrusion` This command, if placed directly to the right (or left) of a word, will prevent protrusion of that word into the margin. It is used in the toc entry lines as they shouldn't protrude. It is implemented as to kerns that cancel each other but being there hide the word so that protrusion is not added. Note that a zero kern or an empty box would not work as the protrusion mechanism will skip over those.

```
247 \DeclareRobustCommand\noprotrusion{\leavevmode\kern-\p@\kern\p@}
```

(End of definition for \noprotrusion.)

```
248 </2ekernel | latexrelease>
249 <latexrelease>\EndIncludeInRelease
250 <latexrelease>\IncludeInRelease{0000/00/00}%
251 <latexrelease>          {\@dottedtocline}{Prevent protrusion}%
252 <latexrelease>\def\@dottedtocline#1#2#3#4#5{%
```

```

253 <latexrelease> \ifnum #1>\c@tocdepth \else
254 <latexrelease> \vskip \z@ \@plus.2\p@
255 <latexrelease> {\leftskip #2\relax \rightskip \@tocrmarg \parfillskip -\rightskip
256 <latexrelease> \parindent #2\relax\@afterindenttrue
257 <latexrelease> \interlinepenalty\@M
258 <latexrelease> \leavevmode
259 <latexrelease> \@tempdima #3\relax
260 <latexrelease> \advance\leftskip \@tempdima \null\nobreak\hskip -\leftskip
261 <latexrelease> {#4}\nobreak
262 <latexrelease> \leaders\hbox{$\m@th
263 <latexrelease> \mkern \@dotsep mu\hbox{.}\mkern \@dotsep
264 <latexrelease> mu$}\hfill
265 <latexrelease> \nobreak
266 <latexrelease> \hb@xt@\@pnumwidth{\hfil\normalfont \normalcolor #5}%
267 <latexrelease> \par}%
268 <latexrelease> \fi}
269 <latexrelease>
270 <latexrelease>\let\noprotrusion\undefined
271 <latexrelease>\EndIncludeInRelease
272 <*2ekernel>

```

Note: `\nobreak`'s added 7 Jan 86 to prevent bad line break that left the page number dangling by itself at left edge of a new line.

Changed 25 Jan 88 to use `\leftskip` instead of `\hangindent` so leaders of multiple-line contents entries would line up properly.

`\numberline` `\numberline{<number>}`: For use in a `\contentsline` command. It puts `<number>` flush-left in a box of width `\@tempdima` (Before 25 Jan 88 change, it also added `\@tempdima` to the hanging indentation.)

```

273 \def\numberline#1{\hb@xt@\@tempdima{#1\hfil}}
274 </2ekernel>

```

(End of definition for \numberline.)

File 45

lfloat.dtx

1 Floats

The different types of floats are identified by a $\langle type \rangle$ name, which is the name of the counter for that kind of float. For example, figures are of type ‘figure’ and tables are of type ‘table’. Each $\langle type \rangle$ has associated a positive $\langle type\ number \rangle$, which is a power of two. E.g., figures might be have type number 1, tables type number 2, programs type number 4, etc.

The locations where a float can go are specified by a $\langle placement\ specifier \rangle$, which is a list of the possible locations, each denoted by a letter as follows:

h : here — at the current location in the text.
t : top — at the top of a text page.
b : bottom — at the bottom of a text page.
p : page — on a separate float page

In addition, in conjunction with these, you can use ‘!’ which means that the current values of the float positioning parameters are ignored for this float. (Has no effect on ‘p’, float page positioning.) For example, ‘pht’ specifies that the float can appear in any of three locations: page, here or top.

1.1 Floating Environments

```
1  $\langle *2kernel \rangle$   
2  $\backslash message\{floats,\}$ 
```

Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

Where floats may appear on a page, and how many may appear there are specified by the following float placement parameters. The numbers are named like counters so the user can set them with the ordinary counter-setting commands.

```
 $\backslash c@topnumber$  : Number of floats allowed at the top of a column.  
 $\backslash topfraction$  : Fraction of column that can be devoted to floats.  
 $\backslash c@dbltopnumber, \dbltopfraction$   
: Same as above, but for double-column floats.  
 $\backslash c@bottomnumber, \bottomfraction$   
: Same as above for bottom of page.  
 $\backslash c@totalnumber$  : Number of floats allowed in a single column,  
including in-text floats.  
 $\backslash textfraction$  : Minimum fraction of column that must contain text.  
 $\backslash floatpagefraction$ : Minimum fraction of page that must be taken  
up by float page.  
 $\backslash dblfloatpagefraction$   
: Same as above, for double-column floats.
```

The document style must define the following.

`\fps@TYPE` : The default placement specifier for floats of type TYPE.

`\ftype@TYPE` : The type number for floats of type TYPE.

`\ext@TYPE` : The file extension indicating the file on which the contents list for float type TYPE is stored.
For example, `\ext@figure = 'lof'`.

`\fnum@TYPE` : A macro to generate the figure number for a caption.
For example, `\fnum@TYPE == Figure \thefigure`.

`\@makecaption{NUM}{TEXT}` :
A macro to make a caption, with NUM the value produced by `\fnum@...` and TEXT the text of the caption. It can assume it's in a `\parbox` of the appropriate width.

`\@float{TYPE}[PLACEMENT]` : This macro begins a float environment for a single-column float of type TYPE with PLACEMENT as the placement specifier. The default value of PLACEMENT is defined by `\fps@TYPE`. The environment is ended by `\end@float`.
E.g., `\figure == \@float{figure}, \endfigure == \end@float`.

```
\@float{TYPE}[PLACEMENT] ==
BEGIN
  if hmode then \@bsphack
    \@floatpenalty := -10002
  else \@floatpenalty := -10003
  fi
  \@capttype ==L TYPE
  \@dblflset
  \@fps ==L PLACEMENT
  \@onelevel@sanitize \@fps
  add default PLACEMENT if at most ! in PLACEMENT ==
\@fpsadddefault
  if inner
    then LaTeX Error: 'Not in outer paragraph mode.'
    \@floatpenalty := 0
  else if \@freelist nonempty
    then \@currbox :=L head of \@freelist
    \@freelist :=G tail of \@freelist
    \@count\@currbox :=G 32*\ftype@TYPE +
      bits determined by PLACEMENT
    else \@floatpenalty := 0
    LaTeX Error: 'Too many unprocessed floats'
  fi
  fi
  \@currbox :=G \color@vbox
```

```

\normalcolor
\ vbox{
  %% 15 Dec 87 -
  %% removed \boxmaxdepth :=L 0pt
  %% that made box 0 depth because it screwed
  %% things up. Instead, added \vskip0pt at end
  \hsize = \columnwidth
  \@parboxrestore
  \@floatboxreset
END

```

```

\caption ==
BEGIN
  \refstepcounter{\@capttype}
  \dblarg{\caption{\@capttype}}
END

```

In following definition, `\par` moved from after `\addcontentsline` to before `\addcontentsline` because the `\write` could cause an extra blank line to be added to the paragraph above the caption. (Change made 12 Jun 87)

```

\caption{TYPE}[STEXT]{TEXT} ==
BEGIN
  \par
  \addcontentsline{\ext@TYPE}{TYPE}{\numberline{\theTYPE}{STEXT}}
  \begingroup
  \@parboxrestore
  \@normalsize
  \makecaption{\fnum@TYPE}{TEXT}
  \par
  \endgroup
END

```

`\dblfloat{TYPE}[PLACEMENT]` : Macro to begin a float environment for a double-column float of type TYPE with PLACEMENT as the placement specifier. The default value of PLACEMENT is 'tp'
The environment is ended by `\enddblfloat`.

E.g., `\figure* == \dblfloat{figure}`,
`\endfigure* == \enddblfloat`.

`\dblfloat{TYPE}[PLACEMENT] ==`
Identical to `\float{TYPE}[PLACEMENT]` except `\hsize` and `\linewidth` are set to `\textwidth`.

End of historical L^AT_EX 2.09 comments.

`\floatpenalty`

```

3 \newcount\floatpenalty

```

(End of definition for \floatpenalty.)

`\caption` This is set to be an error message outside a float since no `capttype` is defined there; this may need to be changed by some classes.

```

4 \def\caption{%
5   \ifx\@capttype\undefined
6     \latex@error{\noexpand\caption outside float}\@ehd
7     \expandafter\@gobble
8   \else
9     \refstepcounter\@capttype
10    \expandafter\@firstofone
11  \fi
12  {\@dblarg{\@caption\@capttype}}%
13 }

```

(End of definition for \caption.)

`\@caption`

```

14 \long\def\@caption#1[#2]#3{%
15   \par
16   \addcontentsline{\csname ext@#1\endcsname}{#1}%
17   {\protect\numberline{\csname the#1\endcsname}{\ignorespaces #2}}%
18   \begingroup

```

The paragraph setting parameters are normalised at this point, however `\@parboxrestore` resets `\everypar` which is not correct in this context so `\@setminipage` is called if needed.

The float mechanism, like `minipage`, sets the flag `@minipage` true before executing the user-supplied text. Many L^AT_EX constructs test for this flag and do not add vertical space when it is true. The intention is that this emulates T_EX's ‘top of page’ behaviour. The flag must be set false at the start of the first paragraph. This is achieved by a redefinition of `\everypar`, but the call to `\@parboxrestore` removes that redefinition, so it is re-inserted if needed. If the flag is already false then the `\caption` was not the first entry in the float, and so some other paragraph has already activated the special `\everypar`. In this case no further action is needed.

```

19   \@parboxrestore
20   \if@minipage
21     \@setminipage
22   \fi
23   \normalsize
24   \@makecaption{\csname fnum@#1\endcsname}{\ignorespaces #3}\par
25   \endgroup}

```

(End of definition for \@caption.)

`\@float`
`\@dblflset`

```

26 \def\@float#1{%
27   \@ifnextchar[%
28     {\@xfloat{#1}}%
29     {\edef\reserved@a{\noexpand\@xfloat{#1}[\csname fps@#1\endcsname]}%
30     \reserved@a}

```

(End of definition for \@float and \@dblflset.)

`\@dblfloat`

```

31 \def\@dblfloat{%
32   \if@twocolumn\let\reserved@a\@dblft\else\let\reserved@a\@float\fi
33   \reserved@a}

```

(End of definition for `\dblfloat`.)

`\fps@dbl` Note that all double floats have default fps ‘tp’.

(End of definition for `\fps@dbl`.)

`\@setfps` This sets the fps, dealing with error conditions by adding the default.

(End of definition for `\@setfps`.)

`\@xfloat` The first part of this sets the count register that stores all the information about the type and fps of the float.

We assume here that the default specifiers already contain no active characters.

It may be better to store the defaults as numbers, rather than symbol strings.

```
34 /2kernel)
35 <latexrelease>\IncludeInRelease{2015/01/01}%
36 <latexrelease>          {\@xfloat}{Check float options}%
37 <*2kernel | latexrelease>
38 \def\@xfloat #1[#2]{%
39   \@nodocument
40   \def \@capttype {#1}%
41   \def \@fps {#2}%
42   \@onelevel@sanitize \@fps
43   \def \reserved@a {!}%
44   \ifx \reserved@a \@fps
45     \fpsadddefault
46   \else
47     \ifx \@fps \@empty
48       \fpsadddefault
49     \fi
50   \fi
51   \ifhmode
52     \bsphack
53     \@floatpenalty -\@Mii
54   \else
55     \@floatpenalty-\@Miii
56   \fi
57   \ifinner
58     \@parmoderr\@floatpenalty\z@
59   \else
60     \@next\@currbox\@freelist
61     {%
62       \@tempcnta \sixt@@n
63       \expandafter \@tfor \expandafter \reserved@a
64       \expandafter : \expandafter = \@fps
65       \do
```

Start of changes, use a nested if structure, ending in an error.

```
66     {%
67       \if \reserved@a h%
68         \ifodd \@tempcnta
69         \else
70           \advance \@tempcnta \@ne
71         \fi
```

```

72         \else\if \reserved@a t%
73             \@setfpsbit \tw@
74         \else\if \reserved@a b%
75             \@setfpsbit 4%
76         \else\if \reserved@a p%
77             \@setfpsbit 8%
78         \else\if \reserved@a !%
79             \ifnum \@tempcnta>15
80                 \advance\@tempcnta -\sixt@@n\relax
81             \fi
82         \else
83             \@latex@error{Unknown float option ‘\reserved@a’}%
84             {Option ‘\reserved@a’ ignored and ‘p’ used.}%
85             \@setfpsbit 8%
86         \fi\fi\fi\fi\fi
87     }%

```

End of changes

```

88         \@tempcntb \csname ftype@\@capttype \endcsname
89         \multiply \@tempcntb \@xxxii
90         \advance \@tempcnta \@tempcntb
91         \global \count\@currbox \@tempcnta
92     }%
93     \@fltovf
94 \fi

```

The remainder sets up the box in which the float is typeset, and the typesetting environment to be used. It is essential to have the extra box to avoid the unwanted space that would otherwise often be put at the top of the float.

It ends with a hook; not sure how useful this is but it is needed at present to deal with double-column floats.

```

95     \global \setbox\@currbox
96         \color@vbox
97         \normalcolor
98         \vbox \bgroup
99             \hsize\columnwidth
100             \@parboxrestore
101             \@floatboxreset
102 }%
103 </2kernel | latexrelease>
104 <latexrelease>\EndIncludeInRelease
105 <latexrelease>\IncludeInRelease{0000/00/00}%
106 <latexrelease>         {\@xfloat}{Check float options}%
107 <latexrelease>\def\@xfloat #1[#2]{%
108 <latexrelease> \@nodocument
109 <latexrelease> \def \@capttype {#1}%
110 <latexrelease> \def \@fps {#2}%
111 <latexrelease> \@onelevel@sanitize \@fps
112 <latexrelease> \def \reserved@b {!}%
113 <latexrelease> \ifx \reserved@b \@fps
114 <latexrelease>     \@fpsaddddefault
115 <latexrelease> \else
116 <latexrelease>     \ifx \@fps \@empty
117 <latexrelease>         \@fpsaddddefault

```

```

118 <latexrelease> \fi
119 <latexrelease> \fi
120 <latexrelease> \ifhmode
121 <latexrelease> \@bsphack
122 <latexrelease> \@floatpenalty -\@Mii
123 <latexrelease> \else
124 <latexrelease> \@floatpenalty-\@Miii
125 <latexrelease> \fi
126 <latexrelease> \ifinner
127 <latexrelease> \@parmoderr\@floatpenalty\z@
128 <latexrelease> \else
129 <latexrelease> \@next\@currbox\@freelist
130 <latexrelease> {%
131 <latexrelease> \@tempcnta \sixt@n
132 <latexrelease> \expandafter \@tfor \expandafter \reserved@a
133 <latexrelease> \expandafter : \expandafter =\@fps
134 <latexrelease> \do
135 <latexrelease> {%
136 <latexrelease> \if \reserved@a h%
137 <latexrelease> \ifodd \@tempcnta
138 <latexrelease> \else
139 <latexrelease> \advance \@tempcnta \@ne
140 <latexrelease> \fi
141 <latexrelease> \fi
142 <latexrelease> \if \reserved@a t%
143 <latexrelease> \@setfpsbit \tw@
144 <latexrelease> \fi
145 <latexrelease> \if \reserved@a b%
146 <latexrelease> \@setfpsbit 4%
147 <latexrelease> \fi
148 <latexrelease> \if \reserved@a p%
149 <latexrelease> \@setfpsbit 8%
150 <latexrelease> \fi
151 <latexrelease> \if \reserved@a !%
152 <latexrelease> \ifnum \@tempcnta>15
153 <latexrelease> \advance\@tempcnta -\sixt@n\relax
154 <latexrelease> \fi
155 <latexrelease> \fi
156 <latexrelease> }%
157 <latexrelease> \@tempcntb \csname ftype@\@capytype \endcsname
158 <latexrelease> \multiply \@tempcntb \@xxxii
159 <latexrelease> \advance \@tempcnta \@tempcntb
160 <latexrelease> \global \count\@currbox \@tempcnta
161 <latexrelease> }%
162 <latexrelease> \fltovf
163 <latexrelease> \fi
164 <latexrelease> \global \setbox\@currbox
165 <latexrelease> \color@vbox
166 <latexrelease> \normalcolor
167 <latexrelease> \vbox \bgroup
168 <latexrelease> \hsize\columnwidth
169 <latexrelease> \@parboxrestore
170 <latexrelease> \@floatboxreset
171 <latexrelease>}%

```

```
172 <latexrelease>\EndIncludeInRelease
173 <*2ekernel>
```

(End of definition for \@xfloat.)

\@floatboxreset The rationale for allowing these normally global flags to be set locally here, via **\@parboxrestore**, was stated originally by Donald Arseneau and extended by Chris Rowley. It is because these flags are only set globally to true by section commands, and these should never appear within marginals or floats or, indeed, in any group; and they are only ever set globally to false when they are definitely true.

If anyone is unhappy with this argument then both flags should be treated as in **\@setnobreak**; otherwise this command will be redundant.

```
174 \def \@floatboxreset {%
175     \reset@font
176     \normalsize
177     \@setminipage
178 }
```

(End of definition for \@floatboxreset.)

\@setnobreak

```
179 \def \@setnobreak{%
180     \if@nobreak
181         \let\outer@nobreak\@nobreaktrue
182         \@nobreakfalse
183     \fi
184 }
```

(End of definition for \@setnobreak.)

\@setminipage

```
185 \def \@setminipage{%
186     \@minipagetrue
187     \everypar{\@minipagefalse\everypar{}}%
188 }
```

(End of definition for \@setminipage.)

\end@float

```
189 \def\end@float{%
190     \@endfloatbox
191     \ifnum\@floatpenalty <\z@
```

We make sure that we never exceed **\textheight**, otherwise float will never get typeset (91/03/15 FMi).

```
192     \@largefloatcheck
193     \@cons\@currlist\@currbox
194     \ifnum\@floatpenalty <-\@Mii
195         \penalty -\@Miv
```

Saving and restoring **\prevdepth** added 26 May 87 to prevent extra vertical space when used in vertical mode.

```
196     \@tempdima\prevdepth
197     \vbox{}%
198     \prevdepth\@tempdima
```

```

199     \penalty\@floatpenalty
200     \else
201     \vadjust{\penalty -\@Miv \vbox{}}\penalty\@floatpenalty}\@Esphack
202     \fi
203     \fi
204 }

```

(End of definition for \end@float.)

\end@dblfloat

```

205 </2kernel>
206 <latexrelease>\IncludeInRelease{2015/01/01}%
207 <latexrelease>           {\end@dblfloat}{float order in 2-column}%
208 <*2kernel | latexrelease>
209 \def\end@dblfloat{%
210   \if@twocolumn
211     \endfloatbox
212     \ifnum\@floatpenalty <\z@
213       \@largefloatcheck

```

Force the depth of two column float boxes.

```

214     \global\dp\@currbox1sp %

```

What follows is essentially \end@float without a starting \endfloatbox.

```

215     \@cons\@currlist\@currbox
216     \ifnum\@floatpenalty <-\@Mii
217       \penalty -\@Miv
218       \@tempdima\prevdepth
219       \vbox{}}%
220     \prevdepth\@tempdima
221     \penalty\@floatpenalty
222     \else
223     \vadjust{\penalty -\@Miv \vbox{}}\penalty\@floatpenalty}\@Esphack
224     \fi
225   \fi
226   \else
227   \end@float
228   \fi
229 }%

```

```

230 </2kernel | latexrelease>
231 <latexrelease>\EndIncludeInRelease
232 <latexrelease>\IncludeInRelease{0000/00/00}%
233 <latexrelease>           {\end@dblfloat}{float order in 2-column}%
234 <latexrelease>\def\end@dblfloat{%
235 <latexrelease>\if@twocolumn
236 <latexrelease> \endfloatbox
237 <latexrelease> \ifnum\@floatpenalty <\z@

```

We make sure that we never exceed \textheight, otherwise float will never get typeset (91/03/15 FMi).

```

238 <latexrelease> \@largefloatcheck
239 <latexrelease> \@cons\@dbldeferlist\@currbox
240 <latexrelease> \fi

```

RmS 92/03/18 changed \@esphack to \@Esphack.

```
241 <latexrelease> \ifnum \@floatpenalty =-\@Mii \@Esphack\fi
242 <latexrelease>\else
243 <latexrelease> \end@float
244 <latexrelease>\fi
245 <latexrelease>}%
246 <latexrelease>\EndIncludeInRelease
247 <*2ekernel>
```

(End of definition for \end@dblfloat.)

\@endfloatbox This macro is not intended to be a hook; it is designed to help maintain the integrity of this code, which is used twice and, as can be seen, is subject to frequent changes.

```
248 \def \@endfloatbox{%
249     \par\vskip\z@skip      %% \par\vskip\z@ added 15 Dec 87

250     \@minipagefalse
251     \outer@nobreak
252     \egroup                %% end of vbox
253     \color@endbox
254 }
```

(End of definition for \@endfloatbox.)

\outer@nobreak

```
255 \let\outer@nobreak\@empty
```

(End of definition for \outer@nobreak.)

\@largefloatcheck This calculates by how much a float is oversize for the page and prints this in a warning message.

```
256 \def \@largefloatcheck{%
257     \ifdim \ht\@currbox>\textheight
258         \@tempdima -\textheight
259         \advance \@tempdima \ht\@currbox

260         \@latex@warning {Float too large for page by \the\@tempdima}%
261         \ht\@currbox \textheight
262     \fi
263 }
```

(End of definition for \@largefloatcheck.)

\@dbflt

```
\@xdblfloat 264 \def\@dbflt#1{\@ifnextchar[{\@xdblfloat{#1}}{\@xdblfloat{#1}[tp]}}
265 \def\@xdblfloat#1[#2]{%
266     \@xfloat{#1}[#2]\hspace\textwidth\linewidth\textwidth}
```

(End of definition for \@dbflt and \@xdblfloat.)

Moved to ltoutput 93/12/16

```
267 %\newcount\c@topnumber
268 %\newcount\c@dbltopnumber
269 %\newcount\c@bottomnumber
270 %\newcount\c@totalnumber
```

`\floatplacement` An analysis of `\@floatplacement`:
This should be called whenever `\@colht` has been set.

```

271 \def\floatplacement{\global\topnum\c@topnumber
272   % Textpage bit, global:
273   \global\@toproom \topfraction\@colht
274   \global\@botnum \c@bottomnumber
275   \global\@botroom \bottomfraction\@colht
276   \global\@colnum \c@totalnumber
277   % Floatpage bit, local:
278   \@fpmin \floatpagefraction\@colht}
279 \if2kernel

```

(End of definition for `\floatplacement`.)

`\dblfloatplacement` This should be called only within a group. Now changed to provide extra checks in `\@addtodblcol`, needed when processing a BANG float.

```

280 \if@latexrelease\IncludeInRelease{2015/01/01}%
281 \if@latexrelease \floatplacement{\float order in 2-column}%
282 \if*2kernel\if@latexrelease

```

When making two column float area, look for floats with 1sp depth.

```

283 \def\dblfloatplacement{\global\@dbltopnum\c@dbltopnumber
284   \global\@dbltoproom \dbltopfraction\@colht
285   \@textmin \@colht
286   \advance \@textmin -\@dbltoproom
287   \@fpmin \dblfloatpagefraction\textheight
288   \@fptop \@dblftop
289   \@fpsep \@dblfpsep
290   \@fpbot \@dblfpbot

```

`\f@depth` is used in `\@testwrongwidth` to look for either column or dbl-column floats. A value of 1sp signals the latter. Because of this setting here, `\dblfloatplacement` needs to be called inside a group which is a questionable design.

```

291   \def\f@depth{1sp}}%
292 \if2kernel\if@latexrelease
293 \if@latexrelease\EndIncludeInRelease
294 \if@latexrelease\IncludeInRelease{0000/00/00}%
295 \if@latexrelease \floatplacement{\float order in 2-column}%
296 \if@latexrelease\def \dblfloatplacement {%

```

Textpage bit: global, but need not be.

```

297 \if@latexrelease \global \@dbltopnum \c@dbltopnumber
298 \if@latexrelease \global \@dbltoproom \dbltopfraction\@colht

```

This new bit uses `\@textmin` to locally store the amount of extra room in the column.

```

299 \if@latexrelease \@textmin \@colht
300 \if@latexrelease \advance \@textmin -\@dbltoproom

```

Floatpage bit: must be local.

```

301 \if@latexrelease \@fpmin \dblfloatpagefraction\textheight
302 \if@latexrelease \@fptop \@dblftop
303 \if@latexrelease \@fpsep \@dblfpsep
304 \if@latexrelease \@fpbot \@dblfpbot
305 \if@latexrelease}}%
306 \if@latexrelease\EndIncludeInRelease
307 \if*2kernel

```


(End of definition for `\dblfloatplacement`.)

Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

MARGINAL NOTES:

Marginal notes use the same mechanism as floats to communicate with the `\output` routine. Marginal notes are distinguished from floats by having a negative placement specification. The command `\marginpar [LTEXT]{RTEXT}` generates a marginal note in a parbox, using LTEXT if it's on the left and RTEXT if it's on the right. (Default is RTEXT = LTEXT.) It uses the following parameters.

`\marginparwidth` : Width of marginal notes.

`\marginparsep` : Distance between marginal note and text.
the page layout to determine how to move the marginal
note into the margin. E.g., `\leftmargin` ==
`\hskip -\marginparwidth \hskip -\marginparsep` .

`\marginparpush` : Minimum vertical separation between `\marginpar`'s

Marginal notes are normally put on the outside of the page if `@mparswitch = true`, and on the right if `@mparswitch = false`. The command `\reversemarginpar` reverses the side where they are put. `\normalmarginpar` undoes `\reversemarginpar`. These commands have no effect for two-column output.

SURPRISE: if two marginal notes appear on the same line of text, then the second one could appear on the next page, in a funny position.

```
\marginpar [LTEXT]{RTEXT} ==
BEGIN
  if hmode then \@bsphack
    \@floatpenalty := -10002
  else \@floatpenalty := -10003
  fi
  if inner
    then LaTeX Error: 'Not in outer paragraph mode.'
    \@floatpenalty := 0
  else if \@freelist has two elements:
    then get \@marbox, \@currbox from \@freelist
    \count\@marbox :=G -1
    else \@floatpenalty := 0
    LaTeX Error: 'Too many unprocessed floats'
    \@currbox, \@marbox := \@tempboxa    %%use \def
  fi
fi
if optional argument
then %% \@xmpar ==
  \@savemarbox\@marbox{LTEXT}
  \@savemarbox\@currbox{RTEXT}
```

```

else %% \@ympar ==
    \@savemarbox\@marbox{RTEXT}
    \box\@currbox :=G \box\@marbox
fi
\@xympar
END

\reversemarginpar == BEGIN \@mparbottom :=G 0
    @reversemargin :=G true
    END

\normalmarginpar == BEGIN \@mparbottom :=G 0
    @reversemargin :=G false
    END

```

End of historical L^AT_EX 2.09 comments.

`\marginpar`

```

308 \def\marginpar{%
309   \ifhmode
310     \@bsphack
311     \@floatpenalty -\@Mii
312   \else
313     \@floatpenalty-\@Miii
314   \fi
315   \ifinner
316     \@parmoderr
317     \@floatpenalty\z@
318   \else
319     \@next\@currbox\@freelist{}{}%
320     \@next\@marbox\@freelist{\global\count\@marbox\m@ne}%
321     {\@floatpenalty\z@
322     \@fltovf\def\@currbox{\@tempboxa}\def\@marbox{\@tempboxa}}%
323   \fi
324   \@ifnextchar [\@xmpar\@ympar}

```

(End of definition for \marginpar.)

`\@xmpar`

```

325 \long\def\@xmpar[#1]#2{%
326   \@savemarbox\@marbox{#1}%
327   \@savemarbox\@currbox{#2}%
328   \@xympar}

```

(End of definition for \@xmpar.)

`\@ympar`

```

329 \long\def\@ympar#1{%
330   \@savemarbox\@marbox{#1}%
331   \global\setbox\@currbox\copy\@marbox
332   \@xympar}

```

(End of definition for \@ympar.)

`\@savemarbox`

```
333 </2kernel>
334 <*2kernel | latexrelease>
335 <latexrelease>\IncludeInRelease{2021/06/01}%
336 <latexrelease>          {\@savemarbox}{Explicit par for marginpar}%
337 \long\def \@savemarbox #1#2{%
338   \global\setbox #1%
339     \color@vbox
340     \vtop{%
341       \hsize\marginparwidth
342       \@parboxrestore
343       \@marginparreset
344       #2\par
345       \@minipagefalse
346       \outer@nobreak
347     }%
348   \color@endbox
349 }
350 </2kernel | latexrelease>
351 <latexrelease>\EndIncludeInRelease
352 <latexrelease>\IncludeInRelease{0000/00/00}%
353 <latexrelease>          {\@savemarbox}{Explicit par for marginpar}%
354 <latexrelease>
355 <latexrelease>\long\def \@savemarbox #1#2{%
356 <latexrelease> \global\setbox #1%
357 <latexrelease>   \color@vbox
358 <latexrelease>   \vtop{%
359 <latexrelease>     \hsize\marginparwidth
360 <latexrelease>     \@parboxrestore
361 <latexrelease>     \@marginparreset
362 <latexrelease>     #2%
363 <latexrelease>     \@minipagefalse
364 <latexrelease>     \outer@nobreak
365 <latexrelease>     }%
366 <latexrelease>   \color@endbox
367 <latexrelease>}
368 <latexrelease>\EndIncludeInRelease
369 <*2kernel>
```

(End of definition for \@savemarbox.)

`\@marginparreset`

The rationale for allowing these normally global flags to be set locally here, via `\@parboxrestore` was stated originally by Donald Arseneau and extended by Chris Rowley. It is because these flags are only set globally to true by section commands, and these should never appear within marginals or floats or, indeed, in any group; and they are only ever set globally to false when they are definitely true.

If anyone is unhappy with this argument then both flags should be treated as in `\@setnobreak`; otherwise this command will be redundant.

```
370 \def \@marginparreset {%
371   \reset@font
372   \normalsize
373 %   \let\if@nobreak\iffalse
374 %   \let\if@noskipsec\iffalse
```

```

375 %      \@setnobreak
376      \@setminipage
377 }

```

(End of definition for \@marginparreset.)

`\@xympar`

Setting the box here is done only because the code uses `\end@float`; it will be empty and gets discarded.

```

378 \def \@xympar{%
379   \ifnum \@floatpenalty < \z@ \@cons \@currlist \@marbox \fi
380   \setbox \@tempboxa
381     \color@vbox
382     \vbox \bgroup
383   \end@float
384   \@ignorefalse
385   \@esphack
386 }

```

(End of definition for \@xympar.)

`\reversemarginpar`

`\normalmarginpar`

```

387 \def \reversemarginpar {\global \@parbottom \z@ \@reversemargintrue}
388 \def \normalmarginpar {\global \@parbottom \z@ \@reversemarginfalse}

```

(End of definition for `\reversemarginpar` and `\normalmarginpar`.)

```

389 \message{footnotes,}

```

1.2 Footnotes

Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

`\footnote{NOTE}` : User command to insert a footnote.

`\footnote[NUM]{NOTE}`: User command to insert a footnote numbered *NUM*, where *NUM* is a number – 1, 2, etc. For example, if footnotes are numbered *, **, etc. within pages, then `\footnote[2]{...}` produces footnote '**'. This command does not step the footnote counter.

`\footnotemark[NUM]` : Command to produce just the footnote mark in the text, but no footnote. With no argument, it steps the footnote counter before generating the mark.

`\footnotetext[NUM]{TEXT}` : Command to produce the footnote but no mark. `\footnote` is equivalent to `\footnotemark \footnotetext` .

As in PLAIN, footnotes use `\insert\footins`, and the following parameters:

`\footnotesize` : Size-changing command for footnotes.

`\footnotesep` : The height of a strut placed at the beginning of every footnote.

`\skip\footins` : Space between main text and footnotes. The rule separating footnotes from text occurs in this space. This space lies above the strut of height `\footnotesep` which is at the beginning of the first footnote.

`\footnoterule` : Macro to draw the rule separating footnotes from text. It is executed right after a `\vspace` of `\skip\footins`. It should take zero vertical space—i.e., it should to a negative skip to compensate for any positive space it occupies. (See PLAIN.TEX.)

`\interfootnotelinepenalty` : Interline penalty for footnotes.

`\thefootnote` : In usual LaTeX style, produces the footnote number. If footnotes are to be numbered within pages, then the document style file must include an `\@addtoreset` command to cause the footnote counter to be reset when the page counter is stepped. This is not a good idea, though, because the counter will not always be reset in time to ensure that the first footnote on a page is footnote number one.

`\@thefnmark` : Holds the current footnote’s mark—e.g., `\dag` or `'1'` or `'a'`.

`\@mpfnnumber` : A macro that generates the numbers for `\footnote` and `\footnotemark` commands. It == `\thefootnote` outside a minipage environment, but can be changed inside to generate numbers for `\footnote`’s.

`\@makefnmark` : A macro to generate the footnote marker from `\@thefnmark`. The default definition was `\hbox{${\@thefnmark}}`.

This is now replaced by `\@thefnmark`

`\@makefntext{NOTE}` :
 Must produce the actual footnote, using `\@thefnmark` as the mark of the footnote and `NOTE` as the text. It is called when effectively inside a `\parbox`, with `\hsize = \columnwidth`.
 For example, it might be as simple as
`${\@thefnmark}$ NOTE`

In a minipage environment, `\footnote` and `\footnotetext` are redefined so that

- (a) they use the counter `mpfootnote`
- (b) the footnotes they produce go at the bottom of the minipage.

The switch is accomplished by letting `\@mpfn == footnote` or `mpfootnote` and `\thempfn == \thefootnote` or `\thempfootnote`, and by redefining `\@footnotetext` to be `\@mpfootnotetext` in the minipage.

```

\footnote{NOTE} ==
BEGIN
  \stepcounter{\@mpfn}
  begingroup
    \protect == \noexpand
    \@thefnmark :=G eval(\thempfn)
  endgroup
  \@footnotemark
  \@footnotetext{NOTE}
END

\footnote[NUM]{NOTE} ==
BEGIN
  begingroup
    \protect == \noexpand
    counter \@mpfn :=L NUM
    \@thefnmark :=G eval(\thempfn)
  endgroup
  \@footnotemark
  \@footnotetext{NOTE}
END

\footnotemark ==
BEGIN \stepcounter{footnote}
  begingroup
    \protect == \noexpand
    \@thefnmark :=G eval(\thefootnote)
  endgroup
  \@footnotemark
END

\footnotemark[NUM] ==
BEGIN
  begingroup
    footnote counter :=L NUM
    \protect == \noexpand
    \@thefnmark :=G eval(\thefootnote)
  endgroup
  \@footnotemark
END

```

```

\@footnotemark ==
  BEGIN
  \leavevmode
  IF hmode THEN \@x@sf := \the\spacefactor FI
  \@makefnmark      % put number in main text
  IF hmode THEN \spacefactor := \@x@sf FI
  END

\footnotetext      ==
  BEGIN begingroup \protect == \noexpand
                \@thefnmark :=G eval (\thempfn)
  endgroup
  \@footnotetext
  END

\footnotetext[NUM] ==
  BEGIN begingroup counter \@mpfn :=L NUM
                \protect == \noexpand
                \@thefnmark :=G eval (\thempfn)
  endgroup
  \@footnotetext
  END

```

End of historical L^AT_EX 2.09 comments.

`\footins` L^AT_EX does use the same insert for footnotes as PLAIN.

```
390 \newinsert\footins
```

L^AT_EX leaves these initializations for the `\footins` insert.

```
391 \skip\footins=\bigskipamount % space added when footnote is present
```

```
392 \count\footins=1000 % footnote magnification factor (1 to 1)
```

```
393 \dimen\footins=8in % maximum footnotes per page
```

(End of definition for \footins.)

`\footnoterule` L^AT_EX keeps PLAIN T_EX's `\footnoterule` as the default.

```
394 \def\footnoterule{\kern-3\p@
```

```
395   \hrule \@width 2in \kern 2.6\p@} % the \hrule is .4pt high
```

(End of definition for \footnoterule.)

`\thefootnote`

```
396 \@definecounter{footnote}
```

```
397 \def\thefootnote{\@arabic\c@footnote}
```

(End of definition for \thefootnote.)

`\thempfootnote` The default display for the footnote counter in minipages is to use italic letters. We use `\itshape` not `\textit` as the latter would add an italic correction.

```
398 \@definecounter{mpfootnote}
```

```
399 \def\thempfootnote{{\itshape\@alph\c@mpfootnote}}
```

(End of definition for \thempfootnote.)

`\@makefnmark` Default definition.

```
400 %\def\@makefnmark{\hbox{$^{\@thefnmark}\m@th$}}
401 \def\@makefnmark{\hbox{\@textsuperscript{\normalfont\@thefnmark}}}
```

(End of definition for \@makefnmark.)

`\textsuperscript` This command provides superscript characters in the current text font. It's implementation might change!!

```
402 \DeclareRobustCommand*\textsuperscript[1]{%
403   \@textsuperscript{\selectfont#1}}
```

(End of definition for \textsuperscript.)

`\@textsuperscript` This command should not be used directly, but may be used to define other commands `\textsuperscript`, `\@makefnmark`. #1 should always start with a font selection command, to activate the font size switch.

```
404 \</2ekernel>
405 <*2ekernel | latexrelease>
406 <latexrelease>\IncludeInRelease{2020/10/01}%
407 <latexrelease>           {\@textsuperscript}{superscript baseline}%
408 \def\@textsuperscript#1{%
409   {\m@th\ensuremath{\^{\mbox{\fontsize\sf@size\sf@size#1}}}}
410 \</2ekernel | latexrelease>
411 <latexrelease>\EndIncludeInRelease
412 <latexrelease>\IncludeInRelease{0000/00/00}%
413 <latexrelease>           {\@textsuperscript}{superscript baseline}%
414 <latexrelease>
415 <latexrelease>\def\@textsuperscript#1{%
416 <latexrelease> {\m@th\ensuremath{\^{\mbox{\fontsize\sf@size\z@#1}}}}
417 <latexrelease>\EndIncludeInRelease
418 <*2ekernel>
```

(End of definition for \@textsuperscript.)

`\textsubscript`

```
419 \</2ekernel>
420 <latexrelease>\IncludeInRelease{2015/01/01}%
421 <latexrelease>           {\textsubscript}{\textsubscript}%
422 <*2ekernel | latexrelease>
423 \DeclareRobustCommand*\textsubscript[1]{%
424   \@textsubscript{\selectfont#1}}%
425 \</2ekernel | latexrelease>
426 <latexrelease>\EndIncludeInRelease
427 <latexrelease>\IncludeInRelease{0000/00/00}%
428 <latexrelease>           {\textsubscript}{\textsubscript}%
429 <latexrelease>\let\textsubscript\@undefined
430 <latexrelease>\EndIncludeInRelease
431 <*2ekernel>
```

(End of definition for \textsubscript.)

`\@textsubscript`

```
432 </2ekernel>
433 <*2ekernel | latexrelease>
434 <latexrelease>\IncludeInRelease{2020/10/01}%
435 <latexrelease>          {\@textsubscript}{subscript baseline}%
436 \def\@textsubscript#1{%
437   {\m@th\ensuremath_{\mbox{\fontsize\sf@size\sf@size#1}}}}
438 </2ekernel | latexrelease>
439 <latexrelease>\EndIncludeInRelease

440 <latexrelease>\IncludeInRelease{2015/01/01}%
441 <latexrelease>          {\@textsubscript}{subscript baseline}%
442 <latexrelease>
443 <latexrelease>\def\@textsubscript#1{%
444 <latexrelease>   {\m@th\ensuremath_{\mbox{\fontsize\sf@size\z@#1}}}}
445 <latexrelease>\EndIncludeInRelease
446 <latexrelease>\IncludeInRelease{0000/00/00}%
447 <latexrelease>          {\@textsubscript}{subscript baseline}%
448 <latexrelease>\let\@textsubscript\@undefined
449 <latexrelease>\EndIncludeInRelease
450 <*2ekernel>
```

(End of definition for \@textsubscript.)

`\footnotesep`

```
451 \newdimen\footnotesep
```

(End of definition for \footnotesep.)

`\footnote`

```
452 \def\footnote{\@ifnextchar [\@xfootnote{\stepcounter\@mpfn
453   \protected@xdef\@thefnmark{\thempfn}%
454   \@footnotemark\@footnotetext}}
```

(End of definition for \footnote.)

`\@xfootnote`

```
455 \def\@xfootnote[#1]{%
456   \begingroup
457   \csname c@\@mpfn\endcsname #1\relax
458   \unrestored@protected@xdef\@thefnmark{\thempfn}%
459   \endgroup
460   \@footnotemark\@footnotetext}
```

(End of definition for \@xfootnote.)

`\@footnotetext`

```
461 </2ekernel>
462 <*2ekernel | latexrelease>
463 <latexrelease>\IncludeInRelease{2021/11/15}%
464 <latexrelease>          {\@footnotetext}{footnotetext tagging}%
465 \long\def\@footnotetext#1{\insert\footins{%
466   \reset@font\footnotesize
467   \interlinepenalty\interfootnotelinepenalty
468   \splittopskip\footnotesep
```

```

469 \splitmaxdepth \dp\strutbox \floatingpenalty \@MM
470 \hsize\columnwidth \@parboxrestore
471 \def\@currentcounter{footnote}%
472 \protected@edef\@currentlabel{%
473 \csname p@footnote\endcsname\@thefnmark
474 }%
475 \color@begingroup
476 \makefnfntext{%
477 \rule\z@\footnotesep\ignorespaces#1\@finalstrut\strutbox}%
478 \par
479 \color@endgroup}}%
480 </2ekernel | latexrelease>
481 <latexrelease>\EndIncludeInRelease

482 <latexrelease>\IncludeInRelease{2021/06/01}%
483 <latexrelease> \{\@footnotetext\}{footnotetext tagging}%
484 <latexrelease>\long\def\@footnotetext#1{\insert\footins{%
485 <latexrelease> \reset@font\footnotesize
486 <latexrelease> \interlinepenalty\interfootnotelinepenalty
487 <latexrelease> \splittopskip\footnotesep
488 <latexrelease> \splitmaxdepth \dp\strutbox \floatingpenalty \@MM
489 <latexrelease> \hsize\columnwidth \@parboxrestore
490 <latexrelease> \protected@edef\@currentlabel{%
491 <latexrelease> \csname p@footnote\endcsname\@thefnmark
492 <latexrelease> }%
493 <latexrelease> \color@begingroup
494 <latexrelease> \makefnfntext{%
495 <latexrelease> \rule\z@\footnotesep\ignorespaces#1\@finalstrut\strutbox}%
496 <latexrelease> \par
497 <latexrelease> \color@endgroup}}%
498 <latexrelease>\EndIncludeInRelease

499 <latexrelease>\IncludeInRelease{0000/00/00}%
500 <latexrelease> \{\@footnotetext\}{footnotetext tagging}%
501 <latexrelease>
502 <latexrelease>\long\def\@footnotetext#1{\insert\footins{%
503 <latexrelease> \reset@font\footnotesize
504 <latexrelease> \interlinepenalty\interfootnotelinepenalty
505 <latexrelease> \splittopskip\footnotesep
506 <latexrelease> \splitmaxdepth \dp\strutbox \floatingpenalty \@MM
507 <latexrelease> \hsize\columnwidth \@parboxrestore
508 <latexrelease> \protected@edef\@currentlabel{%
509 <latexrelease> \csname p@footnote\endcsname\@thefnmark
510 <latexrelease> }%
511 <latexrelease> \color@begingroup
512 <latexrelease> \makefnfntext{%
513 <latexrelease> \rule\z@\footnotesep\ignorespaces#1\@finalstrut\strutbox}%
514 <latexrelease> \color@endgroup}}%
515 <latexrelease>
516 <latexrelease>\EndIncludeInRelease
517 <*2ekernel>

```

(End of definition for \@footnotetext.)

\footnotemark

```

518 \def\footnotemark{%
519   \@ifnextchar[\@xfootnotemark
520     {\stepcounter{footnote}%
521     \protected@xdef\@thefnmark{\thefootnote}%
522     \@footnotemark}}

```

(End of definition for \footnotemark.)

\@xfootnotemark

```

523 \def\@xfootnotemark[#1]{%
524   \begingroup
525     \c@footnote #1\relax
526     \unrestored@protected@xdef\@thefnmark{\thefootnote}%
527   \endgroup
528   \@footnotemark}

```

(End of definition for \@xfootnotemark.)

\@footnotemark

```

529 \def\@footnotemark{%
530   \leavevmode
531   \ifhmode\edef\@x@sf{\the\spacefactor}\nobreak\fi
532   \@makefnmark
533   \ifhmode\spacefactor\@x@sf\fi
534   \relax}

```

(End of definition for \@footnotemark.)

\footnotetext

```

535 \def\footnotetext{%
536   \@ifnextchar [\@xfootnotenext
537     {\protected@xdef\@thefnmark{\thempfn}%
538     \@footnotetext}}

```

(End of definition for \footnotetext.)

\@xfootnotenext

```

539 \def\@xfootnotenext[#1]{%
540   \begingroup
541     \csname c@\mpfn\endcsname #1\relax
542     \unrestored@protected@xdef\@thefnmark{\thempfn}%
543   \endgroup
544   \@footnotetext}

```

(End of definition for \@xfootnotenext.)

\thempfn

\@mpfn

```

545 \def\@mpfn{footnote}
546 \def\thempfn{\thefootnote}

```

(End of definition for \thempfn and \@mpfn.)

`\footref` This command generates a footnote mark. The value is produced by referencing a `\label` placed into a `\footnote` elsewhere (can be one in the main galley or in a minipage).

```
547 </2ekernel>
548 <*2ekernel | latexrelease>
549 <latexrelease>\IncludeInRelease{2021/06/01}%
550 <latexrelease>          {\footref}{Add footref}%
551 \def\footref#1{%
552   \begingroup
553   \unrestored@protected@xdef\@thefnmark{\ref{#1}}%
554   \endgroup
555   \@footnotemark
556 }
557 </2ekernel | latexrelease>
558 <latexrelease>\EndIncludeInRelease
```

We don't remove it when rolling back so that packages offered it in the past do not need to alter their behavior in a rollback situation.

```
559 <latexrelease>\IncludeInRelease{0000/00/00}%
560 <latexrelease>          {\footref}{Add footref}%
561 <latexrelease>
562 <latexrelease> % \let\footref\undefined
563 <latexrelease>
564 <latexrelease>\EndIncludeInRelease
565 <*2ekernel>
```

(End of definition for \footref.)

```
566 </2ekernel>
```

File 46

ltxglo.dtx

1 Index and Glossary Generation

Index and Glossary commands.

```
\makeindex      A preamble command to turn on indexing.
\makeglossary   A preamble command to turn on making glossary entries.
  \index        Make an index entry for #1.
  \glossary     Make a glossary entry for #1.
```

Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

```
\makeindex ==
  BEGIN
    \index == BEGIN \@bsphack
              \begingroup
              \protect{X} == \string X\space
              %% added 3 Feb 87 for \index commands
              %% in \footnotes
              re-\catcode special characters
              to 'other'
              \@wrindex
  END
```

```
\@wrindex{ITEM} ==
  BEGIN
    write of {\indexentry{ITEM}{page number}}
  \endgroup
  \@esphack
  END
```

INITIALIZATION:

```
\index == BEGIN \@bsphack
           \begingroup
           re-\catcode special characters (in case '%' there)
           \@index
  END
```

```
\@index{ITEM} == BEGIN \endgroup \@esphack END
```

Changes made 14 Apr 89 to write \glossaryentry's instead of
\indexentry's on the .glo file.

End of historical L^AT_EX 2.09 comments.

```
1 <*2kernel>
2 \message{index,}
```

`\makeindex`

```
3 \def\makeindex{%
4   \newwrite\@indexfile
5   \immediate\openout\@indexfile=\jobname.idx
6   \def\index{\@bsphack\begingroup
7     \@sanitize
8     \@windex}\typeout
9     {Writing index file \jobname.idx}}%
```

Opening the write channel should be done only once since on some OS multiple opens are forbidden and in any case it is useless. So we turn this into a no-op after use.

```
10 \let\makeindex\@empty
11 }
12 \@onlypreamble\makeindex
```

(End of definition for \makeindex.)

`\@windex`

```
13 \def\@windex#1{%
14   \protected@write\@indexfile{%
15     {\string\indexentry{#1}{\thepage}}%
16   \endgroup
17   \@esphack}
```

(End of definition for \@windex.)

`\index`

```
18 \def\index{\@bsphack\begingroup \@sanitize\@index}
```

(End of definition for \index.)

`\@index`

```
19 \def\@index#1{\endgroup\@esphack}
```

(End of definition for \@index.)

`\makeglossary`

```
20 \def\makeglossary{%
21   \newwrite\@glossaryfile
22   \immediate\openout\@glossaryfile=\jobname.glo
23   \def\glossary{\@bsphack\begingroup
24     \@sanitize
25     \@wrglossary}\typeout
26     {Writing glossary file \jobname.glo }}%
```

Opening the write channel should be done only once since on some OS multiple opens are forbidden and in any case it is useless. So we turn this into a no-op after use.

```
27 \let\makeglossary\@empty
28 }
29 \@onlypreamble\makeglossary
```

(End of definition for \makeglossary.)

`\@wrglossary`

```
30 \def\@wrglossary#1{%  
31   \protected@write\@glossaryfile{%  
32     {\string\glossaryentry{#1}{\thepage}}%  
33   \endgroup  
34   \@esphack}
```

(End of definition for \@wrglossary.)

`\glossary`

```
35 \def\glossary{\@bsphack\begin@group\@sanitize\@index}
```

(End of definition for \glossary.)

```
36 \</2kernel)
```

File 47

ltbibl.dtx

1 Bibliography Generation

A bibliography is created by the `thebibliography` environment, which generates a title such as “References”, and a list of entries. The `BIBTEX` program will create a file containing such an environment, which will be read in by the `\bibliography` command. With `BIBTEX`, the following commands will be used.

`\bibliography` `\bibliography{<file1,file2, ...,filen>}` : specifies the bibdata files. Writes a `\bibdata` entry on the `.aux` file and tries to read in `mainfile.bbl`.

`\bibliographystyle` `\bibliographystyle{<style>}` : Writes a `\bibstyle` entry on the `.aux` file.

`thebibliography (env.)` The `thebibliography` environment is a list environment. To save the use of an extra counter, it should use `enumiv` as the item counter. Instead of using `\item`, items in the bibliography are produced by the following commands:

`\bibitem{<name>}` : Produces a numbered entry cited as `<name>`.

`\bibitem[<label>]{<name>}` : Produces an entry labeled by `<Label>` and cited by `<name>`.
The former is used for bibliographies with citations like [1], [2], etc.; the latter is used for citations like [Knuth82].

The document class must define the `thebibliography` environment. This environment has a single argument, which is the widest bibliography label— e.g., if the [Knuth67] is the widest entry, then this argument will be Knuth67. The `\thebibliography` command must begin a list environment, which the `\endthebibliography` command ends.

`\cite` Entries are cited by the command `\cite{<name>}`.

`\nocite` `\nocite{<citations>}` puts information on the `.aux` file that causes `BIBTEX` to include the `{<citations>}` list in the bibliography, but puts nothing in the text.
`\nocite{*}` is special: it tells `BIBTEX` to put the whole of a collection of references into the bibliography.

```
1 <*2kernel>
2 \message{bibliography,}
```

Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

PARAMETERS

`@cite` : A macro such that `\@cite{LABEL1,LABEL2}{NOTE}` produces the output for a `\cite[NOTE]{FOO1,FOO2}` command, where entry `FOOi` is defined by `\bibitem[LABELi]{FOOi}`. The switch `@tempswa` is true if the optional `NOTE` argument is present.

The default definition is :

```
\@cite{LABELS}{NOTE} ==
  BEGIN [LABELS
    IF @tempswa = T THEN , NOTE FI
  ]
END
```

`@biblabel` : A macro to produce the label in the bibliography entry. For `\bibitem[LABEL]{NAME}`, the label is

generated by `\@biblabel{LABEL}`. It has the default definition `\@biblabel{LABEL} -> [LABEL]`.

CONVENTION

`\b@FOO` : The name or number of the reference created by `\cite{FOO}`
E.g., if `\cite{FOO} -> [17]` , then `\b@FOO -> 17`.

End of historical L^AT_EX 2.09 comments.

`\bibitem`

```
3 \def\bibitem{\@ifnextchar[\@lbibitem\@bibitem}
```

(End of definition for \bibitem.)

`\@lbibitem`

```
4 \def\@lbibitem[#1]#2{\item[\@biblabel{#1}\hfill]\if@filesw
5     {\let\protect\noexpand
6     \immediate
7     \write\@auxout{\string\bibcite{#2}{#1}}\fi\ignorespaces}
```

(End of definition for \@lbibitem.)

`\@bibitem`

```
8 \def\@bibitem#1{\item\if@filesw \immediate\write\@auxout
9     {\string\bibcite{#1}{\the\value{\@listctr}}}\fi\ignorespaces}
```

(End of definition for \@bibitem.)

`\bibcite`

```
10 \def\bibcite{\@newl@bel b}
```

(End of definition for \bibcite.)

`\citation`

```
11 \let\citation\@gobble
```

(End of definition for \citation.)

`\cite`

```
12 \</2ekernel>
13 \< *2ekernel | latexrelease>
14 \< latexrelease>\IncludeInRelease{2022/06/01}%
15 \< latexrelease> \cite}{check for blank}%
16 \DeclareRobustCommand\cite{%
17   \@ifnextchar [{\@tempswatrue\@citex@checkblank}{\@tempswafalse\@citex@checkblank []}]}
```

Due to the way `\@for` as used in `\@citex` behaves an empty argument to `\cite` did not produce any warning for a missing citation. So we now inject a command before calling `\@citex` that does the checking for us. It is not done in `\@citex` directly, because that command is altered by a number of packages/classes and this way it is more likely that the check survives.

```
18 \def\@citex@checkblank[#1]#2{%
19   \IfBlankTF {#2}%
20     {\@citex[#1]{\space}}%
21     {\@citex[#1]{#2}}%
22 }
23 \</2ekernel | latexrelease>
```

```

24 <latexrelease>\EndIncludeInRelease
25 <latexrelease>\IncludeInRelease{0000/00/00}%
26 <latexrelease>          {\cite}{check for blank}%
27 <latexrelease>
28 <latexrelease>\DeclareRobustCommand\cite{%
29 <latexrelease> \ifnextchar [{\@tempswatrue\@citex}{\@tempswafalse\@citex []}]
30 <latexrelease>\let\@citex@checkblank\undefined
31 <latexrelease>
32 <latexrelease>\EndIncludeInRelease
33 <*2ekernel>

```

(End of definition for \cite.)

`\@citex` `\penalty\@m` added to definition of `\@citex` to allow a line break after the ‘;’ in citations like [Jones80,Smith77] (Added 23 Oct 86)
space added after the ‘;’ (21 Nov 87)

```

34 \def\@citex[#1]#2{\leavevmode
35   \let\@citea\@empty
36   \@cite{\@for\@citeb:=#2\do
37     {\@citea\def\@citea{,\penalty\@m\ }%
38     \edef\@citeb{\expandafter\@firstofone\@citeb\@empty}}%
39   \if@filesw\immediate\write\@auxout{\string\citation{\@citeb}}\fi

```

Using `\hbox` instead of `\mbox` is fine because of the `\leavevmode` above. In fact the use of a box around the citation contents is more than questionable in my view (FMi), but within 2e I have to keep that for compatibility reasons as it would probably change too many existing documents. Its main reason is to avoid hyphenation of labels such as [FOOB89] into [FOO- B89] so in certain styles it makes sense; but, for example, in author year citations it becomes more than questionable.

So Chris added yet another hook here, as suggested by, at least, Donald Arseneau. Note that this one is inside the first argument of the `\@cite` hook. This decouples the top-level typesetting of the citation from the details of the other business conducted here. All this really needs a complete rethink to get the right modularity.

```

40   \@ifundefined{b@\@citeb}{\hbox{\reset@font\bfseries ?}}%
41   \G@refundefinedtrue
42   \@latex@warning
43     {Citation ‘\@citeb’ on page \thepage \space undefined}}%
44   {\@cite@ofmt{\csname b@\@citeb\endcsname}}}{#1}}

```

(End of definition for \@citex.)

```

\@bibdata
\@bibstyle
45 \let\@bibdata=\@gobble
46 \let\@bibstyle=\@gobble

```

(End of definition for \@bibdata and \@bibstyle.)

```

\bibliography
47 \def\bibliography#1{%
48   \if@filesw
49     \immediate\write\@auxout{\string\bibdata{\zap@space#1 \@empty}}%
50   \fi
51   \@input@{\jobname.bbl}}

```

(End of definition for `\bibliography`.)

`\bibliographystyle`

```
52 \def\bibliographystyle#1{%
53   \ifx\@begindocumenthook\@undefined\else
54     \expandafter\AtBeginDocument
55   \fi
56   {\if@filesw
57     \immediate\write\@auxout{\string\bibstyle{#1}}%
58   \fi}}
```

(End of definition for `\bibliographystyle`.)

`\nocite` (Added 14 Jun 85)

This puts information on the `.aux` file that causes `BIBTEX` to include the citation list in the bibliography, but puts nothing in the text.

RmS 93/08/06: Made loop for `\nocite` like that for `\@citex`, to get rid of leading spaces.

```
59 \</2kernel)
60 \<*2kernel | latexrelease)
61 \<latexrelease)\IncludeInRelease{2021/06/01}%
62 \<latexrelease)          {\nocite}{Allow nocite in preamble}%
63 \def\nocite#1{\@bsphack
```

With the implementation designed already in `LATEX 2.09` the `\nocite` command will not work before `\begin{document}` since it tries to write to the `.aux` file which is not open before that point. As a result the “reference” will appear on the terminal and nothing else will happen.

[This would be easy to fix, but then a document using the fix will silently fail on an older release of `LATEX`, missing all citations done with `\nocite`. Thus we do only generate an error message and leave the fix for a `LATEX 2ε` successor.]

Given that we are now a quarter century into using `LATEX 2ε` there is no good reason any more do limit ourself to 2.09 considerations. So we now simply delay the `\nocite` if it is issued in the preamble.

```
64   \ifx\@onlypreamble\document
```

Since we are after `\begin{document}` we can do the citations:

```
65     \@for\@citeb:=#1\do{%
66       \edef\@citeb{\expandafter\@firstofone\@citeb}%
67       \if@filesw\immediate\write\@auxout{\string\citation{\@citeb}}\fi
68       \@ifundefined{b@\@citeb}{\G@refundefinedtrue
69         \@latex@warning{Citation ‘\@citeb’ undefined}}{}%
70     \else
```

But before `\begin{document}` we raised an error message in the past but as of 2021/05 not any longer.

```
71 %   \@latex@error{Cannot be used in preamble}\@eha
```

Instead we delay the declaration to the start of the document. We have to use a late hook for this, so that it comes after the `.aux` file is open for writing and after `\@preamblecmds` was executed to change the above test. Therefore `\AtBeginDocument` would still be too early.

```
72     \AddToHook{begindocument/end}[kernel]{\nocite{#1}}%
73   \fi
```

```

74 \@esphack}
75 </2ekernel | latexrelease>
76 <latexrelease>\EndIncludeInRelease
77 <latexrelease>\IncludeInRelease{0000/00/00}%
78 <latexrelease> \nocite}{Allow nocite in preamble}%
79 <latexrelease>
80 <latexrelease>\def\nocite#1{\@bsphack
81 <latexrelease> \ifx\@onlypreamble\document
82 <latexrelease> \@for\@citeb:=#1\do{%
83 <latexrelease> \edef\@citeb{\expandafter\@firstofone\@citeb}%
84 <latexrelease> \if@files\immediate\write\@auxout{\string\citation{\@citeb}}\fi
85 <latexrelease> \@ifundefined{b@\@citeb}{\G@refundefinedtrue
86 <latexrelease> \@latex@warning{Citation ‘\@citeb’ undefined}}{}}%
87 <latexrelease> \else
88 <latexrelease> \@latex@error{Cannot be used in preamble}\@eha
89 <latexrelease> \fi
90 <latexrelease> \@esphack}
91 <latexrelease>
92 <latexrelease>\EndIncludeInRelease
93 <*2ekernel>

```

Since `\nocite{*}` should not produce a warning about undefined citation keys (see PR 557), we need to set the control sequence ‘`b@*`’ to something other than `\relax`. As a result `\cite{*}` will not warn either (but that never worked with `BIBTEX` in the first place).

```
94 \expandafter\let\csname b@*\endcsname\@empty
```

(End of definition for \nocite.)

1.1 Default definitions

This hook determines the ‘relative formatting’ of the two logical parts of a citation with comment.

`\@cite`

```
95 \def\@cite#1#2{[#1\if@tempswa , #2\fi]}
```

(End of definition for \@cite.)

`\@cite@ofmt`

This is, in general, a command that appears to have one argument whose value is, in the kernel, a single cs whose name is the expansion of `b@\@citeb`; the expansion of this cs will typically be some hmode material that produces the detailed typeset form of just the citations themselves.

```
96 \let\@cite@ofmt\hbox
```

(End of definition for \@cite@ofmt.)

`\@biblabel`

```
97 \def\@biblabel#1{[#1]}
98 </2ekernel>
```

(End of definition for \@biblabel.)

File 48

ltmarks.dtx

Abstract

Marks are used to communicate information about the content of a page to the output routine. For example, in order to construct running headers, the output routine needs information about which section names are present on a page, and this information is passed to it through the mark system. However, marks may also be used for other purposes. This module provides a generalized mechanism for marks of independent classes.

1 Introduction

The $\text{T}_{\text{E}}\text{X}$ engines offer a low-level mark mechanism to communicate information about the content of the current page to the asynchronous operating output routine. It works by placing `\mark` commands into the source document. When the material for the current page is assembled in box 255, $\text{T}_{\text{E}}\text{X}$ scans for such marks and sets the commands `\topmark`, `\firstmark` and `\botmark`. The `\firstmark` receives the content of the first `\mark` seen in box 255 and `\botmark` the content of the last mark seen. The `\topmark` holds the content of the last mark seen on the previous page or more exactly the value of `\botmark` from the previous page. If there are no marks on the current page then all three are made equal to the `\botmark` from the previous page.

This mechanism works well for simple formats (such as plain $\text{T}_{\text{E}}\text{X}$) whose output routines are only called to generate pages. It fails, however, in $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ (and other more complex formats), because here the output routine is sometimes called without producing a page, e.g., when encountering a float and placing it into one of the float regions. In that case the output routine is called, determines where to place the float, alters the goal for assembling text material (if the float was added to the top or bottom region) and then it resumes collecting textual material.

As a result the `\botmark` gets updated and so `\topmark` no longer reflects the situation at the top of the next page when that page is finally boxed.

Another problem for $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ was that it wanted to use several “independent” marks and in the early implementations of $\text{T}_{\text{E}}\text{X}$ there was only a single `\mark` command available. For that reason $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ implemented its own mark mechanism where the marks always contained two parts with their own interfaces: `\markboth` and `\markright` to set marks and `\leftmark` and `\rightmark` to retrieve them.

However, this extended mechanism (while supporting scenarios such as chapter/section marks) was far from general. The mark situation at the top of a page (i.e., `\topmark`) remained unusable and the two marks offered were not really independent of each other because `\markboth` (as the name indicates) was always setting both.

The new mechanism overcomes both issues:

- It provides arbitrarily many, fully independent named marks, that can be allocated and, from that point onwards, used.
- It offers access for each such marks to retrieve its top, first, and bottom values separately.
- Furthermore, the mechanism is augmented to give access to marks in different “regions” which may not be just full pages.

2 Design-level and code-level interfaces

The interfaces are mainly meant for package developers, but they are usable (with appropriate care) also in the document preamble, for example, when setting up special running headers with `fancyhdr`, etc. They are therefore available both as CamelCase commands as well as commands for use in the L3 programming layer. Both are described together below.

<code>\NewMarkClass</code>	<code>\NewMarkClass {<class>}</code>
<code>\mark_new_class:n</code>	<code>\mark_new_class:n {<class>}</code>

Declares a new `<class>` of marks to be tracked by L^AT_EX. Each `<class>` must be declared before it is used.

Mark classes can only be declared before `\begin{document}`.

<code>\InsertMark</code>	<code>\InsertMark {<class>} {<text>}</code>
<code>\mark_insert:nn</code>	<code>\mark_insert:nn {<class>} {<text>}</code>

Adds a mark to the current galley for the `<class>`, containing the `<text>`.

It has no effect in places in which you can't place floats, e.g., a mark inside a box or inside a footnote never shows up anywhere.

If used in vertical mode it obeys L^AT_EX's internal `\@nobreak` switch, i.e., it does not introduce a breakpoint if used after a heading. If used in horizontal mode it doesn't handle spacing (like, for example, `\index` or `\label` does, so it should be attached to material that is typeset.

<code>insertmark</code>	<code>\AddToHook {insertmark} {<code>}</code>
-------------------------	---

When marks are inserted, the mark content may need some special treatment, e.g., by default `\label`, `\index`, and `\glossary` do not expand at this time (but only later if and when the mark content is actually used. In order to allow packages to augment or alter this setup there is a public hook `insertmark` that is executed at this point. It runs in a group so local modification to commands are only applied to the `<text>` argument of `\InsertMark` or `\mark_insert:nn`.

<code>\TopMark</code>	<code>*</code>	<code>\TopMark</code>	<code>[(<i>region</i>)]</code>	<code>{(<i>class</i>)}</code>
<code>\FirstMark</code>	<code>*</code>	<code>\FirstMark</code>	<code>[(<i>region</i>)]</code>	<code>{(<i>class</i>)}</code>
<code>\LastMark</code>	<code>*</code>	<code>\LastMark</code>	<code>[(<i>region</i>)]</code>	<code>{(<i>class</i>)}</code>
<code>\mark_use_top:nn</code>	<code>*</code>	<code>\mark_use_top:nn</code>	<code>{(<i>region</i>)}</code>	<code>{(<i>class</i>)}</code>
<code>\mark_use_first:nn</code>	<code>*</code>	<code>\mark_use_first:nn</code>	<code>{(<i>region</i>)}</code>	<code>{(<i>class</i>)}</code>
<code>\mark_use_last:nn</code>	<code>*</code>	<code>\mark_use_last:nn</code>	<code>{(<i>region</i>)}</code>	<code>{(<i>class</i>)}</code>

These functions expand to the appropriate mark `<text>` for the given `<class>` in the specified `<region>`. The default `<region>` in the design-level commands is `page`. Note that with the L3 layer commands there are no optional arguments, i.e., both arguments have to be provided.

T_EXhackers note: The result is returned within the `\unexpanded` primitive (`\exp_not:n`), which means that the `<text>` does not expand further when appearing in an `x`-type or `e`-type argument expansion.

The “first” and “last” marks are those seen first and last in the current region/page, respectively. The “top” mark is the last mark of the `<class>` seen in an earlier region, i.e., the `<text>` what would be “current” at the very top of the region.

Important!

The commands are only meaningful inside the output routine, in other places their result is (while not random) unpredictable due to the way L^AT_EX cuts text material into pages. There is, however, one exception: if you produce multiple columns using the `multicol` package, it is possible to retrieve mark values from the regions `first-column`, `last-column`, `mcol-1`, `mcol-2`,... directly after the environment has ended. This can, for example, be useful if a `multicols` has been used inside a box.

Currently, `<region>` is one of `page`, `previous-page`, `column`, `previous-column`, `first-column`, `last-column`, and `mcol-1` (first column in a `multicols`), `mcol-2` (second column in a `multicols`), up to `mcol-20` (twentieth column in a `multicols`). See section ?? for discussion of how these regions behave and how one can make use of them.

<code>\IfMarksEqualTF</code>	<code>*</code>	<code>\IfMarksEqualTF</code>	<code>[(<i>region</i>)]</code>	<code>{(<i>class</i>)}</code>	<code>{(<i>pos</i>₁)}</code>	<code>{(<i>pos</i>₂)}</code>	<code>{(true)}</code>	<code>{(false)}</code>
<code>\IfMarksEqualT</code>	<code>*</code>	<code>\mark_if_eq:nnnnTF</code>	<code>{(<i>region</i>)}</code>	<code>{(<i>class</i>)}</code>	<code>{(<i>pos</i>₁)}</code>	<code>{(<i>pos</i>₂)}</code>	<code>{(true)}</code>	<code>{(false)}</code>
<code>\IfMarksEqualF</code>	<code>*</code>	<code>\mark_if_eq:nnnnnnTF</code>	<code>{(<i>region</i>₁)}</code>	<code>{(<i>class</i>₁)}</code>	<code>{(<i>pos</i>₁)}</code>			
<code>\mark_if_eq:nnnnTF</code>	<code>*</code>		<code>{(<i>region</i>₂)}</code>	<code>{(<i>class</i>₂)}</code>	<code>{(<i>pos</i>₂)}</code>	<code>{(true)}</code>	<code>{(false)}</code>	
<code>\mark_if_eq:nnnnnnTF</code>	<code>*</code>	These conditionals allow you to compare the content of two marks and act based on the result. The commands work in an expansion context, if necessary.						

It is quite common when programming with marks to need to interrogate conditions such as whether marks have appeared on a previous page, or if there are multiple marks present on the current page, and so on. The tests above allow for the construction of a variety of typical test scenarios, with three examples presented below.

The first two conditionals cover only the common scenarios. Both marks are picked up from the same `<region>` (by default `page`) and they have to be of the same `<class>`.⁴¹ The `<posi>` argument can be either `top`, `first`, or `last`.

Important to note is that the comparison is not with respect to the textual content of the marks but whether or not they originated from the same `\InsertMark` command (or the L3 layer version `\mark_insert:nn`).

If you wish to compare marks across different regions or across different classes, you have to do it using the generic test only available in the L3 programming layer or do it manually, i.e., get the marks and then compare the values yourself.⁴²

⁴¹If an undeclared mark class is used the tests return `true` (not an error).

⁴²If two undeclared mark classes are compared the result is always `true`; if a declared and an undeclared

2.1 Use cases for conditionals

However, the basic version is enough for the following typical use cases:

Test for at most one mark of class `myclass` on current page: If the first and last mark in a region are the same then either there was no mark at all, or there was at most one. To test this on the current page:

```
\NewMarkClass{myclass}
\IfMarksEqualTF{myclass}{first}{last}
  { <zero or one mark> }{ <two or more marks> }
```

Test for no mark of class `myclass` in the previous page: If the top mark is the same as the first mark, there is no mark in the region at all. If we wanted to do this test for the previous page:

```
\IfMarksEqualTF[previous-page]{myclass}{top}{first}
  { <no marks> }{ <at least one mark> }
```

Comparing top and last would give you the same result.

Test for zero, one, or more than one: Combining the two tests from above you can test for zero, one or more than one mark.

```
\IfMarksEqualTF{myclass}{top}{first}
  { <no marks> }
  {\IfMarksEqualTF{myclass}{first}{last}
  { <exactly one mark> }{ <more than one mark> }}
```

If you need one of such tests more often (or if you want a separate command for it for readability), then consider defining:

```
\providecommand\IfNoMarkTF[2][page]{\IfMarksEqualTF[#1]{#2}{first}{last}}
```

2.2 Understanding regions

If a page has just been finished then the region `page` refers to the current page and `previous-page`, as the name indicates, refers to the page before the current page. This means you are able to access mark information for the current page as well as for the page before (as long as you are inside the output routine) without the need to explicitly save that information beforehand. The `page` region is the region that is most often queried, which is why commands like `\FirstMark` use that region by default.

In single column documents the `column` is the same as the `page` region, but in two-column documents (if not produced by `multicols`), `column` refers to the current column that just got finished and `previous-column` to the one previously finished. Code for running headers is (in standard L^AT_EX) evaluated only after both columns have been assembled, which is another way of saying that in that case `previous-column` refers to the left column and `column` to the right column. However, to make these somewhat easier to use, there are also aliased names for these two regions: `first-column` and `last-column`.⁴³

mark class is used it is always *false*.

⁴³The region is called “last-column” not “second-column” in anticipation of extending the mechanism to multiple columns, where first and last would still make sense. There aren’t any `previous-first-column` and `previous-last-column` regions to access the corresponding columns from the previous page.

Note that you can only look backwards at already processed regions, e.g., in a `twoside` document finishing a recto (odd, right-hand) page you can access the data from the facing verso (left-hand) page, but if you are finishing a left-hand page you can't integrate data from the upcoming right-hand page. If such a scenario needs to be realized then it is necessary to save the left-hand page temporarily instead of finalizing it, process material for the right-hand page and once both are ready, attach running headers and footers and shipout out both in one go.⁴⁴

The situation starts getting rather complex if you allow for multiple columns in the way they are supported by the `multicol` package. In this case you might have a variable number of “columns” on a single page to be shipped out. And even if not, then a `multicols` might start or end in the middle of the page; in either case, the regions `column` and `previous-column` become rather meaningless and you should therefore not use them.⁴⁵ Instead, the algorithm offers `mcol-1`, `mcol-2`, `mcol-3`, etc., to represent the columns in the `multicols` on the current page to be shipped out. If there is more than one `multicols` on the current page then in the output routine only the columns of the last one will be accessible.

These provisions cover, out of the box, a number of layouts and use cases, but obviously not all. However, more cases can be supported by storing away mark information during the processing. Here is the full algorithm:

- The `column` region is used by the “current column” that is being built (moving through all columns with `previous-column` trailing behind (to handle top marks properly).
- When the `multicols` starts, the `column` region is cleared, i.e., from that point on it looks as if there have not been any marks so far. This will make sure that the top mark in the first column is always empty.
- If the `multicols` extends beyond the current page, then the material designated for the current page is split into columns. The `column` region is used to represent each column in turn.
 - First we copy the current data from `column` to `previous-column`. Then the mark data from the current column is placed into the `column` region. Then we alias `column` to `mcol-1`.
 - These steps are repeated for all columns of the `multicols` environment.
 - Finally, the first and the last column of that page is also made available as `first-column` and `last-column`, respectively.
- All those marks inside any of the columns are also available in the `page` region. Thus, if you are interested in the top, first, or last mark of a specific class on the whole page you simply need to query for it in the `page` region.
- If the `multicols` continues across several pages then this algorithm above is repeated for each page, except that the `column` region is not cleared again. This means that the top mark of the first column of the next page will be the last mark of the last column from the previous page.

⁴⁴As of now that scenario is not (yet) officially supported but it would be possible to achieve this using the shipout hooks to store the verso page and then on the next shipout use the hook to shipout both with running headers and footers attached.

⁴⁵They return something, because they represent the last two columns of the `multicols` when you are inside the output routine, but that is obviously of little use.

- When the `multicols` finishes the remaining material for the current page is balanced to produce columns of roughly equal height.
- Again `column` and `previous-column` are used while this balancing happens and `mcol-1`, `mcol-2`, etc., are used to represent the column regions and `first-column` and `last-column` are set appropriately.
- Then the balanced set of columns is returned back to the page (since there may be space for further material). In addition, all marks inside that material are reinserted so that they become available in the `page` region.
- As a side effect, it is possible (and useful in certain circumstances) to query for mark classes directly after the `multicols` has ended without the need to be inside the output routine. The regions that can be queried this way are `mcol-1`, `mcol-2`, etc. (up to the number of columns the multicol had) and `first-column` and `last-column`.

2.3 Debugging mark code

<code>\DebugMarksOn</code> <code>\DebugMarksOff</code> <code>\mark_debug_on:</code> <code>\mark_debug_off:</code>	<code>\DebugMarksOn ... \DebugMarksOff</code> <p>Commands to turn the debugging of mark code on or off. The debugging output is rather coarse and not really intended for normal use at this point in time.</p>
--	--

3 Application examples

If you want to figure out if a break was taken at a specific point, e.g., whether a heading appears at the top of the page, you can do something like this:

```

\newcounter{breakcounter}
\NewMarkClass{break}
\newcommand\markedbreak[1]{\stepcounter{breakcounter}%
                           \InsertMark{break}{\arabic{breakcounter}}%
                           \penalty #1\relax
                           \InsertMark{break}{-\arabic{breakcounter}}}

```

To test if the break was taken you can test if `\TopMark{break}` is positive (taken) or negative (not taken) or zero (there was never any marked break so far). The absolute value can be used to keep track of which break it was (with some further coding).

to be extended with additional application examples

4 Legacy L^AT_EX 2_ε interface

Here we describe the interfaces that L^AT_EX 2_ε offered since the early nineties and some minor extensions.

4.1 Legacy design-level and document-level interfaces

```
\markboth \markboth {(left)} {(right)}  
\markright \markright {(right)}
```

L^AT_EX 2_ε uses two marks which aren't fully independent. A “left” mark generated by the first argument of `\markboth` and a “right” mark generated by the second argument of `\markboth` or by the only argument of `\markright`. The command `\markboth` and `\markright` are in turn called from heading commands such as `\chaptermark` or `\sectionmark` and their behavior is controlled by the document class.

For example, in the `article` class with `twoside` in force the `\sectionmark` will issue `\markboth` with an empty second argument and `\subsectionmark` will issue `\markright`. As a result the left mark will contain chapter titles and the right mark subsection titles.

Note, however, that in one-sided documents the standard behavior is that only `\markright` is used, i.e., there will only be right-marks but no left marks!

```
\leftmark * \leftmark  
\rightmark * \rightmark
```

These functions return the appropriate mark value from the current page and work as before, that is `\leftmark` will get the last (!) left mark from the page and `\rightmark` the first (!) right mark.

In other words they work reasonably well if you want to show the section title that is current when you are about to turn the page and also show the first subsection title on the current page (or the last from the previous page if there wasn't one). Other combinations can't be shown using this interface.

The commands are fully expandable, because this is how they have been always defined in L^AT_EX. However, this is of course only true if the content of the mark they return is itself expandable and does not contain any fragile material. Given that this can't be guaranteed for arbitrary content, a programmer using them in this way should use `\protected@edef` and *not* `\edef` to avoid bad surprises as far as this is possible, or use the new interfaces (`\TopMark`, `\FirstMark`, and `\LastMark`) which return the `(text)` in `\exp_not:n` to prevent uncontrolled expansion.

4.2 Legacy interface extensions

The new implementation adds three mark classes: `2e-left`, `2e-right` and `2e-right-nonempty` and patches `\markboth` and `\markright` slightly so that they also update these new mark classes, so that the new classes work with existing document classes.

As a result you can use `\LastMark{2e-left}` and `\FirstMark{2e-right}` instead of `\leftmark` and `\rightmark`. But more importantly, you can use any of the other retrieval commands to get a different status value from those marks, e.g., `\LastMark{2e-right}` would return the last subsection on the page (instead of the first as returned by `\rightmark`).

The difference between `2e-right` and `2e-right-nonempty` is that the latter will only be updated if the material for the mark is not empty. Thus `\markboth{title}{}` as issued by, say, `\sectionmark`, sets a `2e-left` mark with `title` and a `2e-right` mark with the empty string but does not add a `2e-right-nonempty` mark.

Thus, if you have a section at the start of a page and you would ask for `\FirstMark{2e-right}` you would get an empty string even if there are subsections on that page. But `2e-right-nonempty` would then give you the first or last subsection

on that page. Of course, nothing is simple. If there are no subsections it would tell you the last subsection from an earlier page. We therefore need comparison tools, e.g., if top and first are identical you know that the value is bogus, i.e., a suitable implementation would be

```
\IfMarksEqualTF{2e-right-nonempty}{top}{first}
  { <appropriate action if there was no real mark> }
  {\FirstMark{2e-right-nonempty}}
```

5 Notes on the mechanism

In contrast to vanilla T_EX, ϵ -T_EX extends the mark system to allow multiple independent marks. However, it does not solve the `\topmark` problem which means that L^AT_EX still needs to manage marks almost independently of T_EX. The reason for this is that the more complex output routine used by L^AT_EX to handle floats (and related structures) means that `\topmark(s)` remain unreliable. Each time the output routine is fired up, T_EX moves `\botmark` to `\topmark`, and while ϵ -T_EX extends this to multiple registers the fundamental concept remains the same. That means that the state of marks needs to be tracked by L^AT_EX itself. An early implementation of this package used T_EX's `\botmark` only to ensure the correct interaction with the output routine (this was before the ϵ -T_EX mechanism was even available). However, other than in a prototype implementation for L^AT_EX3, this package was never made public.

The new implementation now uses ϵ -T_EX's marks as they have some advantages, because with them we can leave the mark text within the galley and only extract the marks during the output routine when we are finally shipping out a page or storing away a column for use in the next page. That means we do not have to maintain a global data structure that we have to keep in sync with informational marks in the galley but can rely on everything being in one place and thus manipulations (e.g. reordering of material) will take the marks with them without a need for updating a fragile linkage.

To allow for completely independent marks we use the following procedure:

- For every type of marks we allocate a mark class so that in the output routine T_EX can calculate for each class the current top, first, and bottom mark independently. For this we use `\newmarks`, i.e., one marks register per class.
- As already mentioned firing up an output routine without shipping out a page means that T_EX's top marks get wrong so it is impossible to rely on T_EX's approach directly. What we do instead is to keep track of the real marks (for the last page or more generally last region) in some global variables.
- These variables are updated in the output routine at defined places, i.e., when we do real output processing but not if we use special output routines to do internal housekeeping.
- The trick we use to get correctly updated variables is the following: the material that contains new marks (for example the page to be shipped out) is stored in a box. We then use T_EX primitive box splitting functions by splitting off the largest amount possible (which should be the whole box if nothing goes really wrong). While that seems a rather pointless thing to do, it has one important side effect: T_EX sets up first and bottom marks for each mark class from the material it has split off. This way we get the first and last marks (if there have been any) from the material in the box.

- The top marks are simply the last marks from the previous page or region. And if there hasn't been a first or bottom mark in the box then the new top mark also becomes new first and last mark for that class.
- That mark data is then stored in global token lists for use during the output routine and legacy commands such as `\leftmark` or new commands such as `\TopMark` simply access the data stored in these token lists.

That's about it in a nutshell. Of course, there are some details to be taken care of—those are discussed in the implementation sections.

6 Public interfaces for packages such as `multicol`

The functions in this section are public so that packages can make use of them. However, this must be done with great care, e.g., `\mark_update_structure_from_material:nn` updates the global mark structure and can therefore be used only in places where such an update is meaningful, e.g., in special output routines. Elsewhere, a change to the mark structure would break the whole mechanism and querying the marks would return incorrect data.

`\mark_update_structure_from_material:nn` `\mark_update_structure_from_material:nn` $\langle\mathit{region}\rangle$ $\langle\mathit{material\ with\ marks}\rangle$

Helper function that inspects the marks inside the second argument and assigns new mark values based on that to the $\langle\mathit{region}\rangle$ given in the first argument. For this it first copies the mark structure from $\langle\mathit{region}\rangle$ to `previous- $\langle\mathit{region}\rangle$` and then takes all last mark values currently in the region and makes them the new top mark values. Finally it assigns new first and last values for all mark classes based on what was found in the second argument.

As a consequence, the allowed values for $\langle\mathit{region}\rangle$ are `page` and `column` because only they have `previous-...` counterparts.

Another important aspect to keep in mind is that marks are recognized only if they appear on the top level, e.g., if we want to process material stored in boxes we need to put it unboxed (using `\unvcopy` etc.) into the second argument.

`\mark_copy_structure:nn` `\mark_copy_structure:nn` $\langle\mathit{alias}\rangle$ $\langle\mathit{source}\rangle$

Helper function that copies all mark values in the $\langle\mathit{source}\rangle$ region to $\langle\mathit{alias}\rangle$, i.e., make the structures identical. Used to update the `previous-...` structures inside `\mark_update_structure_from_material:nn` and `first-column` and `last-column` structures inside the internal commands `__mark_update_singlecol_structures:` or `__mark__update_dbcol_structures:`.

`\mark_set_structure_to_err:n` `\mark_set_structure_to_err:n` $\langle\mathit{region}\rangle$

Helper function that sets all mark values in the $\langle\mathit{region}\rangle$ to an error message. This is currently used for `last-column` at times where using marks from it would be questionable/wrong, i.e., when we have just processed the first column in a two-column document.

`\mark_clear_structure:n` `\mark_clear_structure:n <{region}>`

Helper function that sets all mark values in the `<region>` to empty. This is currently used for `column` when a multicol environment starts; this is because it wouldn't make sense if the top mark in the first column returned the last mark from a previous multicol (which may have been much earlier, with intermediate material).

`\mark_get_marks_for_reinsertion:nNN` `\mark_get_marks_for_reinsertion:nNN <{source}>`
<token-list-var for collecting first marks>
<token-list-var for collecting last marks>

Helper function for extracting marks that would otherwise get lost, for example when they are hidden inside a box. This helper does not update mark structures and can therefore be used outside the output routine as well.

It collects all the top-level marks from inside the `<source>` and then adds suitable `\mark_insert:nn` commands to each of the two token lists. These token lists can then be executed at the right place to reinsert the marks, e.g., directly after the box. This is, for example, going to be used⁴⁶ by `multicol` when a short balanced `multicols` is returned to the galley for typesetting.

If the `<source>` consists of a single vertical box (plus possibly followed by some glue but nothing else) then the box is unpacked and the top-level marks are collected from its content. However, if it is not a vertical box or there are other data then nothing is unpacked and you have to do the unpacking yourself to get at the marks inside.

It is quite likely that one only needs a single token list for returning the `\mark_insert:nn` statements. If that is the case this command may change to take only two arguments.

7 Internal functions for the standard output routine of L^AT_EX

The functions in this section are tied to the output routine and used in the interface to L^AT_EX 2_ε and perhaps at some later time within a new output routine for L^AT_EX. They are not (yet) meant for general use and are therefore made internal, even though we already use them in `multicol`. Internal means that `@@` automatically gets replaced in the code (and in the documentation) so we have to give it a suitable value.

1 `<@@=mark>`

`_mark_update_singlecol_structures:` `_mark_update_singlecol_structures:`

L^AT_EX 2_ε integration function in case we are doing single column layouts. It assumes that the page content is already stored in `\@outputbox` and processes the marks inside that box. It is called as part of `\@opcol`.

`_mark_update dblcol_structures:` `_mark_update_singlecol_structures:`

L^AT_EX 2_ε integration function mark used when we are doing double column documents. It assumes that the page content is already stored in `\@outputbox` and processes the marks inside that box. It then does different post-processing depending on the start of the switch `\if@firstcolumn`. If we are in the second column it also has to update page marks, otherwise it only updates column marks. It too is called as part of `\@opcol`.

⁴⁶Probably not before 2025, though.

8 The Implementation

```
2 <*2ekernel | latexrelease>
3 \ExplSyntaxOn
4 <latexrelease>\NewModuleRelease{2022/06/01}{ltmarks}
5 <latexrelease>           {Marks-handling}
```

8.1 Allocating new mark classes

`\g__mark_classes_seq` A list holding all the mark classes that have been declared.

```
6 \seq_new:N \g__mark_classes_seq
```

`\mark_new_class:n` A mark class is created by initializing a number of data structures. First, we get a register number to refer to the mark class. The new mark class is then added to the `\g__mark_classes_seq` sequence to be able to easily loop over all classes. Finally a number of top-level global token lists are declared that hold various versions of the mark for access.

`__mark_new_class:nn`

```
7 \cs_new_protected:Npn \mark_new_class:n #1
8 {
9   \seq_if_in:NnTF \g__mark_classes_seq {#1}
10     {
11       \msg_error:nnn { mark } { class-already-defined }
12         {#1}
13     }
14     { \__mark_new_class:nn {#1} }
15 }
```

This is only available in the preamble.

```
16 \@onlypreamble \mark_new_class:n
```

The internal command carries out the necessary allocations.

```
17 \cs_new_protected:Npn \__mark_new_class:nn #1
18 {
19   <*trace>
20   \__mark_debug:n { \iow_term:x { Marks:~new-mark:~#1~\msg_line_context: } }
21   </trace>
```

Use the L^AT_EX 2_ε interface for now as the L3 programming layer doesn't have one for marks yet.

```
22 \exp_args:Nc \newmarks {c__mark_class_ #1 _mark}
```

Remember the new class in the sequence.

```
23 \seq_gput_right:Nn \g__mark_classes_seq {#1}
24 \__mark_init_region:nn {page}{#1}
```

For the page region we also keep track of the previous-page.

```
25 \__mark_init_region:nn {previous-page}{#1}
```

Same game for column and previous-column

```
26 \__mark_init_region:nn {column}{#1}
27 \__mark_init_region:nn {previous-column}{#1}
```

But for columns we also allocate token lists for the alias regions `first-column` and `last-column`.

```
28 \__mark_init_region:nn {first-column}{#1}
29 \__mark_init_region:nn {last-column}{#1}
```

To support multiple columns produced by the `multicol` package, we preallocate twenty alias regions (since this is the number of columns that `multicol` supports as a maximum). They are filled by copying the current column into the appropriate `mcol-...`

```
30 %fmi \__mark_init_region:nn {mcol}{#1}
31 %fmi \__mark_init_region:nn {previous-mcol}{#1}
32 \__mark_init_region:nn {mcol-1}{#1}
33 \__mark_init_region:nn {mcol-2}{#1}
34 \__mark_init_region:nn {mcol-3}{#1}
35 \__mark_init_region:nn {mcol-4}{#1}
36 \__mark_init_region:nn {mcol-5}{#1}
37 \__mark_init_region:nn {mcol-6}{#1}
38 \__mark_init_region:nn {mcol-7}{#1}
39 \__mark_init_region:nn {mcol-8}{#1}
40 \__mark_init_region:nn {mcol-9}{#1}
41 \__mark_init_region:nn {mcol-10}{#1}
42 \__mark_init_region:nn {mcol-11}{#1}
43 \__mark_init_region:nn {mcol-12}{#1}
44 \__mark_init_region:nn {mcol-13}{#1}
45 \__mark_init_region:nn {mcol-14}{#1}
46 \__mark_init_region:nn {mcol-15}{#1}
47 \__mark_init_region:nn {mcol-16}{#1}
48 \__mark_init_region:nn {mcol-17}{#1}
49 \__mark_init_region:nn {mcol-18}{#1}
50 \__mark_init_region:nn {mcol-19}{#1}
51 \__mark_init_region:nn {mcol-20}{#1}
52 }
```

(End of definition for `\mark_new_class:n` and `__mark_new_class:nn`. This function is documented on page 1004.)

`__mark_init_region:nn` For each class (#2) and region (#1), we need three token lists: one for top, first, and last.
`\c__mark_empty_tl` The default value to be returned is “empty”.

```
53 \cs_new_protected:Npn \__mark_init_region:nn #1 #2 {
54 \tl_new:c { g__mark_#1_top_ #2 _tl }
55 \tl_new:c { g__mark_#1_first_ #2 _tl }
56 \tl_new:c { g__mark_#1_last_ #2 _tl }
57 \tl_gset_eq:cN { g__mark_#1_top_ #2 _tl } \c__mark_empty_tl
58 \tl_gset_eq:cN { g__mark_#1_first_ #2 _tl } \c__mark_empty_tl
59 \tl_gset_eq:cN { g__mark_#1_last_ #2 _tl } \c__mark_empty_tl
60 }
```

All marks will have an identification in the form of a number⁴⁷ that is incremented each time a mark insertion happens; therefore the initial empty values should also have such a number, so that data extraction will be uniform.

```
61 \tl_const:Nn \c__mark_empty_tl { \__mark_value:nn{0}{-} }
```

(End of definition for `__mark_init_region:nn` and `\c__mark_empty_tl`.)

⁴⁷There are a few cases where special identification strings are used, e.g., `2.09-compat`.

8.2 Updating mark structures

```

\l__mark_box
\l__mark_ii_box
\g__mark_tmp_tl
\g__mark_new_top_tl

```

For some operations we need two temporary private boxes and two private global token lists.

```

62 \box_new:N \l__mark_box
63 \box_new:N \l__mark_ii_box
64 \tl_new:N \g__mark_tmp_tl
65 \tl_new:N \g__mark_new_top_tl

```

(End of definition for `\l__mark_box` and others.)

```

\_mark_extract_and_handle_marks:nn

```

This is the main macro to extract and handle marks inside some vertical material. It is used by `\mark_update_structure_from_material:nn` (for updating the mark structure for a region based on the marks found) and by `\mark_get_marks_for_reinsertion:nNN` (for extracting marks from some material and prepare for reinserting them later (e.g., out of a box that is placed as a box into the main galley)).

```

66 \cs_new_protected:Npn \_mark_extract_and_handle_marks:nn #1#2 {

```

This macro expects code to handle extracted marks in its first argument and vertical material (not boxed or just consisting of a single vertical box) as its second. It extracts top-level mark information from #2, stores them as split marks and then calls #1 to make use of this information.

If it finds a forced break in the material it removes it and then restarts the attempt without it.

We start with a group to keep most changes local.

```

67 \group_begin:

```

Getting the first and last marks out of the material in #2 is done by putting the material in a box and then doing a split operation to the maximum size possible (which hopefully gets us all of the content).⁴⁸ Because this action is used only to get the mark values, we don't want any underfull box warnings so we (locally) turn those off.

```

68 \dim_set_eq:NN \tex_splitmaxdepth:D \c_max_dim
69 \int_set_eq:NN \tex_vbadness:D \c_max_int
70 \dim_set_eq:NN \tex_vfuzz:D \c_max_dim

```

There is a further complication: if the material contains infinite shrinking glue then a `\vsplit` operation will balk with a low-level error. Now pages or columns, which are our main concern here, can't have such infinite shrinkage if they are cut straight from the galley, however the use of `\enlargethispage` actually does add some at the very bottom (and also wraps the whole page into a box by itself, so if we leave it this way then a) we get this error and b) we don't see any marks because they are hidden one level down).

Another possible issue are packages or user code that place stray `\vboxes` directly into the main galley (an example is `marginnote` that attaches its marginals in this way). If such boxes end up as the last item on the page we should not unpack them.

All these issues need to be handled, which is done in `_mark_prepare_and_extract:nn`.

```

71 \_mark_prepare_and_extract:nn {#1} {#2}

```

⁴⁸With normal column material cut from the main galley we should always get all material in one go, but in certain situations, for example, in a `multicols` environment that contains some `\columnbreaks` a single split operation will not be enough. Thus, this is something we need to handle.

Once all mark classes have been processed, the data structures are updated and we can close the group, which undoes our local changes and retains only the global ones.

```
72     \group_end:
73   }
```

(End of definition for `_mark_extract_and_handle_marks:nn`.)

`_mark_prepare_and_extract:nn` This macro does the dirty work. It is not directly integrated in `_mark_extract_and_handle_marks:nn` because we may have to call it recursively if we find forced breaks.

```
74 \cs_new_protected:Npn \_mark_prepare_and_extract:nn #1#2 {
```

To handle the `\enlargethispage` case we do an `\unskip` to get rid of any glue that is present at the very end of the material and also check if we have then a `\vbox` as the last item and if so unpack that too, but only under certain conditions, see below. All this is temporary done in a group, just for getting the marks out, so it doesn't affect the final page production.

```
75   \vbox_set:Nn \l__mark_box
76   {
77     #2
78     \tex_unskip:D
79     \box_set_to_last:N \l__mark_box
```

After having removed the last box from the current list (if there was one) we check whether the vertical list is now empty. If not, then the last box is definitely not the one from `\enlargethispage` and so we can, and should, leave it alone. Otherwise we check if this last box is a `\vbox`.

```
80     \int_compare:nNnT \tex_lastnodetype:D < 0
81     {
82       \box_if_vertical:NT \l__mark_box
```

If it is, we unpack the box.

```
83         { \vbox_unpack:N \l__mark_box }
84     }
```

If it wasn't a `\vbox`, it was either an `\hbox` or there was no box. Given that we are only interested in the marks we don't need put it back in that case.

```
85   }
```

We are now ready to `\vsplit` the box to get at the marks. If the box contains some infinite negative glue the `TEX` will produce an error complaining about it but it will correctly find the split marks. Given that we can't prevent that error, we hide it from the user and ensure that `TEX` doesn't stop. The error message still shows in the log, but even that is mitigated as best as possible—see the definition of `_mark_vbox_set_split_to_maxdimen:NN` for the tricks employed.

```
86   \_mark_vbox_set_split_to_maxdimen:NN \l__mark_ii_box \l__mark_box
```

After splitting we check if there is anything left in `\l__mark_box`. If not then the above split has set some split marks that we can then use to finish the extraction:

```
87   \box_if_empty:NTF \l__mark_box
88     { #1 }
```

If we have a remainder after the split then this means that there was some forced break in the material. We get rid of that by combining the content of the two boxes and restart.

```

89     {
90     <*trace>
91     \__mark_debug:n { \iow_term:x
92     { Marks:~ mark~ extraction~needs~ recursion~
93     \msg_line_context: } }
94     </trace>
95     \__mark_prepare_and_extract:nn {#1}
96     { \vbox_unpack:N \l__mark_ii_box
97     \vbox_unpack:N \l__mark_box }
98     }
99 }

```

(End of definition for __mark_prepare_and_extract:nn.)

__mark_vbox_set_split_to_maxdimen:NN

Split a box to get at its marks without pausing even if T_EX is producing an error message because of infinite negative glue in the box. If there is such an error we ensure that it only shows up in the log but not on the terminal.

The nice low-level hack by DPC records in the .log that a glue shrinkage error is harmless.

We disguise \c_max_dim in an odd looking csname, which then shows up as part of the display of an error message if that error happens. This csname forms part of the error display so what you get is something like

```

! Infinite glue shrinkage found in box being split.
<argument> Infinite shrink error above ignored !
1. ... }

```

which hopefully makes it clear that the error is harmless and should be ignored by the reader of the .log.

```

100 \cs_set_eq:cN {Infinite~shrink~error~above~ignored~!}\c_max_dim

```

The whole definition of __mark_vbox_set_split_to_maxdimen:NN below is fully expanded, so we have to use a lot of \exp_not:N commands to prevent expansion where necessary.

```

101 \cs_new_protected:Npx \__mark_vbox_set_split_to_maxdimen:NN #1#2 {

```

We start by saving the current interaction and escape char settings.

```

102 \tl_set:Nc \exp_not:N \l__mark_saved_parameters_tl
103 {
104 \tex_interactionmode:D
105 \exp_not:N \int_use:N \tex_interactionmode:D \scan_stop:
106 \tex_escapechar:D
107 \exp_not:N \int_use:N \tex_escapechar:D \scan_stop:
108 }

```

Then we change them so that no escape char is printed in the error message (accounts for the missing backslash in front of `Infinite shrink ...`) and we set the interaction to \nonstopmode so that the error (if any) just goes into the .log file and T_EX doesn't stop at that point.

```

109 \tex_escapechar:D -1 \scan_stop:
110 \tex_interactionmode:D 0 \scan_stop:

```

Then we do the splitting of the box to `\c_max_dim` to get at the marks. This may generate the error we are worried about, i.e., if the box contains infinite negative glue. However, \TeX makes this glue finite and continues, which means we get our split marks which is really all we care about.

```
111 \tex_setbox:D #1 \tex_vsplit:D #2 to
```

The `\use:n` may seem pointless, and it is to some extent, but we need it to get our disguised `\c_max_dim` displayed properly as part of the error message if there is one. Without it, the display would show only part of what we want it to show (try it).

```
112     \exp_not:N \use:n {
113         \use:c{Infinite-shrink-error-above-ignored-!}
114     }
```

Finally, we change the escape char and the interaction mode back to what it was before:

```
115 \exp_not:N \l__mark_saved_parameters_tl
116 }
```

(End of definition for `__mark_vbox_set_split_to_maxdimen:NN`.)

`\l__mark_saved_parameters_tl` The temporary variable used for resetting escape char and interaction mode.

```
117 \tl_new:N \l__mark_saved_parameters_tl
```

(End of definition for `\l__mark_saved_parameters_tl`.)

`\mark_update_structure_from_material:nn` This function updates the mark structures of a region. The first argument is the region to update and second argument receives the material that holds the marks. Out of this material we extract the first and last marks for all classes (if there are any) to do the assignments.

```
118 \cs_new_protected:Npn \mark_update_structure_from_material:nn #1#2 {
119     \__mark_extract_and_handle_marks:nn
```

Once the marks can be extracted we update the structure from the split marks (code in `__mark_update_structure_from_splitmarks:n`).

```
120     { \__mark_update_structure_from_splitmarks:n {#1} }
121     { #2 }
122 }
```

(End of definition for `\mark_update_structure_from_material:nn`. This function is documented on page 1011.)

`__mark_update_structure_from_splitmarks:n` This macro is called after we have done a `\tex_vsplit:D` operation and the mark data is in the split marks.

```
123 \cs_new_protected:Npn \__mark_update_structure_from_splitmarks:n #1 {
```

The first thing we do is to copy the current region structure to `previous-...`; this leaves the current structure untouched so we can update it class by class (as is necessary).

```
124     \mark_copy_structure:nn { previous-#1 } {#1}
```

After this action we can get first and last marks of the various classes through `\tex_splitfirstmarks:D` and `\tex_splitbotmarks:D`. So now we loop over all classes stored in `\g__mark_classes_seq`.

```
125     \seq_map_inline:Nn \g__mark_classes_seq
126     {
```

First action: get the last mark from the previous region, i.e., `previous-#1`. But because it is also still inside `#1`, at the moment we use that to construct the name because this is a tiny bit faster. Given that we need this value in various assignments we store it away which avoids unnecessary further `cname` generations.

```
127         \tl_gset_eq:Nc \g__mark_new_top_tl { g__mark_#1_last_##1_tl }
```

This will first of all become the new top mark for the current class.

```
128         \tl_gset_eq:cN { g__mark_#1_top_##1_tl } \g__mark_new_top_tl
```

Next action is to get ourselves the new last mark from the material supplied.

```
129         \tl_gset:No \g__mark_tmp_tl
130         { \tex_splitbotmarks:D \use:c { c__mark_class_##1_mark } }
```

If this mark doesn't exist then obviously neither does the first mark, so both become the last mark from the previous region. We have to be a little careful here: something like `\mark_insert:nn{foo}{}` adds an "empty" mark that should not be confused with no mark at all. But no mark in our material will result in `\g__mark_tmp_tl` being fully empty. This is why we have to make sure that "empty" from `\mark_insert:nn` only appears to be empty when typeset but fails the next test (see below how this is done).

```
131         \tl_if_empty:NTF \g__mark_tmp_tl
132         {
133             \tl_gset_eq:cN { g__mark_#1_last_ ##1_tl }
134             \g__mark_new_top_tl
135             \tl_gset_eq:cN { g__mark_#1_first_##1_tl }
136             \g__mark_new_top_tl
137         }
```

If it wasn't empty, i.e., if it had a real value then we use this value for our new last mark instead.

```
138         {
139             \tl_gset_eq:cN { g__mark_#1_last_##1_tl } \g__mark_tmp_tl
```

Because we had a last mark we also have a first mark (which might be the same, but might be not), so we pick that up and assign it to the appropriate token list. This explains why we first checked for the last mark because that makes the processing faster in case there is none.

```
140         \tl_gset:co { g__mark_#1_first_##1_tl }
141         {
142             \tex_splitfirstmarks:D
143             \use:c { c__mark_class_##1_mark }
144         }
145     }
146 }
147 }
```

(End of definition for `__mark_update_structure_from_splitmarks:n`.)

`\mark_get_marks_for_reinsertion:nNN`

This function extracts the marks from the material in the first argument but it does not update any the mark structures. Instead, it collects the marks in the token lists given as the second and third argument, in such a way that they can be reinserted by just executing the token lists.⁴⁹

```
148 \cs_new_protected:Npn \mark_get_marks_for_reinsertion:nNN #1#2#3 {
```

⁴⁹It is probably enough to collect everything in a single token list as long as we put the first marks first and the last marks last). But for extra flexibility, I currently use 2 token lists. This might change when it is really clear that this is never needed.

First we clear the temporary token lists as we haven't seen any marks yet.

```
149 \tl_gclear:N \g__mark_first_marks_tl
150 \tl_gclear:N \g__mark_last_marks_tl
```

Then we extract all top-level marks, thereby filling the token lists with suitable `\mark_insert:nn` calls.

```
151 \__mark_extract_and_handle_marks:nn
```

The first argument holds the code used for filling the token lists and the second holds the material from which all marks should be extracted.

```
152 \__mark_get_from_splitmarks:
153 { #1 }
```

Finally, we copy the updated (or not updated) temporary token lists to the two that have been supplied when the function was called. By convention “get” operations return their values in local variables and `__mark_extract_and_handle_marks:nn` runs in a group, which is why we have to use global temporary variables for collecting.

```
154 \tl_set_eq:NN #2 \g__mark_first_marks_tl
155 \tl_set_eq:NN #3 \g__mark_last_marks_tl
156 }
```

(End of definition for `\mark_get_marks_for_reinsertion:nnn`. This function is documented on page 1012.)

`__mark_get_from_splitmarks:` This function is called after we have done a `\vsplit` to update the split marks. It loops through all mark classes to find out if there are marks for this class and if so updates the global tls used for collecting.

```
157 \cs_new_protected:Npn \__mark_get_from_splitmarks: {
158 \seq_map_inline:Nn \g__mark_classes_seq
159 {
```

First we to get the last mark for the current class from the material supplied.

```
160 \tl_gset:No \g__mark_tmp_tl
161 { \tex_splitbotmarks:D \use:c { c__mark_class_##1_mark } }
```

If this mark doesn't exist then obviously first mark doesn't either, so we do nothing (other than issuing some debugging info).

We have to be a little careful here: something like `\mark_insert:nn{foo}{}` adds an “empty” mark that we should not confuse with the case where there is no mark at all.

When there is no mark at all we get a truly empty `\g__mark_tmp_tl` as a result. This is why we have to make sure that an “empty” mark generated with `\mark_insert:nn` only appears to be empty when it is typeset, but fails the next test (see below how this is done).

```
162 \tl_if_empty:NTF \g__mark_tmp_tl
163 {
164 (*trace)
165 \__mark_debug:n { \iow_term:x { Marks:~no~ marks~
166 for~ class~ '##1'~\msg_line_context: } }
167 
```

If it wasn't empty, i.e., if it had a real value then we use this value for our new last mark instead. This means we put an appropriate `\mark_insert:nn` statement into `\g__mark_last_marks_tl`.

```

169             {
170 (*trace)           \__mark_debug:n { \iow_term:x { Marks:~ extract~ last~
171                   \tl_gput_right:Ne \g__mark_last_marks_tl
The mark content in \g__mark_tmp_tl may contain arbitrary code that may react badly
if it is expanded in a write. So we better avoid that expansion, otherwise debugging
might generate spurious errors when turned on.
172                   mark~ for~ class~ '##1'~ =~ \exp_not:o \g__mark_tmp_tl } }
173 
```

Because we had a last mark we also have a first mark (which might be the same, but might not be), so we pick that up and add it to the `\g__mark_first_marks_tl` token list. This explains why we first checked for the last mark because that makes the processing faster in case there is none.

```

176 (*trace)
177             \__mark_debug:n { \iow_term:x {
178                   Marks:~ extract~ first~ mark~ for~ class~ '##1'~ =~

```

Again no expansion for the mark content.

```

179                   \exp_not:o {
180                     \tex_splitfirstmarks:D
181                     \use:c { c__mark_class_##1_mark }
182                   }
183             } }
184 
```

We better drop the id from the returned value otherwise they will accumulate in the marks when reinserted.

```

188                   \__mark_drop_id:o {
189                     \tex_splitfirstmarks:D
190                     \use:c { c__mark_class_##1_mark }
191                   }
192             }
193           }
194         }
195       }
196 }

```

(End of definition for `__mark_get_from_splitmarks:.`)

`\g__mark_first_marks_tl` These are two global temporary variables used in the code above.

```

\g__mark_last_marks_tl
197 \tl_new:N \g__mark_first_marks_tl
198 \tl_new:N \g__mark_last_marks_tl

```

(End of definition for `\g__mark_first_marks_tl` and `\g__mark_last_marks_tl`.)

`\mark_copy_structure:nn` This function copies the structure for one region to another, e.g., from page to previous-page above, or later from column to first-column, etc.

```
199 \cs_new_protected:Npn \mark_copy_structure:nn #1#2 {
```

This requires a simple loop through all mark classes copying the token list from one name to the next.

```
200   \seq_map_inline:Nn \g__mark_classes_seq
201   {
202     \tl_gset_eq:cc { g__mark_ #1 _top_   ##1 _tl }
203                   { g__mark_ #2 _top_   ##1 _tl }
204     \tl_gset_eq:cc { g__mark_ #1 _first_ ##1 _tl }
205                   { g__mark_ #2 _first_ ##1 _tl }
206     \tl_gset_eq:cc { g__mark_ #1 _last_  ##1 _tl }
207                   { g__mark_ #2 _last_  ##1 _tl }
208   }
209 }
```

(End of definition for \mark_copy_structure:nn. This function is documented on page 1011.)

`\mark_clear_structure:n` This function sets the structure of one region back to an initial state, so that all classes return an empty value if queried.

```
210 \cs_new_protected:Npn \mark_clear_structure:n #1 {
```

This requires a simple loop through all mark classes.

```
211   \seq_map_inline:Nn \g__mark_classes_seq
212   {
213     \tl_gset_eq:cN { g__mark_ #1 _top_   ##1 _tl }
214                   \c__mark_empty_tl
215     \tl_gset_eq:cN { g__mark_ #1 _first_ ##1 _tl }
216                   \c__mark_empty_tl
217     \tl_gset_eq:cN { g__mark_ #1 _last_  ##1 _tl }
218                   \c__mark_empty_tl
219   }
220 }
```

(End of definition for \mark_clear_structure:n. This function is documented on page 1012.)

`\mark_set_structure_to_err:n` A slight variation is to install a fixed error message as the value.

```
  \__mark_error:n
```

```
221 \cs_new_protected:Npn \mark_set_structure_to_err:n #1 {
222   \seq_map_inline:Nn \g__mark_classes_seq
223   {
224     \tl_gset:ce { g__mark_ #1 _top_   ##1 _tl } { \__mark_value:nn{?}{\__mark_error:nn {#1}}
225     \tl_gset:ce { g__mark_ #1 _first_ ##1 _tl } { \__mark_value:nn{?}{\__mark_error:nn {#1}}
226     \tl_gset:ce { g__mark_ #1 _last_  ##1 _tl } { \__mark_value:nn{?}{\__mark_error:nn {#1}}
227   }
228 }
```

Given that this is used in only one place, we could hardwire the argument which would be a bit more compact, but who knows, perhaps we end up with another reason to use this error command elsewhere, so for now we keep the argument.

```
229 \cs_new_protected:Npn \__mark_error:nn #1#2 {
230   \msg_error:nnnn { mark } { invalid-use } {#1} {#2}
231 }
```

(End of definition for \mark_set_structure_to_err:n and __mark_error:n. This function is documented on page 1011.)

8.3 Placing and retrieving marks

`\mark_insert:nn` This function puts a mark for some $\langle class \rangle$ at the current point.

```
232 \cs_new_protected:Npn \mark_insert:nn #1#2
233 {
234   \seq_if_in:NnTF \g__mark_classes_seq {#1}
235     {
```

We need to pass the evaluated argument into the mark but protected commands should not expand including those protected using the `\protect` approach of L^AT_EX 2_ε. We also disable `\label` and the like.⁵⁰

At this point the code eventually should get a public (and a kernel) hook instead of a set of hardwired settings.

```
236     \group_begin:
```

Within the group we alter some comments, e.g. `\label` or `\index`, to do the right at this point. This is done in the kernel hook `\@kernel@before@insertmark` which is followed by the public hook `insertmark` that can be used by packages to augment or alter that setup as necessary.

```
237         \@kernel@before@insertmark
238         \hook_use:n { insertmark }
239         \unrestored@protected@xdef \g__mark_tmp_tl
240         {
```

To ensure that marks are unique we insert a hidden sequence marker at the beginning of the content of the mark containing the sequence number of the mark.

```
241             \__mark_value:nn{ \int_use:N\g__mark_int }{#2}
242             }
243 < *trace >
244         \__mark_debug:n{ \iow_term:x { Marks:~ set~#1~<--
245             '\tl_to_str:V \g__mark_tmp_tl' ~ \msg_line_context: } }
246 < /trace >
247         \tex_marks:D \use:c { c__mark_class_ #1 _mark }
248         {
```

Here is the trick to avoid truly empty marks: if the result from the above processing is empty we add something which eventually becomes empty, but not immediately; otherwise we just put `\g__mark_tmp_tl` in.

```
249 % This is no longer needed with 1.0f
250 %         \tl_if_empty:NTF \g__mark_tmp_tl
251 %         { \exp_not:n { \prg_do_nothing: } }
252 %         { \exp_not:o { \g__mark_tmp_tl } }
253         \exp_not:o { \g__mark_tmp_tl }
254     }
255     \group_end:
```

A mark introduces a possible break point and in certain situations that should not happen in vertical mode in L^AT_EX. This may need some checking and possibly cleanup . . .

```
256         \if@nobreak\ifvmode\nobreak\fi\fi
257     }
```

⁵⁰Straight copy from `latex.ltx` but is this even correct? At least a label in a running header makes little sense if it get set several times! Maybe that needs looking at in the 2e kernel.

If the mark class was not known, raise an error.

```

258     {
259         \msg_error:nxx { mark } { unknown-class }
260         { \tl_to_str:n {#1} }
261     }
262 }

```

(End of definition for `\mark_insert:nn`. This function is documented on page 1004.)

`__mark_value:nn` A hidden marker is placed into every mark added by `\mark_insert:nn`. It will not show up in the output but its argument (a counter value that is incremented) makes all marks unique so the test for “equal” is not fooled by two different marks having the same mark text.

```

263 \cs_new_protected:Npn \__mark_value:nn #1#2 { #2 }

```

(End of definition for `__mark_value:nn`.)

`\@kernel@before@insertmark` By default `\label`, `\index`, and `\glossary` do nothing when the mark is inserted.

`insertmark`

```

264 \int_new:N \g__mark_int
265 \cs_new:Npn \@kernel@before@insertmark {
266     \cs_set_eq:NN \label \scan_stop:
267     \cs_set_eq:NN \index \scan_stop:
268     \cs_set_eq:NN \glossary \scan_stop:

```

We count each mark and use that to place a hidden marker in front of the mark text. To ensure that there is no overflow (very unlikely but you never know) we restart every 100000 marks. Thus, if somebody puts more than that number of marks on a single page you could construct a scenario in which that approach fails.

```

269     \int_compare:nNnTF \g__mark_int < {99999}
270     { \int_gincr:N \g__mark_int }
271     { \int_gzero:N \g__mark_int }
272
273 }

```

The public hook to augment the setup.

```

274 \hook_new:n {insertmark}

```

(End of definition for `\@kernel@before@insertmark` and `insertmark`.)

`\mark_use_top:nn` To retrieve the first, last or top region mark, we grab the appropriate value stored in the corresponding token list variable and pass its contents back. These functions should be used only in output routines and only after `\mark_update_structure_from_material:nn` has acted, otherwise their value will be wrong.

`\mark_use_first:nn`
`\mark_use_last:nn`

```

275 \cs_new:Npn \mark_use_first:nn #1#2 { \__mark_use_check:nnn { g__mark_#1_first_#2_tl } {#1} {#2} }
276 \cs_new:Npn \mark_use_last:nn #1#2 { \__mark_use_check:nnn { g__mark_#1_last_#2_tl } {#1} {#2} }
277 \cs_new:Npn \mark_use_top:nn #1#2 { \__mark_use_check:nnn { g__mark_#1_top_#2_tl } {#1} {#2} }

```

If used with an unknown class or region these commands will generate an error. If that happens in an expandable context then the error generation is delayed (e.g., if used in a `\section`) and happens when the code is finally used in typesetting, e.g., in the TOC or a running header. If used in a `\typeout` you only see something like `__mark_error:n{page}`. This is not too good, but probably better than low-level errors, I guess, and I don’t want to use an expandable error because of the size restrictions in such error messages.

```

278 \cs_new:Npn \__mark_use_check:nnn #1#2#3 {
279   \tl_if_eq:cNTF {#1} \relax
280     { \__mark_error:nn {#2} {#3} }
281     { \__mark_drop_id:v {#1} }
282 }

```

Each mark starts with an id and while the id does not print it is nevertheless better to remove it when returning the mark, so that downstream manipulation of the data doesn't have to deal with it. This is what the `\exp_not:o` accomplishes.

```

283 \cs_new:Npn \__mark_drop_id:n #1 { \exp_not:o { #1 } }
284 \cs_generate_variant:Nn \__mark_drop_id:n { o, v }

```

(End of definition for `\mark_use_top:nn`, `\mark_use_first:nn`, and `\mark_use_last:nn`. These functions are documented on page 1005.)

8.4 Comparing mark values

`\mark_if_eq:nnnnTF` Test if in a given region (#1) for a given class (#2) the marks in position #3 and #4 (top, first, or last) are identical

```

285 \prg_new_conditional:Npnn \mark_if_eq:nnnn #1#2#3#4 { T , F , TF }
286 {
287   \tl_if_eq:ccTF { g__mark_ #1 _#3_ #2 _t1 }
288                 { g__mark_ #1 _#4_ #2 _t1 }
289                 \prg_return_true:
290                 \prg_return_false:
291 }

```

The fully general test (with two triplets of the form `<region>`, `<class>`, and `<position>`) is this:

```

292 \prg_new_conditional:Npnn \mark_if_eq:nnnnnn #1#2#3#4#5#6 { T , F , TF }
293 {
294   \tl_if_eq:ccTF { g__mark_ #1 _#3_ #2 _t1 }
295                 { g__mark_ #4 _#6_ #5 _t1 }
296                 \prg_return_true:
297                 \prg_return_false:
298 }

```

(End of definition for `\mark_if_eq:nnnnTF` and `\mark_if_eq:nnnnnnTF`. These functions are documented on page 1005.)

8.5 Messages

Mark errors are L^AT_EX kernel errors:

```

299 \prop_gput:Nnn \g_msg_module_type_prop { mark } { LaTeX }
300 \msg_new:nnnn { mark } { class-already-defined }
301   { Mark~class~'#1'~already~defined }
302   {
303     \c__msg_coding_error_text_tl
304     LaTeX~was~asked~to~define~a~new~mark~class~called~'#1':~
305     this~mark~class~already~exists.
306     \c__msg_return_text_tl
307   }

```

```

308 \msg_new:nnnn { mark } { unknown-class }
309   { Unknown-mark-class-’#1’. }
310   {
311     \c__msg_coding_error_text_tl
312     LaTeX-was-asked-to-manipulate-a-mark-of-class-’#1’,~
313     but-this-class-of-marks-does-not-exist.
314   }

```

The next error can also happen if the mark class is unknown, so this should perhaps be separated into two different errors.

```

315 \msg_new:nnnn { mark } { invalid-use }
316   { Mark-region-’#1’-not-usable-or-class-’#2’-unknown }
317   {
318     \c__msg_coding_error_text_tl
319     The-region-’#1’-is-either-not-known-or-data-for-it-
320     still-needs-to-be-assembled,~e.g.,~last-column-
321     while-building-the-first-column.~
322     Also-possible:-the-class-name-’#2’-is-misspelled.
323     \c__msg_return_text_tl
324   }

```

8.6 Debugging the mark structures

Code and commands in this section are not final, it needs more experimentation to see what kind of tracing information is going to be useful in practice. For now the tracing is mainly meant to be used for code testing and not so much for application testing.

It is quite likely that the commands and the behavior of the tracing might change in the future once we gained some experience with it.

```

\g__mark_debug_bool Holds the current debugging state.
325 \bool_new:N \g__mark_debug_bool
(End of definition for \g__mark_debug_bool.)

\mark_debug_on: Turns debugging on and off by redefining \__mark_debug:n.
\mark_debug_off:
\__mark_debug:n
\__mark_debug_gset:
326 \cs_new_eq:NN \__mark_debug:n \use_none:n
327 \cs_new_protected:Npn \mark_debug_on:
328   {
329     \bool_gset_true:N \g__mark_debug_bool
330     \__mark_debug_gset:
331   }
332 \cs_new_protected:Npn \mark_debug_off:
333   {
334     \bool_gset_false:N \g__mark_debug_bool
335     \__mark_debug_gset:
336   }
337 \cs_new_protected:Npn \__mark_debug_gset:
338   {
339     \cs_gset_protected:Npx \__mark_debug:n ##1
340     { \bool_if:NT \g__mark_debug_bool {##1} }
341   }

```

(End of definition for \mark_debug_on: and others. These functions are documented on page 1008.)

`\DebugMarksOn` CamelCase commands for debugging.

`\DebugMarksOff`

```
342 \cs_new_eq:NN \DebugMarksOn \mark_debug_on:
343 \cs_new_eq:NN \DebugMarksOff \mark_debug_off:
```

(End of definition for `\DebugMarksOn` and `\DebugMarksOff`. These functions are documented on page 1008.)

`__mark_class_status:nnn` Shows the mark values across all regions for one mark class (#2).

The first argument gives some *info* to help in identifying where the command was called, the second is the class and the third holds the number of `mcol-...` we should display: inside a `multicols` environment this will be `\col@number`, in L^AT_EX's normal output routines it will be 0.

```
344 (*trace)
345 \cs_new_protected:Npn \__mark_class_status:nnn #1#2#3 {
346   \typeout{ Marks:~#2~ #1:}
347   \__mark_region_status:nnn {#2}{ page~ (previous) } { previous-page }
348   \__mark_region_status:nnn {#2}{ page~ (current)~ } { page }
349   \__mark_region_status:nnn {#2}{ column~ (previous) }{ previous-column }
350   \__mark_region_status:nnn {#2}{ column~ (current)~ }{ column }
351   \__mark_region_status:nnn {#2}{ column~ (first) } { first-column }
352   \__mark_region_status:nnn {#2}{ column~ (last)~ } { last-column }
```

Then finish by displaying a subset of the `mcol-...` regions: none (0) in the standard L^AT_EX output routine and `\col@number` within a `multicols` environment.

```
353   \int_step_inline:nn {#3}
354   {
355     \__mark_region_status:nnn {#2}{ column~ (##1)~ } { mcol-##1 }
356   }
357 }
```

(End of definition for `__mark_class_status:nnn`.)

`__mark_region_status:nnn` Display the top, first, and last mark of a region unless none of them exist or all of them are empty.

```
358 \cs_new_protected:Npn \__mark_region_status:nnn #1#2#3 {
359   \group_begin:
360   \cs_set:Npn \__mark_value:nn ##1##2{ \exp_not:n{ {##1} ~ ##2 } }
361   \tl_if_exist:cT { g__mark_#3_last_ #1 _tl }
362   {
363     \tl_if_eq:cNF { g__mark_#3_last_ #1 _tl } \c__mark_empty_tl
364     {
365       \typeout{\@spaces #2 =
366         ~|~ \use:c { g__mark_#3_top_ #1 _tl } ~|~
367         \use:c { g__mark_#3_first_ #1 _tl } ~|~
368         \use:c { g__mark_#3_last_ #1 _tl } ~|
369     }
370   }
371 }
372 \group_end:
373 }
```

(End of definition for `__mark_region_status:nnn`.)

`__mark_status:nn` Show a snapshot of all mark class values across all regions. The first argument is a string to identify the output, the second argument is the number of `mcol-...` regions to show. Outside of a `multicols` environment this is normally set to 0.

```

374 \cs_new_protected:Npn \__mark_status:nn #1#2
375   {
376     \seq_map_inline:Nn \g__mark_classes_seq
377       { \__mark_class_status:nnn {#1} {##1} {#2} }
378   }
379 </trace>

```

(End of definition for __mark_status:nn.)

`\ShowMarksAt` Debugging helper that displays a snapshot of all known mark structures. The first argument is a text string that is displayed to help identifying when the snapshot was made. The optional second one determines how many `mcol-...` regions are displayed (by default 4).

This may not stay like this (or at all), which is why it isn't yet documented as an official command.

```

380 \NewDocumentCommand \ShowMarksAt {m O{4} } {
381 <*trace>
382   \__mark_debug:n { \__mark_status:nn {#1}{#2} }
383 </trace>
384 }

```

(End of definition for \ShowMarksAt.)

8.7 Designer-level interfaces

`\NewMarkClass` These two are identical to the L3 programming layer commands.

```

\InsertMark
385 \cs_new_eq:NN \NewMarkClass \mark_new_class:n
386 \@onlypreamble \NewMarkClass
387 \cs_new_eq:NN \InsertMark \mark_insert:nn

```

(End of definition for \NewMarkClass and \InsertMark. These functions are documented on page 1004.)

`\TopMark` The following commands take an optional argument that defaults to page. There is no
`\FirstMark` checking that the region is actually valid. If not there is simply an empty return.
`\LastMark`

```

388 \NewExpandableDocumentCommand \FirstMark { O{page} m }
389   { \mark_use_first:nn {#1}{#2} }
390 \NewExpandableDocumentCommand \LastMark { O{page} m }
391   { \mark_use_last:nn {#1}{#2} }
392 \NewExpandableDocumentCommand \TopMark { O{page} m }
393   { \mark_use_top:nn {#1}{#2} }

```

(End of definition for \TopMark, \FirstMark, and \LastMark. These functions are documented on page 1005.)

`\IfMarksEqualTF` We only provide CamelCase commands for the case with one region (optional) and one class. One could think of also providing a version for the general case with several optional arguments, but use cases for this are most likely rare, so not done yet.

```

394 \NewExpandableDocumentCommand \IfMarksEqualTF {0{page}mmm} {
395   \mark_if_eq:nnnnTF {#1}{#2}{#3}{#4}
396 }
397 \NewExpandableDocumentCommand \IfMarksEqualT {0{page}mmm} {
398   \mark_if_eq:nnnnT {#1}{#2}{#3}{#4}
399 }
400 \NewExpandableDocumentCommand \IfMarksEqualF {0{page}mmm} {
401   \mark_if_eq:nnnnF {#1}{#2}{#3}{#4}
402 }

```

(End of definition for `\IfMarksEqualTF`, `\IfMarksEqualT`, and `\IfMarksEqualF`. These functions are documented on page 1005.)

9 L^AT_εX 2_ε integration

9.1 Core L^AT_εX 2_ε integration

`_mark_update_singlecol_structures`: This command updates the mark structures if we are producing a single column document.

```

403 \cs_new_protected:Npn \_mark_update_singlecol_structures: {

```

First we update the page region (which also updates the previous-page).

The `\@outputbox` is normally in `\vbox` in L^AT_εX but we can't take that for granted (an `amsmath` test document changed it to an `\hbox` just to trip me up) so we are a little careful with `unpack` now.

```

404   \box_if_vertical:NTF \@outputbox
405   {
406     \mark_update_structure_from_material:nn {page}
407     { \vbox_unpack:N \@outputbox }
408   }
409   {
410     \mark_update_structure_from_material:nn {page}
411     { \hbox_unpack:N \@outputbox }
412   }

```

Then we provide the necessary updates for the aliases.

```

413   \mark_copy_structure:nn {previous-column}{previous-page}
414   \mark_copy_structure:nn {column}{page}
415   \mark_copy_structure:nn {first-column}{page}
416   \mark_copy_structure:nn {last-column}{page}
417   <*trace>
418   % move this into status itself?
419   \_mark_debug:n
420   {
421     \_mark_status:nn
422     { in~ OR~ (
423       \legacy_if:nTF {@twoside}
424       { twoside-
425         \int_if_odd:nTF \c@page
426         { odd }{ even }

```

```

427             }
428             { onside }
429         )
430     }
431     {0}
432 }
433 </trace>
434 }

```

(End of definition for `_mark_update_singlecol_structures:`.)

`_mark_update_dblcol_structures:` This commands handles the updates if we are doing two-column pages.

```

435 \cs_new_protected:Npn \_mark\_update\_dblcol\_structures: {

```

First we update the `column` and `previous-column` regions using the material assembled in `\@outputbox`.

```

436   \box_if_vertical:NTF \@outputbox
437     {
438       \mark\_update\_structure\_from\_material:nn {column}
439       { \vbox\_unpack:N \@outputbox }
440     }
441     {
442       \mark\_update\_structure\_from\_material:nn {column}
443       { \hbox\_unpack:N \@outputbox }
444     }

```

How we have to update the alias regions depends on whether or not `\@opcol` was called to process the first column or to produce the completed page

```

445   \legacy\_if:nTF {@firstcolumn}
446     {

```

If we are processing the first column then `column` is our `first-column` and there is no `last-column` yet, so we make those an error.

```

447       \mark\_copy\_structure:nn {first-column}{column}
448       \mark\_set\_structure\_to\_err:n {last-column}
449     }
450     {

```

If we produce the completed page then the `first-column` is the same as the new `previous-column`. However, the structure should already be correct if you think about it (because it was set to `column` last time which is now the `previous-column`), thus there is no need to make an update.

```

451 %   \mark\_copy\_structure:nn {first-column}{previous-column}

```

However, we now have a proper `last-column` so we assign that.

```

452   \mark\_copy\_structure:nn {last-column}{column}

```

What now remains doing is to update the `page` and `previous-page` regions. For this we have to copy the settings in `page` into `previous-page` and then update `page` such that the top and first marks are taken from the `first-column` region and the last marks are taken from the `last-column` region. All this has to be done for all mark classes so we loop over our sequence.

Note that one loop is needed if we arrange the copy statements in a suitable way.

```

453   \seq\_map\_inline:Nn \g\_mark\_classes\_seq
454     {

```


The `previous-page` updates need to come before the updates for `page` region because otherwise the values to copy are already overwritten. necessary values.

```

455     \tl_gset_eq:cc { g__mark_previous-page_top_   ##1 _tl }
456                   { g__mark_page_top_           ##1 _tl }
457     \tl_gset_eq:cc { g__mark_previous-page_first_ ##1 _tl }
458                   { g__mark_page_first_         ##1 _tl }
459     \tl_gset_eq:cc { g__mark_previous-page_last_  ##1 _tl }
460                   { g__mark_page_last_         ##1 _tl }

```

To update the `top` we only have to copy what is in `first-column`:

```

461     \tl_gset_eq:cc { g__mark_page_top_           ##1 _tl }
462                   { g__mark_first-column_top_   ##1 _tl }
463

```

Updating the `first` mark for the `page` region is more complicated. We first have to find out if there is any mark in the first column (this can be done by comparing the `top` and the `first` mark of that region).

```

464     \tl_if_eq:ccTF { g__mark_first-column_top_   ##1 _tl }
465                   { g__mark_first-column_first_ ##1 _tl }
466     {

```

If there is no mark in the first column we copy the first mark of the last column. If that doesn't contain a mark we still get the right result because the first mark is then equal to the top mark.

```

467         \tl_gset_eq:cc { g__mark_page_first_     ##1 _tl }
468                       { g__mark_last-column_first_ ##1 _tl }
469     }
470     {

```

On the other hand, if there is a mark in the first column we copy over the `first` mark from that column.

```

471         \tl_gset_eq:cc { g__mark_page_first_     ##1 _tl }
472                       { g__mark_first-column_first_ ##1 _tl }
473     }

```

The logic for the `last` page mark is again simple, we can just copy the value in the `last` mark of the last column. If that column doesn't contain any marks, then the value in `last` will be automatically the same as the `last` from the first column.

```

474     \tl_gset_eq:cc { g__mark_page_last_         ##1 _tl }
475                   { g__mark_last-column_last_  ##1 _tl }
476   }
477 }

```

478 `<{*trace}`

```

479   \__mark_debug:n
480   {
481     \__mark_status:nn
482     { in~ OR~ (
483       \legacy_if:nTF {@twoside}
484       { twoside-
485         \int_if_odd:nTF \c@page
486         { odd }{ even }
487       }
488       { onside }
489       \space
490       \legacy_if:nTF {@firstcolumn}

```

```

491             { first~ }{ second~ }
492             column )
493         }
494     {0}
495 }
496 \
```

(End of definition for `_mark_update_dblcol_structures:`.)

9.2 Other L^AT_EX 2_ε output routines

This section will cover support for packages that alter the L^AT_EX output routine (as necessary). The support for multicols (for now) is handled directly in that package.

```

498 \@@=
```

`\@expl@@mark@update@singlecol@structures@@`

```

499 \cs_new_eq:NN \@expl@@mark@update@singlecol@structures@@
500     \_mark_update_singlecol_structures:
```

(End of definition for `\@expl@@mark@update@singlecol@structures@@`.)

`\@expl@@mark@update@dblcol@structures@@`

```

501 \cs_new_eq:NN \@expl@@mark@update@dblcol@structures@@
502     \_mark_update_dblcol_structures:
```

(End of definition for `\@expl@@mark@update@dblcol@structures@@`.)

9.3 Rollback information

```

503 \latexrelease\IncludeInRelease{0000/00/00}{ltmarks}%
504 \latexrelease          {Undo-Marks-handling}
505 \latexrelease
```

We keep the interface commands around even if we roll back in case they are used in packages that don't roll back. Not likely to do a lot of good, but then there is not much we can do, but this at least they won't give unknown csname errors.

```

506 \latexrelease\DeclareRobustCommand \NewMarkClass[1]{}
507 \latexrelease\DeclareRobustCommand \InsertMark[2]{}
508 \latexrelease\RenewExpandableDocumentCommand \FirstMark { 0{} m } { }
509 \latexrelease\RenewExpandableDocumentCommand \LastMark { 0{} m } { }
510 \latexrelease\RenewExpandableDocumentCommand \TopMark { 0{} m } { }
511 \latexrelease\RenewExpandableDocumentCommand \IfMarksEqualTF { 0{} mmm }{ }
512 \latexrelease
```

Same here, this avoided extra roll back code in the OR.

```

513 \latexrelease\let \@expl@@mark@update@singlecol@structures@@ \relax
514 \latexrelease\let \@expl@@mark@update@dblcol@structures@@ \relax
515 \latexrelease
516 \latexrelease
517 \latexrelease\EndModuleRelease
518 \ExplSyntaxOff
519 \
```

Reset module prefix:

```

520 \@@=
```

File 49

ltpage.dtx

1 Page styles and related commands

1.1 Page Style Commands

`\pagestyle{<style>}` : sets the page style of the current and succeeding pages to *style*

`\thispagestyle{<style>}` : sets the page style of the current page only to *style*.

To define a page style *style*, you must define `\ps@style` to set the page style parameters.

1.2 How a page style makes running heads and feet

The `\ps@...` command defines the macros `\@oddhead`, `\@oddfoot`, `\@evenhead`, and `\@evenfoot` to define the running heads and feet. (See output routine.) To make headings determined by the sectioning commands, the page style defines the commands `\chaptermark`, `\sectionmark`, etc., where `\chaptermark{<text>}` is called by `\chapter` to set a mark. The `\...mark` commands and the `\...head` macros are defined with the help of the following macros.

(All the `\...mark` commands should be initialized to no-ops.)

1.3 marking conventions

L^AT_EX extends T_EX's `\mark` facility by producing two kinds of marks a 'left' and a 'right' mark, using the following commands:

`\markboth{<left>}{<right>}` : Adds both marks.

`\markright{<right>}` : Adds a 'right' mark.

`\leftmark` : Used in the output routine, gets the current 'left' mark. Works like T_EX's `\botmark`.

`\rightmark` : Used in the output routine, gets the current 'right' mark. Works like T_EX's `\firstmark`. The marking commands work reasonably well for right marks 'numbered within' left marks—e.g., the left mark is changed by a `\chapter` command and the right mark is changed by a `\section` command. However, it does produce somewhat anomalous results if 2 `\markboth`'s occur on the same page.

Commands like `\tableofcontents` that should set the marks in some page styles use a `\@mkboth` command, which is `\let` by the `pagestyle` command (`\ps@...`) to `\markboth` for setting the heading or to `\@gobbletwo` to do nothing.

1 `<*2kernel>`

`\pagestyle` User command to set the page style for this and following pages.

```
2 \def\pagestyle#1{%
3   \@ifundefined{ps@#1}%
4     \undefinedpagestyle
5     {\@nameuse{ps@#1}}
```

(End of definition for `\pagestyle`.)

`\thispagestyle` User command to set the page style for this page only.

```
6 \def\thispagestyle#1{%
7   \ifundefined{ps@#1}%
8     \undefinedpagestyle
9     {\global\@specialpagetrue\gdef\@specialstyle{#1}}
```

(End of definition for \thispagestyle.)

`\ps@empty` The empty page style: No head or foot line.

```
10 \def\ps@empty{%
11   \let\@mkboth\@gobbletwo\let\@oddhead\@empty\let\@oddfoot\@empty
12   \let\@evenhead\@empty\let\@evenfoot\@empty}
```

(End of definition for \ps@empty.)

`\ps@plain` The plain page style: No head, centred page number in foot.

```
13 \def\ps@plain{\let\@mkboth\@gobbletwo
14   \let\@oddhead\@empty\def\@oddfoot{\reset@font\hfil\thepage
15   \hfil}\let\@evenhead\@empty\let\@evenfoot\@oddfoot}
```

(End of definition for \ps@plain.)

`\@leftmark` We implement `\@leftmark` and `\@rightmark` in terms of already defined commands to
`\@rightmark` save token space. We can't get rid of them since they are sometimes used in applications.

```
16 \let\@leftmark\@firstoftwo
17 \let\@rightmark\@secondoftwo
```

(End of definition for \@leftmark and \@rightmark.)

```
18 </2ekernel>
19 <*2ekernel | latexrelease>
20 <latexrelease>\IncludeInRelease{2025/06/01}%
21 <latexrelease>          {\markboth}{Drop legacy mark support}%
```

`\markboth` User commands for setting L^AT_EX marks.

`\markright` Test for `\@nobreak` added 15 Apr 86 in `\markboth` and `\markright` letting `\label` and `\index` to `\relax` added 22 Feb 86 so these commands can appear in sectioning command arguments RmS 91/06/21 Same for `\glossary`

```
22 \ExplSyntaxOn
23 \DeclareRobustCommand*\markboth[2]{%
24   \mark_insert:nn{2e-left}{#1}
25   \mark_insert:nn{2e-right}{#2}
26   \tl_if_empty:nF{#2}{ \mark_insert:nn{2e-right-nonempty}{#2} }
27 }
28 \DeclareRobustCommand*\markright[1]{%
29   \mark_insert:nn{2e-right}{#1}
30   \tl_if_empty:nF{#1}{ \mark_insert:nn{2e-right-nonempty}{#1} }
31 }
32 \ExplSyntaxOff
```

(End of definition for \markboth and \markright. These functions are documented on page 1009.)

```
33 </2ekernel | latexrelease>
34 <latexrelease>\EndIncludeInRelease
35 <latexrelease>\IncludeInRelease{2022/06/01}%
36 <latexrelease>          {\markboth}{New mark support}%
37 <latexrelease>\ExplSyntaxOn
38 <latexrelease>\DeclareRobustCommand*\markboth[2]{%
39 <latexrelease>  \begingroup
40 <latexrelease>    \let\label\relax \let\index\relax \let\glossary\relax
41 <latexrelease>    \unrestored@protected@xdef\@themark {#1}{#2}%
42 <latexrelease>    \@temptokena \expandafter{\@themark}%
43 <latexrelease>    \mark_insert:nn{2e-left}{#1}
44 <latexrelease>    \mark_insert:nn{2e-right}{#2}
45 <latexrelease>    \tl_if_empty:nF{#2}{ \mark_insert:nn{2e-right-nonempty}{#2} }
46 <latexrelease>    \mark{\the\@temptokena}%
47 <latexrelease>  \endgroup
48 <latexrelease>  \if@nbreak\ifvmode\nobreak\fi\fi}
49 <latexrelease>\DeclareRobustCommand*\markright[1]{%
50 <latexrelease>  \begingroup
51 <latexrelease>    \let\label\relax \let\index\relax \let\glossary\relax
52 <latexrelease>    \expandafter\@markright\@themark {#1}%
53 <latexrelease>    \@temptokena \expandafter{\@themark}%
54 <latexrelease>    \mark_insert:nn{2e-right}{#1}
55 <latexrelease>    \tl_if_empty:nF{#1}{ \mark_insert:nn{2e-right-nonempty}{#1} }
56 <latexrelease>    \mark{\the\@temptokena}%
57 <latexrelease>  \endgroup
58 <latexrelease>  \if@nbreak\ifvmode\nobreak\fi\fi}
59 <latexrelease>\ExplSyntaxOff
60 <latexrelease>\EndIncludeInRelease
61 <latexrelease>\IncludeInRelease{2019/10/01}%
62 <latexrelease>          {\markboth}{Make commands robust}%
63 <latexrelease>
64 <latexrelease>\DeclareRobustCommand*\markboth[2]{%
65 <latexrelease>  \begingroup
66 <latexrelease>    \let\label\relax \let\index\relax \let\glossary\relax
67 <latexrelease>    \unrestored@protected@xdef\@themark {#1}{#2}%
68 <latexrelease>    \@temptokena \expandafter{\@themark}%
69 <latexrelease>    \mark{\the\@temptokena}%
70 <latexrelease>  \endgroup
71 <latexrelease>  \if@nbreak\ifvmode\nobreak\fi\fi}
72 <latexrelease>\DeclareRobustCommand*\markright[1]{%
73 <latexrelease>  \begingroup
74 <latexrelease>    \let\label\relax \let\index\relax \let\glossary\relax
75 <latexrelease>    \expandafter\@markright\@themark {#1}%
76 <latexrelease>    \@temptokena \expandafter{\@themark}%
77 <latexrelease>    \mark{\the\@temptokena}%
78 <latexrelease>  \endgroup
79 <latexrelease>  \if@nbreak\ifvmode\nobreak\fi\fi}
80 <latexrelease>
81 <latexrelease>\EndIncludeInRelease
82 <latexrelease>\IncludeInRelease{0000/00/00}%
83 <latexrelease>          {\markboth}{Make commands robust}%
84 <latexrelease>
```

Using `\kernel@make@fragile` doesn't really work if there are more redefinitions later on, so we have to repeat the above definition first and then undo it (or use `\def` directly).

```

85 <latexrelease>\kernel@make@fragile\markboth
86 <latexrelease>\kernel@make@fragile\markright
87 <latexrelease>\def\markboth#1#2{%
88 <latexrelease> \begingroup
89 <latexrelease> \let\label\relax \let\index\relax \let\glossary\relax
90 <latexrelease> \unrestored@protected@xdef\@themark {#1}{#2}%
91 <latexrelease> \temptokena \expandafter{\@themark}%
92 <latexrelease> \mark{the\@temptokena}%
93 <latexrelease> \endgroup
94 <latexrelease> \if@nobreak\ifvmode\nobreak\fi\fi}
95 <latexrelease>\def\markright#1{%
96 <latexrelease> \begingroup
97 <latexrelease> \let\label\relax \let\index\relax \let\glossary\relax
98 <latexrelease> \expandafter\@markright\@themark {#1}%
99 <latexrelease> \temptokena \expandafter{\@themark}%
100 <latexrelease> \mark{the\@temptokena}%
101 <latexrelease> \endgroup
102 <latexrelease> \if@nobreak\ifvmode\nobreak\fi\fi}
103 <latexrelease>
104 <latexrelease>\EndIncludeInRelease
105 <*2ekernel>

```

`\leftmark`
`\rightmark`

```

106 </2ekernel>
107 <*2ekernel | latexrelease>
108 <latexrelease>\IncludeInRelease{2025/06/01}%
109 <latexrelease> \leftmark{Use new mark mechanism}%
110 \ExplSyntaxOn
111 \cs_new:Npn \leftmark {\mark_use_last:nn{page}{2e-left}}
112 \cs_new:Npn \rightmark {\mark_use_first:nn{page}{2e-right}}
113 \ExplSyntaxOff
114 </2ekernel | latexrelease>
115 <latexrelease>\EndIncludeInRelease
116 <latexrelease>\IncludeInRelease{0000/00/00}%
117 <latexrelease> \leftmark{Use new mark mechanism}%
118 <latexrelease>
119 <latexrelease>\def\leftmark{\expandafter\@leftmark\botmark\@empty\@empty}
120 <latexrelease>\def\rightmark{\expandafter\@rightmark\firstmark\@empty\@empty}

```

```

121 <latexrelease>\def\@themark{#1}
122 <latexrelease>\def\@markright#1#2#3{\@temptokena {#1}%
123 <latexrelease>\unrestored@protected@xdef\@themark{\the\@temptokena}{#3}}
124 <latexrelease>\EndIncludeInRelease
125 <*2ekernel>

```

(End of definition for `\leftmark` and others. These functions are documented on page 1009.)

`\raggedbottom` `\raggedbottom` typesets pages with no vertical stretch, so they have their natural height instead of all being exactly the same height. (Uses a space of `.0001fil` to avoid interfering with the `1fil` space of `\newpage`.)

```

126 \DeclareRobustCommand\raggedbottom{%
127   \def\@textbottom{\vskip \z@ \@plus.0001fil}\let\@texttop\relax}

```

(End of definition for \raggedbottom.)

\flushbottom \flushbottom: Inverse of \raggedbottom — makes all pages the same height.

```

128 \DeclareRobustCommand\flushbottom{%
129   \let\@textbottom\relax \let\@texttop\relax}

```

(End of definition for \flushbottom.)

\sloppy \sloppy will never (well, hardly ever) produce overfull boxes, but may produce underfull ones. (14 June 85)

```

130 \DeclareRobustCommand\sloppy{%
131   \tolerance 9999%
132   \emergencystretch 3em%
133   \hfuzz .5\p@
134   \vfuzz\hfuzz}

```

(End of definition for \sloppy.)

sloppypar (*env.*) A sloppypar environment is equivalent to {\par \sloppy ... \par}.

```

135 \def\sloppypar{\par\sloppy}
136 \def\endsloppypar{\par}

```

\fussy Resets TeX's parameters to their normal finicky values.

```

137 \DeclareRobustCommand\fussy{%
138   \emergencystretch\z@
139   \tolerance 200%
140   \hfuzz .1\p@
141   \vfuzz\hfuzz}

```

(End of definition for \fussy.)

\overfullrule L^AT_EX default is no overfull box rule. Changed by document class option.

```

142 \overfullrule 0pt

```

(End of definition for \overfullrule.)

```

143 </2ekernel>

```

File 50

ltclass.dtx

1 Introduction

This file implements the following declarations, which replace `\documentstyle` in $\text{\LaTeX} 2_{\epsilon}$ documents.

Note that old documents containing `\documentstyle` will be run using a compatibility option—thus keeping everyone happy, we hope!

The overall idea is that there are two types of ‘style files’: ‘class files’ which define elements and provide a default formatting for them; and ‘packages’ which provide extra functionality. One difference between $\text{\LaTeX} 2_{\epsilon}$ and $\text{\LaTeX} 2.09$ is that $\text{\LaTeX} 2_{\epsilon}$ packages may have options. Note that options to classes packages may be implemented such that they input files, but these file names are not necessarily directly related to the option name.

2 User interface

```
\documentclass[<main-option-list>]{<class>}[<version>]
```

There must be exactly one such declaration, and it must come first. The *<main-option-list>* is a list of options which can modify the formatting of elements which are defined in the *<class>* file as well as in all following `\usepackage` declarations (see below). The *<version>* is a version number, beginning with a date in the format YYYY/MM/DD. If an older version of the class is found, a warning is issued.

```
\documentstyle[<main-option-list>]{<class>}[<version>]
```

The `\documentstyle` declaration is kept in order to maintain upward compatibility with $\text{\LaTeX} 2.09$ documents. It is similar to `\documentclass`, but it causes all options in *<main-option-list>* that the *<class>* does not use to be passed to `\RequirePackage` after the options have been processed. This maintains compatibility with the 2.09 behaviour. Also a flag is set to indicate that the document is to be processed in $\text{\LaTeX} 2.09$ compatibility mode. As far as most packages are concerned, this only affects the warnings and errors \LaTeX generates. This flag does affect the definition of font commands, and `\sloppy`.

```
\usepackage[<package-option-list>]{<package-list>}[<version>]
```

There can be any number of these declarations. All packages in *<package-list>* are called with the same options.

Each *<package>* file defines new elements (or modifies those defined in the *<class>*), and thus extends the range of documents which can be processed. The *<package-option-list>* is a list of options which can modify the formatting of elements defined in the *<package>* file. The *<version>* is a version number, beginning with a date in the format YYYY/MM/DD. If an older version of the package is found, a warning is issued.

Each package is loaded only once. If the same package is requested more than once, nothing happens, unless the package has been requested with options that were not given the first time it was loaded, in which case an error is produced.

As well as processing the options given in the $\langle package-option-list \rangle$, each package processes the $\langle main-option-list \rangle$. This means that options that affect all of the packages can be given globally, rather than repeated for every package.

Note that class files have the extension `.cls`, packages have the extension `.sty`.

`filecontents` (*env.*)

The environment `filecontents` is intended for passing the contents of packages, options, or other files along with a document in a single file. It has one argument, which is the name of the file to create. If that file already exists (maybe only in the current directory if the OS supports a notion of a ‘current directory’ or ‘default directory’) then nothing happens (except for an information message) and the body of the environment is bypassed. Otherwise, the body of the environment is written verbatim to the file name given as the first argument, together with some comments about how it was produced.

The environment can also be called with an optional argument which is used to alter some of its behavior: option `force` or `overwrite` will allow for overwriting existing files, option `nosearch` will only check the current directory when looking if the file exists. This can be useful if you want to generate a local (modified) copy of some file that is already in the search tree of \TeX . Finally, you can use `noheader` to prevent it from writing the standard blurb at the top of the file (this is actually the same as using the star form of the environment).

The environment is now allowed anywhere in the document, but to ensure that all packages or options necessary are available when the document is run, it is normally best to place it at the top of your file (before `\documentclass`). A possible use case for using it inside the document body is if you want to reuse some text several times in the document you could then write it and later use `\input` to retrieve it where needed.

The begin and end tags should each be on a line by itself.

2.1 Option processing

When the options are processed, they are divided into two types: *local* and *global*:

- For a class, the options in the `\documentclass` command are local.
- For a package, the options in the `\usepackage` command are local, and the options in the `\documentclass` command are global.

The options for `\documentclass` and `\usepackage` are processed in the following way:

1. The local and global options that have been declared (using `\DeclareOption` as described below) are processed first.

In the case of `\ProcessOptions`, they are processed in the order that they were declared in the class or package.

In the case of `\ProcessOptions*`, they are processed in the order that they appear in the option-lists. First the global options, and then the local ones.

2. Any remaining local options are dealt with using the default option (declared using the `\DeclareOption*` declaration described below). For document classes, this usually does nothing, but records the option on a list of unused options. For packages, this usually produces an error.

Finally, when `\begin{document}` is reached, if there are any global options which have not been used by either the class or any package, the system will produce a warning.

3 Class and Package interface

3.1 Class name and version

`\ProvidesClass` A class can identify itself with the `\ProvidesClass{<name>}[<version>]` command. The `<version>` should begin with a date in the format YYYY/MM/DD.

3.2 Package name and version

`\ProvidesPackage` A package can identify itself with the `\ProvidesPackage{<name>}[<version>]` command. The `<version>` should begin with a date in the format YYYY/MM/DD.

3.3 Requiring other packages

`\RequirePackage` Packages or classes can load other packages using `\RequirePackage[<options>]{<name>}[<version>]`. If the package has already been loaded, then nothing happens unless the requested options are not a subset of the options with which it was loaded, in which case an error is called.

`\LoadClass` Similar to `\RequirePackage`, but for classes, may not be used in package files.

`\PassOptionsToPackage` Packages can pass options to other packages using:

`\PassOptionsToPackage{<options>}{<package>}`.

`\PassOptionsToClass` This adds the `<options>` to the options list of any future `\RequirePackage` or `\usepackage` command. For example:

```
\PassOptionsToPackage{foo,bar}{fred}
```

is the same as:

```
\RequirePackage[foo,bar,baz]{fred}
```

`\LoadClassWithOptions` `\LoadClassWithOptions{<name>}[<version>]:`

This is similar to `\LoadClass`, but it always calls class `<name>` with exactly the same option list that is being used by the current class, rather than an option explicitly supplied or passed on by `\PassOptionsToClass`. `\RequirePackageWithOptions` is the analogous command for packages.

This is mainly intended to allow one class to simply build on another, for example:

```
\LoadClassWithOptions{article}
```

This should be contrasted with the slightly different construction

```
\DeclareOption*{\PassOptionsToClass{\CurrentOption}{article}}
\ProcessOptions
\LoadClass{article}
```

As used here, the effects are more or less the same, but the version using `\LoadClassWithOptions` is slightly quicker (and less to type). If, however, the class declares options of its own then the two constructions are different; compare, for example:

```
\DeclareOption{landscape}{...}
\ProcessOptions
\LoadClassWithOptions{article}
```

with:

```
\DeclareOption{landscape}{...}
\DeclareOption*{\PassOptionsToClass{\CurrentOption}{article}}
\ProcessOptions
\LoadClass{article}
```

In the first case, the `article` class will be called with option `landscape` precisely when the current class is called with this option; but in the second example it will not as in that case `article` is only passed options by the default option handler, which is not used for `landscape` as that option is explicitly declared.

```
\IfPackageLoadedTF      To find out if a package has already been loaded, use
\IfClassLoadedTF        \IfPackageLoadedTF{<package>}{<true>}{<false>}
\@ifpackageloaded      \IfPackageLoadedTF{<package>}{<true>}{<false>}
\@ifclassloaded
```

or the old name `\@ifpackageloaded`.

```
\IfPackageAtLeastTF    To find out if a package has already been loaded with a version equal to or more
\IfClassAtLeastTF      recent than <date>, use
\IfFileAtLeastTF        \IfPackageAtLeastTF{<package>}{<date>}{<true>}{<false>}
\@ifpackagelater        \IfPackageAtLeastTF{<package>}{<date>}{<true>}{<false>}
\@ifclasslater          or the old name \@ifpackagelater.
\IfFormatAtLeastTF     To test the format date use
                        \IfFormatAtLeastTF{<date>}{<true>}{<false>}
```

```
\IfPackageLoadedWithOptionsTF  To find out if a package has already been loaded with at least the options <options>,
\IfClassLoadedWithOptionsTF use
\@ifpackagewith                \IfPackageLoadedWithOptionsTF{<package>}{<options>}{<true>}{<false>}
\@ifclasswith
```

or the old name `\@ifpackagewith`.

There exists one package that can't be tested with the above commands: the `fontenc` package pretends that it was never loaded to allow for repeated reloading with different options (see `ltoutenc.dtx` for details).

3.4 Declaring new options

Options for classes and packages are built using the same macros.

```
\DeclareOption      To define a builtin option, use \DeclareOption{<name>}{<code>}.
\DeclareOption*     To define the default action to perform for local options which have not been declared,
                    use \DeclareOption*{<code>}.
```

Note: there should be no use of

```
\RequirePackage, \DeclareOption, \DeclareOption* or \ProcessOptions
inside \DeclareOption or \DeclareOption*.
```

Possible uses for `\DeclareOption*` include:

```
\DeclareOption*{}
```

Do nothing. Silently accept unknown options. (This suppresses the usual warnings.)

```
\DeclareOption*{\@unkownoptionerror}
```

Complain about unknown local options. (The initial setting for package files.)

```
\DeclareOption*{\PassOptionsToPackage{\CurrentOption}{<pkg-name>}}
```

Handle the current option by passing it on to the package `<pkg-name>`, which will presumably be loaded via `\RequirePackage` later in the file. This is useful for building

‘extension’ packages, that perhaps handle a couple of new options, but then pass everything else on to an existing package.

```
\DeclareOption*{\InputIfFileExists{xx-\CurrentOption.yyy}%
  {}%
  {\OptionNotUsed}}
```

Handle the option `foo` by loading the file `xx-foo.yyy` if it exists, otherwise do nothing, but declare that the option was not used. Actually the `\OptionNotUsed` declaration is only needed if this is being used in class files, but does no harm in package files.

3.5 Safe Input Macros

```
\InputIfFileExists \InputIfFileExists{<file>}{<then>}{<else>}
Inputs <file> if it exists. Immediately before the input, <then> is executed. Otherwise
<else> is executed.
  \IfFileExists    As above, but does not input the file.
                   One thing you might like to put in the <else> clause is
\@missingfileerror This starts an interactive request for a filename, supplying default extensions. Just
hitting return causes the whole input to be skipped and entering x quits the current run,
  \input           This has been redefined from the LATEX 2.09 definition, in terms of the new commands
\InputIfFileExists and \@missingfileerror.
  \listfiles       Giving this declaration in the preamble causes a list of all files input via the ‘safe
input’ commands to be listed at the end. Any strings specified in the optional argument
to \ProvidesPackage are listed alongside the file name. So files in standard (and other
non-standard) distributions can put informative strings in this argument.
```

4 Implementation

```
  1 <*2kernel>
\if@compatibility The flag for compatibility mode.
  2 \newif\if@compatibility
   (End of definition for \if@compatibility.)
\@documentclasshook This legacy hook is called after the first \documentclass command. It is not inte-
grated with the new 2020 hook management system! By default this checks to see if
\@normalsize is undefined, and if so, sets it to \normalsize.
  3 \def\@documentclasshook{%
  4   \ifx\@normalsize\@undefined
  5     \let\@normalsize\normalsize
  6   \fi
  7 }
   (End of definition for \@documentclasshook.)
\@declaredoptions This list is automatically built by \DeclareOption. It is the list of options (separated by
commas) declared in the class or package file and it defines the order in which the corre-
sponding \ds@<option> commands are executed. All local <option>s which are not de-
clared will be processed in the order defined by the optional argument of \documentclass
or \usepackage.
  8 \let\@declaredoptions\@empty
```

(End of definition for \@declaredoptions.)

`\@classoptionslist` List of options of the main class.
9 `\let\@classoptionslist\relax`
10 `%\@onlypreamble\@classoptionslist`
(End of definition for \@classoptionslist.)

`\@raw@classoptionslist` List of options of the main class (unprocessed).
11 `\let\@raw@classoptionslist\relax`
(End of definition for \@raw@classoptionslist.)

`\@unusedoptionlist` List of options of the main class that haven't been declared or loaded as class option files.
12 `\let\@unusedoptionlist\@empty`
13 `%\@onlypreamble\@unusedoptionlist`
(End of definition for \@unusedoptionlist.)

`\CurrentOption` Name of current package or option.
14 `\let\CurrentOption\@empty`
(End of definition for \CurrentOption.)

`\@currpath` Path to the current file if explicitly given.
15 `</2kernel>`
16 `<*2kernel | latexrelease>`
17 `<latexrelease>`
18 `<latexrelease>\IncludeInRelease{2020/10/01}{\@currpath}%`
19 `<latexrelease> {Add \@currpath}%`
20 `\let\@currpath\@empty`
21 `<latexrelease>\EndIncludeInRelease`
22 `%`
23 `<latexrelease>\IncludeInRelease{0000/00/00}{\@currpath}%`
24 `<latexrelease> {Add \@currpath}%`
25 `<latexrelease>\let\@currpath\@undefined`
26 `<latexrelease>\EndIncludeInRelease`
27 `</2kernel | latexrelease>`
28 `<*2kernel>`
(End of definition for \@currpath.)

`\@currname` Name of current package or option.
29 `\let\@currname\@empty`
(End of definition for \@currname.)

`\@currentx` The current file extension.
30 `\global\let\@currentx=\@empty`
(End of definition for \@currentx.)

`\@clsextension` The two possible values of \@currentx.
`\@pkgextension` 31 `\def\@clsextension{cls}`
32 `\def\@pkgextension{sty}`

(End of definition for `\@clsextension` and `\@pkgextension`.)

```
\@pushfilename Commands to push and pop the file name and extension.
\@popfilename #1 current name.
\@currnamestack #2 current extension.
                #3 current catcode of @.
                #4 Rest of the stack.
33 </2kernel>
34 <*2kernel | latexrelease>
35 <latexrelease>
36 <latexrelease>\IncludeInRelease{2020/10/01}{\@pushfilename}%
37 <latexrelease> {Add \@expl@push@filename@@ and \@expl@push@filename@aux@@}%
38 \def\@pushfilename{%
```

The push and pop macros are injected in `\@pushfilename` and `\@popfilename` so that they correctly keep track of the hook labels.

This needs cleanup with the `expl3` interfaces also playing here, e.g., `\@expl@push@filename@@` needs cleanup and (and should probably not have this name either).

```
39 \@expl@push@filename@@
40 \xdef\@currnamestack{%
41   {\@currname}%
42   {\@currentx}%
43   {\the\catcode'\@}%
44   \@currnamestack}%
```

Temporarily add a stack for `\@currpath` here. This should be integrated in the main file stack eventually, but other packages rely on `\@currnamestack` having three elements per file, so that isn't a trivial change. The prefix `\@kernel@...` hopefully discourages people from using it.

```
45 \xdef\@kernel@currpathstack{%
46   {\@currpath}%
47   \@kernel@currpathstack}%
48 \@expl@push@filename@aux@@}
49 <latexrelease>\EndIncludeInRelease
```

The following version of `\@pushfilename` didn't formally exist in this file, but in the 2020/02/02 release, `expl3` was preloaded and it patched `\@pushfilename` (and `\@popfilename`) by adding some hooks in there. But rolling back to 2020/02/02, `expl3` doesn't patch these macros again, so rolling back has to take those hooks into account. Same goes for `\@popfilename`.

```
50 <latexrelease>
51 <latexrelease>\IncludeInRelease{2020/02/02}{\@pushfilename}%
52 <latexrelease> {Add \@expl@push@filename@@}%
53 <latexrelease>\def\@pushfilename{%
54 <latexrelease> \@expl@push@filename@@
55 <latexrelease> \xdef\@currnamestack{%
56 <latexrelease>   {\@currname}%
57 <latexrelease>   {\@currentx}%
58 <latexrelease>   {\the\catcode'\@}%
59 <latexrelease>   \@currnamestack}%
60 <latexrelease>   \@expl@push@filename@aux@@}
61 <latexrelease>\EndIncludeInRelease
62 <latexrelease>
```

When we roll back from a release that has `expl3` preloaded, the definitions of `\@pushfilename` and `\@popfilename` can't be completely rolled back otherwise `expl3`-based packages won't have the automatic `\ExplSyntaxOff` at the end. Here and below for `\@popfilename`, we don't roll back all the way through if coming from `LATEX > 2020-02-02`.

```

63 <latexrelease>\IncludeInRelease{0000/00/00}{\@pushfilename}%
64 <latexrelease> {Add \@expl@push@filename@@ and \@expl@push@filename@aux@@}%
65 <latexrelease>\ifnum\sourceLaTeXdate<20200202\relax
66 <latexrelease> \GenericInfo{#{Defining 00-00-00\string\@pushfilename.}
67 <latexrelease>\def\@pushfilename{%
68 <latexrelease> \xdef\@currnamestack{%
69 <latexrelease>   {\@currname}%
70 <latexrelease>   {\@currentx}%
71 <latexrelease>   {\the\catcode'\@}%
72 <latexrelease>   \@currnamestack}}
73 <latexrelease>\else
74 <latexrelease> \GenericInfo{#{Defining 2020-02-02\string\@pushfilename.}
75 <latexrelease>\def\@pushfilename{%
76 <latexrelease> \@expl@push@filename@@
77 <latexrelease> \xdef\@currnamestack{%
78 <latexrelease>   {\@currname}%
79 <latexrelease>   {\@currentx}%
80 <latexrelease>   {\the\catcode'\@}%
81 <latexrelease>   \@currnamestack}%
82 <latexrelease>   \@expl@push@filename@aux@@}
83 <latexrelease>\fi
84 <latexrelease>\EndIncludeInRelease
85 \@onlypreamble\@pushfilename

86 <latexrelease>
87 <latexrelease>\IncludeInRelease{2020/10/01}{\@popfilename}%
88 <latexrelease> {Add \@expl@pop@filename@@}%
89 \def\@popfilename{\@expl@@@hook@curr@name@pop@@
90   \expandafter\@p@pfilename\@currnamestack\@nil

```

Same for popping:

```

91   \expandafter\@p@pfilepath\@kernel@currpathstack\@nil
92   \@expl@pop@filename@@}
93 <latexrelease>\EndIncludeInRelease
94 <latexrelease>
95 <latexrelease>\IncludeInRelease{2020/02/02}{\@popfilename}%
96 <latexrelease> {Add \@expl@push@filename@@}%
97 <latexrelease>\def\@popfilename{\expandafter\@p@pfilename\@currnamestack\@nil
98 <latexrelease> \@expl@pop@filename@@}
99 <latexrelease>\EndIncludeInRelease
100 <latexrelease>

101 <latexrelease>\IncludeInRelease{0000/00/00}{\@popfilename}%
102 <latexrelease> {Add \@expl@push@filename@@ and \@expl@push@filename@aux@@}%
103 <latexrelease>\ifnum\sourceLaTeXdate<20200202\relax
104 <latexrelease> \GenericInfo{#{Defining 00-00-00\string\@popfilename.}
105 <latexrelease>\def\@popfilename{\expandafter\@p@pfilename\@currnamestack\@nil}
106 <latexrelease>\else
107 <latexrelease> \GenericInfo{#{Defining 2020-02-02\string\@popfilename.}

```

```

108 <latexrelease>\def\@popfilename{\expandafter\@p@pfilename\@currnamestack\@nil
109 <latexrelease> \@expl@pop@filename@@}
110 <latexrelease>\fi
111 <latexrelease>\EndIncludeInRelease
112 \@onlypreamble\@popfilename

113 </2ekernel | latexrelease>
114 <*2ekernel>

115 \def\@p@pfilename#1#2#3#4\@nil{%
116 \gdef\@currname{#1}%
117 \gdef\@current{#2}%
118 \catcode'\@#3\relax
119 \gdef\@currnamestack{#4}}
120 \@onlypreamble\@p@pfilename

121 \gdef\@currnamestack{}
122 \@onlypreamble\@currnamestack

```

(End of definition for \@pushfilename, \@popfilename, and \@currnamestack.)

`\@kernel@currpathstack` Path to the current file if explicitly given. The auxiliary is needed here to insert a `\@empty` to prevent the loss of braces.

```

123 </2ekernel>
124 <*2ekernel | latexrelease>
125 <latexrelease>
126 <latexrelease>\IncludeInRelease{2020/10/01}{\@kernel@currpathstack}%
127 <latexrelease> {Add \@kernel@currpathstack}%

```

If rolling backwards to this release, `\@kernel@currpathstack` will be defined, so the `\gdef` line should not be executed, thus the `\@gobblethree` will take it out, so the `sack` isn't touched.

```

128 <latexrelease>\@ifundefined{@kernel@currpathstack}{\@gobblethree}
129 \gdef\@kernel@currpathstack{}%

```

If rolling forward to this release, then the `\gdef` line above will define the path stack to be empty (which it can't be, inside a file), so the code below will traverse the `\@currnamestack`, and add as many empty items to `\@kernel@currpathstack` as there are items in `\@currnamestack`, so both are back in sync. Most of the time `latexrelease` is loaded on top-level, so only one item is needed, but `platexrelease` loads it internally, so the more complicated loop is needed.

```

130 <latexrelease>\ifx\@kernel@currpathstack\@empty
131 <latexrelease> \def\reserved@a#1#2#3{%
132 <latexrelease> \ifx\relax#3\else
133 <latexrelease> \g@addto@macro\@kernel@currpathstack{}}%
134 <latexrelease> \expandafter\reserved@a
135 <latexrelease> \fi}%
136 <latexrelease> \expandafter\reserved@a\@currnamestack{}}{\relax}%
137 <latexrelease>\fi
138 \def\@p@pfilepath#1{%
139 \gdef\@currpath{#1}\@p@pfilepath@aux\@empty}
140 \def\@p@pfilepath@aux#1\@nil{%
141 \xdef\@kernel@currpathstack{#1}}
142 <latexrelease>\EndIncludeInRelease
143 %
144 <latexrelease>\IncludeInRelease{0000/00/00}{\@kernel@currpathstack}%

```



```

145 <latexrelease> {Add \@kernel@currpathstack}%
146 <latexrelease>\let\@kernel@currpathstack\@undefined
147 <latexrelease>\let\@p@pfilepath\@undefined
148 <latexrelease>\let\@p@pfilepath@aux\@undefined
149 <latexrelease>\EndIncludeInRelease
150 </2ekernel | latexrelease>
151 <*2ekernel>

```

(End of definition for \@kernel@currpathstack.)

`\@optionlist` Returns the option list of the file.

```

152 \def\@optionlist#1{%
153   \@ifundefined{opt@#1}\@empty{\csname opt@#1\endcsname}}
154 %\@onlypreamble\@optionlist

```

(End of definition for \@optionlist.)

`\@ifpackageloaded` `\@ifpackageloaded{<name>}` Checks to see whether a file has been loaded.

```

\@ifclassloaded
155 \def\@ifpackageloaded{\@ifl@aded\@pkgextension}
156 \def\@ifclassloaded{\@ifl@aded\@clsextension}

157 \def\@ifl@aded#1#2{%
158   \expandafter\ifx\csname ver@#2.#1\endcsname\relax
159     \expandafter\@secondoftwo
160   \else
161     \expandafter\@firstoftwo
162   \fi}

```

(End of definition for \@ifpackageloaded and \@ifclassloaded.)

`\@ifpackagelater` `\@ifpackagelater{<name>}{YYYY/MM/DD}{<true code>}{<>false code>}` Checks that the package loaded is more recent or equal to the given date. A better name for it would therefore be `\@ifpackagelaterorequal` but it is in use for more than 30 years, so ...

```

163 \def\@ifpackagelater{\@ifl@ter\@pkgextension}
164 \def\@ifclasslater{\@ifl@ter\@clsextension}

```

(End of definition for \@ifpackagelater and \@ifclasslater.)

`\IfPackageAtLeastTF` `\IfFormatAtLeastTF{YYYY/MM/DD}{<true code>}{<>false code>}` Test if the format is later or equal to the given date.

```

\IfClassAtLeastTF
\IfFileAtLeastTF
\IfFormatAtLeastTF
165 </2ekernel>
166 <*2ekernel | latexrelease>
167 <latexrelease>\IncludeInRelease{2020/10/01}%
168 <latexrelease>           {\IfFormatAtLeastTF}{Test format date}%
169 \def\IfFormatAtLeastTF{\@ifl@t@r\fmtversion}
170 \let\IfPackageAtLeastTF\@ifpackagelater
171 \let\IfClassAtLeastTF\@ifclasslater
172 \def\IfFileAtLeastTF#1{\expandafter\@ifl@t@r\csname ver@#1\endcsname}

```

For rollback pretend it was available since the beginning of dawn.

```

173 </2ekernel | latexrelease>
174 <latexrelease>\EndIncludeInRelease
175 <latexrelease>\IncludeInRelease{0000/00/00}%
176 <latexrelease>           {\IfFormatAtLeastTF}{Test format date}%
177 <latexrelease>\def\IfFormatAtLeastTF{\@ifl@t@r\fmtversion}

```

```

178 <latexrelease>\let\IfPackageAtLeastTF\ifpackagelater
179 <latexrelease>\let\IfClassAtLeastTF\ifclasslater
180 <latexrelease>\def\IfFileAtLeastTF#1{\expandafter\@ifl@t@r\csname ver@#1\endcsname}
181 <latexrelease>\EndIncludeInRelease
182 <*2ekernel>

```

(End of definition for \IfPackageAtLeastTF and others.)

`\@ifl@ter`

```

183 \def\@ifl@ter#1#2{%
184   \expandafter\@ifl@t@r
185     \csname ver@#2.#1\endcsname}
186 </2ekernel>

```

This internal macro is also used in `\NeedsTeXFormat`.

```

187 <latexrelease>\IncludeInRelease{2018/04/01}%
188 <latexrelease>      {\@ifl@t@r}{Guard against bad input}%
189 <*2ekernel | latexrelease>
190 \def\@ifl@t@r#1#2{%
191   \ifnum\expandafter\@parse@version@#1//00\@nil<%
192     \expandafter\@parse@version@#2//00\@nil
193     \expandafter\@secondoftwo
194   \else
195     \expandafter\@firstoftwo
196   \fi}
197 \def\@parse@version@#1{\@parse@version0#1}
198 </2ekernel | latexrelease>
199 <latexrelease>\EndIncludeInRelease
200 <latexrelease>\IncludeInRelease{0000/00/00}%
201 <latexrelease>      {\@ifl@t@r}{Guard against bad input}%
202 <latexrelease>\def\@ifl@t@r#1#2{%
203 <latexrelease>  \ifnum\expandafter\@parse@version#1//00\@nil<%
204 <latexrelease>    \expandafter\@parse@version#2//00\@nil
205 <latexrelease>    \expandafter\@secondoftwo
206 <latexrelease>  \else
207 <latexrelease>    \expandafter\@firstoftwo
208 <latexrelease>  \fi}
209 <latexrelease>\let\@parse@version@\@undefined
210 <latexrelease>\EndIncludeInRelease
211 <*2ekernel>

```

(End of definition for \@ifl@ter.)

```

212 </2ekernel>
213 <*2ekernel | latexreleasefirst>
214 \def\@parse@version#1/#2/#3#4#5\@nil{%
215 \@parse@version@dash#1-#2-#3#4\@nil
216 }

```

The `\if` test here ensures that an argument with no `/` or `-` produces 0 (actually 00).

```

217 \def\@parse@version@dash#1-#2-#3#4#5\@nil{%
218   \if\relax#2\relax\else#1\fi#2#3#4 }
219 </2ekernel | latexreleasefirst>
220 <*2ekernel>

```

`\@ifpackagewith` `\@ifpackagewith{<name>}{<option-list>}` Checks that `<option-list>` is a subset of
`\@ifclasswith` the options **with** which `<name>` was loaded.

```

221 \def\@ifpackagewith{\@ifoptions\@pkgextension}
222 \def\@ifclasswith{\@ifoptions\@clsextension}

223 \def\@ifoptions#1#2{%
224   \expandtwoargs\@ifpti@ns{\@optionlist{#2.#1}}

   Probably shouldn't use \CurrentOption here... (changed to \reserved@b.)

225 </2ekernel>
226 <latexrelease>\IncludeInRelease{2017/01/01}%
227 <latexrelease>           {\@ifpti@ns}{Spaces in option clash check}%
228 <*2ekernel | latexrelease>
229 \def\@ifpti@ns#1#2{%
230   \let\reserved@a\@firstoftwo

231   \edef\reserved@b{\zap@space#2 \@empty}%
232   \@for\reserved@b:=\reserved@b\do{%
233     \ifx\reserved@b\@empty
234       \else
235         \expandafter\in@\expandafter{\expandafter,\reserved@b,}{, #1,}%
236         \ifin@
237         \else
238           \let\reserved@a\@secondoftwo
239         \fi
240       \fi
241     }%
242     \reserved@a}
243 </2ekernel | latexrelease>
244 <latexrelease>\EndIncludeInRelease
245 <latexrelease>\IncludeInRelease{0000/00/00}%
246 <latexrelease>           {\@ifpti@ns}{Spaces in option clash check}%
247 <latexrelease>\def\@ifpti@ns#1#2{%
248 <latexrelease> \let\reserved@a\@firstoftwo
249 <latexrelease> \@for\reserved@b:=#2\do{%
250 <latexrelease> \ifx\reserved@b\@empty
251 <latexrelease>   \else
252 <latexrelease>   \expandafter\in@\expandafter
253 <latexrelease>           {\expandafter,\reserved@b,}{, #1,}%
254 <latexrelease>   \ifin@
255 <latexrelease>   \else
256 <latexrelease>     \let\reserved@a\@secondoftwo
257 <latexrelease>   \fi
258 <latexrelease> \fi
259 <latexrelease> }%
260 <latexrelease> \reserved@a}
261 <latexrelease>\EndIncludeInRelease
262 <*2ekernel>

```

(End of definition for \@ifpackagewith and \@ifclasswith.)

`\IfPackageLoadedTF` More public names for the commands already available for a long time.

```

\IfPackageLoadedWithOptionsTF 263 </2ekernel>
\IfClassLoadedTF              264 <*2ekernel | latexrelease>
\IfClassLoadedWithOptionsTF  265 <latexrelease>\IncludeInRelease{2024/06/01}%

```

```

266 <latexrelease>                {\IfPackageLoadedTF}{Test package loading}%
267 \let \IfPackageLoadedTF        \@ifpackageloaded
268 \let \IfClassLoadedTF          \@ifclassloaded
269 \let \IfPackageLoadedWithOptionsTF \@ifpackagewith
270 \let \IfClassLoadedWithOptionsTF \@ifclasswith

```

For rollback/rollforward pretend everything was available since the beginning of dawn.

```

271 </2ekernel | latexrelease>
272 <latexrelease>\EndIncludeInRelease
273 <latexrelease>\IncludeInRelease{0000/00/00}%
274 <latexrelease>                {\IfPackageLoadedTF}{Test package loading}%
275 <latexrelease>
276 <latexrelease>\let \IfPackageLoadedTF        \@ifpackageloaded
277 <latexrelease>\let \IfClassLoadedTF          \@ifclassloaded
278 <latexrelease>\let \IfPackageLoadedWithOptionsTF \@ifpackagewith
279 <latexrelease>\let \IfClassLoadedWithOptionsTF \@ifclasswith
280 <latexrelease>
281 <latexrelease>\EndIncludeInRelease
282 <*2ekernel>

```

(End of definition for \IfPackageLoadedTF and others.)

```

\IfPackageLoadedT A few more conditionals for convenience
\IfPackageLoadedF 283 </2ekernel>
\IfPackageAtLeastT 284 <*2ekernel | latexrelease>
\IfPackageAtLeastF 285 <latexrelease>\IncludeInRelease{2024/06/01}%
\IfClassAtLeastT 286 <latexrelease>                {\IfPackageLoadedT}{More conditionals}%
\IfClassAtLeastF 287 \def\IfPackageLoadedT #1#2{\IfPackageLoadedTF{#1}{#2}{}}
\IfFileAtLeastT 288 \def\IfPackageLoadedF #1{\IfPackageLoadedTF{#1}{}}
\IfFileAtLeastF 289 \def\IfClassLoadedT #1#2{\IfClassLoadedTF{#1}{#2}{}}
\IfFormatAtLeastT 290 \def\IfClassLoadedF #1{\IfClassLoadedTF{#1}{}}
\IfFormatAtLeastF 291 \def\IfPackageAtLeastT#1#2#3{\IfPackageAtLeastTF{#1}{#2}{#3}{}}
\IfPackageLoadedWithOptionsT 292 \def\IfPackageAtLeastF #1#2{\IfPackageAtLeastTF{#1}{#2}{}}
\IfPackageLoadedWithOptionsF 293 \def\IfClassAtLeastT #1#2#3{\IfClassAtLeastTF{#1}{#2}{#3}{}}
\IfClassLoadedT 294 \def\IfClassAtLeastF #1#2{\IfClassAtLeastTF{#1}{#2}{}}
\IfClassLoadedF 295 \def\IfFileAtLeastT #1#2#3{\IfFileAtLeastTF{#1}{#2}{#3}{}}
\IfClassLoadedWithOptionsT 296 \def\IfFileAtLeastF #1#2{\IfFileAtLeastTF{#1}{#2}{}}
\IfClassLoadedWithOptionsF 297 \def\IfFormatAtLeastT #1#2{\IfFormatAtLeastTF{#1}{#2}{}}
\IfClassLoadedWithOptionsTF 298 \def\IfFormatAtLeastF #1{\IfFormatAtLeastTF{#1}{}}
299 \def\IfPackageLoadedWithOptionsT #1#2#3{\IfPackageLoadedWithOptionsTF{#1}{#2}{#3}{}}
300 \def\IfPackageLoadedWithOptionsF #1#2{\IfPackageLoadedWithOptionsTF{#1}{#2}{}}
301 \def\IfClassLoadedWithOptionsT #1#2#3{\IfClassLoadedWithOptionsTF{#1}{#2}{#3}{}}
302 \def\IfClassLoadedWithOptionsF #1#2{\IfClassLoadedWithOptionsTF{#1}{#2}{}}

```

These three commands haven't been there at all in the past.

```

303 \def\IfFileLoadedTF#1{%
304   \expandafter\ifx\csname ver@#1\endcsname\relax
305     \expandafter\@secondoftwo
306   \else
307     \expandafter\@firstoftwo
308   \fi}
309 \def\IfFileLoadedT#1#2{\IfFileLoadedTF{#1}{#2}{}}
310 \def\IfFileLoadedF #1{\IfFileLoadedTF{#1}{}}

```

```

\IfFileLoadedTF
\IfFileLoadedT
\IfFileLoadedF

```

For rollback/rollforward pretend everything was available since the beginning of dawn.

```

311 </2ekernel | latexrelease>
312 <latexrelease>\EndIncludeInRelease
313 <latexrelease>\IncludeInRelease{0000/00/00}%
314 <latexrelease>          {\IfPackageLoadedT}{More conditionals}%
315 <latexrelease>
316 <latexrelease>\def\IfPackageLoadedT #1#2{\IfPackageLoadedTF{#1}{#2}{}}
317 <latexrelease>\def\IfPackageLoadedF #1{\IfPackageLoadedTF{#1}{}}
318 <latexrelease>\def\IfClassLoadedT #1#2{\IfClassLoadedTF{#1}{#2}{}}
319 <latexrelease>\def\IfClassLoadedF #1{\IfClassLoadedTF{#1}{}}
320 <latexrelease>\def\IfPackageAtLeastT#1#2#3{\IfPackageAtLeastTF{#1}{#2}{#3}{}}
321 <latexrelease>\def\IfPackageAtLeastF #1#2{\IfPackageAtLeastTF{#1}{#2}{}}
322 <latexrelease>\def\IfClassAtLeastT #1#2#3{\IfClassAtLeastTF{#1}{#2}{#3}{}}
323 <latexrelease>\def\IfClassAtLeastF #1#2{\IfClassAtLeastTF{#1}{#2}{}}
324 <latexrelease>\def\IfFileAtLeastT #1#2#3{\IfFileAtLeastTF{#1}{#2}{#3}{}}
325 <latexrelease>\def\IfFileAtLeastF #1#2{\IfFileAtLeastTF{#1}{#2}{}}
326 <latexrelease>\def\IfFormatAtLeastT #1#2{\IfFormatAtLeastTF{#1}{#2}{}}
327 <latexrelease>\def\IfFormatAtLeastF #1{\IfFormatAtLeastTF{#1}{}}
328 <latexrelease>\def\IfPackageLoadedWithOptionsT #1#2#3{\IfPackageLoadedWithOptionsTF{#1}{#2}{#3}{}}
329 <latexrelease>\def\IfPackageLoadedWithOptionsF #1#2{\IfPackageLoadedWithOptionsTF{#1}{#2}{}}
330 <latexrelease>\def\IfClassLoadedWithOptionsT #1#2#3{\IfClassLoadedWithOptionsTF{#1}{#2}{#3}{}}
331 <latexrelease>\def\IfClassLoadedWithOptionsF #1#2{\IfClassLoadedWithOptionsTF{#1}{#2}{}}
332 <latexrelease>
333 <latexrelease>\def\IfFileLoadedTF#1{%
334 <latexrelease> \expandafter\ifx\csname ver@#1\endcsname\relax
335 <latexrelease> \expandafter\@secondoftwo
336 <latexrelease> \else
337 <latexrelease> \expandafter\@firstoftwo
338 <latexrelease> \fi}
339 <latexrelease>\def\IfFileLoadedT#1#2{\IfFileLoadedTF{#1}{#2}{}}
340 <latexrelease>\def\IfFileLoadedF #1{\IfFileLoadedTF{#1}{}}
341 <latexrelease>
342 <latexrelease>\EndIncludeInRelease
343 <*2ekernel>

```

(End of definition for \IfPackageLoadedT and others.)

\ProvidesPackage Checks that the current filename is correct, and defines `\ver@filename`.

```

344 </2ekernel>
345 <latexrelease>\IncludeInRelease{2020/10/01}%
346 <latexrelease> {\ProvidesPackage}{Check name with \strcmp}%
347 <*2ekernel | latexrelease>
348 \def\ProvidesPackage#1{%
349 \xdef\@gtempa{#1}%

```

Here `\@currpath` is explicitly added to the file name to report when a package or class is loaded using an explicit path. Loading using a path in the argument is supported but not encouraged.

```

350 \expandtwoargs\expl@str@if@eq@nnTF
351 {\@gtempa}{\@currpath\@currname}{}%
352 \@latex@warning@no@line{You have requested
353 \@cls@pkg\space'\@currpath\@currname',\MessageBreak
354 but the \@cls@pkg\space provides '#1'}%
355 }%

```

```

356 \ifnextchar[\@pr@videpackage{\@pr@videpackage[]}]%
357 \@onlypreamble\ProvidesPackage
358 </2ekernel | latexrelease>
359 <latexrelease>\EndIncludeInRelease
360 %
361 <latexrelease>\IncludeInRelease{0000/00/00}%
362 <latexrelease> {\ProvidesPackage}{Undo: check name with \strcmp}%
363 <latexrelease>\def\ProvidesPackage#1{%
364 <latexrelease> \xdef\@gtempa{#1}%
365 <latexrelease> \ifx\@gtempa\@currname\else
366 <latexrelease> \latex@warning@no@line{You have requested
367 <latexrelease> \@cls@pkg\space'\@currname',\MessageBreak
368 <latexrelease> but the \@cls@pkg\space provides '#1'}%
369 <latexrelease> \fi
370 <latexrelease> \ifnextchar[\@pr@videpackage{\@pr@videpackage[]}]%
371 <latexrelease>\EndIncludeInRelease
372 <*2ekernel>

```

(End of definition for \ProvidesPackage.)

\@pr@videpackage This is the helper command for `\ProvidesPackage`. It tries to be cautious when handling the identification string in case it contains UTF-8 characters.

```

373 </2ekernel>
374 <*2ekernel | latexrelease>
375 <latexrelease>\IncludeInRelease{2020/10/01}%
376 <latexrelease> {\@pr@videpackage}{Allow for package substitution}%
377 \def\@pr@videpackage[#1]{%
378 \expandafter\protected@xdef % <-- protected...
379 \csname ver@\@currname.\@current\endcsname{#1}% Loaded package
380 \expandafter\let
381 \csname ver@\@currpkg@reqd\expandafter\endcsname % Requested package
382 \csname ver@\@currname.\@current\endcsname
383 \ifx\@current\@clsextension
384 \typeout{Document Class: \@gtempa\space#1}%
385 \else
386 \protected@wlog{Package: \@gtempa\space#1}% <--- protected
387 \fi}

```

(End of definition for \@pr@videpackage.)

\protected@wlog This is like plain T_EX's `\wlog` but gracefully handles protected commands.

```

388 \long\def\protected@wlog#1{\begingroup
389 \set@display@protect
390 \immediate \write \m@ne {#1}\endgroup }

```

(End of definition for \protected@wlog.)

```

391 </2ekernel | latexrelease>
392 <latexrelease>\EndIncludeInRelease
393 <latexrelease>\IncludeInRelease{2020/02/02}%
394 <latexrelease> {\@pr@videpackage}{Protection for package info}%
395 <latexrelease>
396 <latexrelease>\def\@pr@videpackage[#1]{%
397 <latexrelease> \expandafter\protected@xdef % <-- protected...
398 <latexrelease> \csname ver@\@currname.\@current\endcsname{#1}%

```

```

399 <latexrelease>\ifx\@current\@clsextension
400 <latexrelease> \typeout{Document Class: \@gtempa\space#1}%
401 <latexrelease> \else
402 <latexrelease> \protected@wlog{Package: \@gtempa\space#1}% <--- protected
403 <latexrelease> \fi}
404 <latexrelease>
405 <latexrelease>\EndIncludeInRelease
406 <latexrelease>\IncludeInRelease{0000/00/00}%
407 <latexrelease> \{\@pr@videpackage}{Protection for package info}%
408 <latexrelease>
409 <latexrelease>\def\@pr@videpackage[#1]{%
410 <latexrelease> \expandafter\xdef\csname ver@\@currname.\@current\endcsname{#1}%
411 <latexrelease> \ifx\@current\@clsextension
412 <latexrelease> \typeout{Document Class: \@gtempa\space#1}%
413 <latexrelease> \else
414 <latexrelease> \wlog{Package: \@gtempa\space#1}%
415 <latexrelease> \fi}
416 <latexrelease>\let\protected@wlog\@undefined
417 <latexrelease>
418 <latexrelease>\EndIncludeInRelease
419 <*2ekernel>
420 \@onlypreamble\@pr@videpackage

```

\ProvidesClass Like **\ProvidesPackage**, but for classes. This needs a dummy **latexrelease** block to copy the definition of **\ProvidesPackage** as it changes across releases.

```

421 </2ekernel>
422 <latexrelease>\IncludeInRelease{0000/00/00}%
423 <latexrelease> {\ProvidesClass}{Track \ProvidesPackage}%
424 <*2ekernel | latexrelease>
425 \let\ProvidesClass\ProvidesPackage
426 \@onlypreamble\ProvidesClass
427 </2ekernel | latexrelease>
428 <latexrelease>\EndIncludeInRelease
429 <*2ekernel>

```

(End of definition for \ProvidesClass.)

\ProvidesFile Like **\ProvidesPackage**, but for arbitrary files. Do not apply **\@onlypreamble** to these, as we may want to label files input during the document.

```

\@providesfile 430 \def\ProvidesFile#1{%
431 \begingroup
432 \catcode'\ 10 %
433 \ifnum \endlinechar<256 %
434 \ifnum \endlinechar>\m@ne
435 \catcode\endlinechar 10 %
436 \fi
437 \fi
438 \@makeother\%
439 \@makeother\&%
440 \kernel@ifnextchar[{\@providesfile{#1}}{\@providesfile{#1}[]}]

```

During initex a special version of `\@providesfile` is used. The real definition is installed right at the end, in `ltxfinal.dtx`.

```
def\@providesfile#1[#2]{%
  \wlog{File: #1 #2}%
  \expandafter\xdef\csname ver@#1\endcsname{#2}%
\endgroup}
```

(End of definition for \ProvidesFile and \@providesfile.)

`\PassOptionsToPackage` If the package has been loaded, we check that it was first loaded with the options.
`\PassOptionsToClass` Otherwise we add the option list to that of the package.

```
441 </2kernel>
442 <latexrelease>\IncludeInRelease{2021/06/01}%
443 <latexrelease>          {\@passOptions}{Raw option lists}%
444 <*2kernel | latexrelease>
445 \def\@passOptions#1#2#3{%
446   \@expl@@@filehook@set@curr@file@@nNN
447     {\@expl@@@filehook@resolve@file@subst@@w #3.#1\@nil}%
448     \reserved@a\reserved@b
449   \@expl@@@filehook@clear@replacement@flag@@
450   \expandafter\protected\xdef\csname opt@\reserved@a\endcsname{%
451     \@ifundefined{opt@\reserved@a}\@empty
452     {\csname opt@\reserved@a\endcsname,}%
453     \zap@space#2 \@empty}%
454   \expandafter\let
455     \csname opt@#3.#1\expandafter\endcsname
456     \csname opt@\reserved@a\endcsname
```

Extend raw option list

```
457   \@ifundefined{@raw@opt@#3.#1}%
458     {\expandafter\gdef\csname @raw@opt@#3.#1\expandafter\endcsname
459       \expandafter{#2}}%
460     {\expandafter\gaddto@macro\csname @raw@opt@#3.#1\expandafter\endcsname
461       \expandafter{\expandafter,#2}}%
462 }
463 </2kernel | latexrelease>
464 <latexrelease>\EndIncludeInRelease
465 <latexrelease>\IncludeInRelease{2020/10/01}{\@passOptions}
466 <latexrelease> {Add file replacement in \@passOptions}%
467 <latexrelease>
468 <latexrelease>\def\@passOptions#1#2#3{%
469 <latexrelease>   \@expl@@@filehook@set@curr@file@@nNN
470 <latexrelease>     {\@expl@@@filehook@resolve@file@subst@@w #3.#1\@nil}%
471 <latexrelease>     \reserved@a\reserved@b
472 <latexrelease>   \@expl@@@filehook@clear@replacement@flag@@
473 <latexrelease>   \expandafter\xdef\csname opt@\reserved@a\endcsname{%
474 <latexrelease>     \@ifundefined{opt@\reserved@a}\@empty
475 <latexrelease>     {\csname opt@\reserved@a\endcsname,}%
476 <latexrelease>     \zap@space#2 \@empty}%
477 <latexrelease>   \expandafter\let
478 <latexrelease>     \csname opt@#3.#1\expandafter\endcsname
479 <latexrelease>     \csname opt@\reserved@a\endcsname}
480 <latexrelease>\EndIncludeInRelease
```



```

481 <latexrelease>\IncludeInRelease{0000/00/00}{\@pass@options}
482 <latexrelease>  {\@pass@options}%
483 <latexrelease>
484 <latexrelease>\def\@pass@options#1#2#3{%
485 <latexrelease>  \expandafter\xdef\csname opt@#3.#1\endcsname{%
486 <latexrelease>    \ifundefined{opt@#3.#1}\@empty
487 <latexrelease>    {\csname opt@#3.#1\endcsname,}%
488 <latexrelease>    \zap@space#2 \@empty}}
489 <latexrelease>\EndIncludeInRelease
490 <*2ekernel>

491 \@onlypreamble\@pass@options

492 \def\PassOptionsToPackage{\@pass@options\@pkgextension}
493 \def\PassOptionsToClass{\@pass@options\@clsextension}
494 \@onlypreamble\PassOptionsToPackage
495 \@onlypreamble\PassOptionsToClass

```

(End of definition for \PassOptionsToPackage and \PassOptionsToClass.)

\DeclareOption Adds an option as a `\ds@` command, or the default `\default@ds` command.

```

\DeclareOption*
496 \def\DeclareOption{%
497   \let\@fileswith@pti@n@badrequireerror
498   \@ifstar\@defdefault@ds\@declareoption}
499 \long\def\@declareoption#1#2{%
500   \xdef\@declaredoptions{\@declaredoptions,#1}%
501   \toks@{#2}%
502   \expandafter\edef\csname ds@#1\endcsname{\the\toks@}}
503 \long\def\@defdefault@ds#1{%
504   \toks@{#1}%
505   \edef\default@ds{\the\toks@}}
506 \@onlypreamble\DeclareOption
507 \@onlypreamble\@declareoption
508 \@onlypreamble\@defdefault@ds

```

(End of definition for \DeclareOption and \DeclareOption.)*

\OptionNotUsed If we are in a class file, add `\CurrentOption` to the list of unused options. Otherwise, in
\@remove@eq@value a package file do nothing.

```

509 </2ekernel>
510 <latexrelease>\IncludeInRelease{2021/06/01}%
511 <latexrelease>  {\OptionNotUsed}{filter unused option list}%
512 <*2ekernel | latexrelease>
513 \ExplSyntaxOn
514 \def\@remove@eq@value#1=#2\@nil{\tl_trim_spaces:n{#1}}
515 \ExplSyntaxOff

516 \def\OptionNotUsed{%
517   \ifx\@current\@clsextension
518     \xdef\@unusedoptionlist{%
519       \ifx\@unusedoptionlist\@empty\else\@unusedoptionlist,\fi
520       \expandafter\@remove@eq@value\CurrentOption=\@nil}%
521   \fi}
522 </2ekernel | latexrelease>
523 <latexrelease>\EndIncludeInRelease
524 <latexrelease>\IncludeInRelease{0000/00/00}%

```

```

525 <latexrelease>          {\OptionNotUsed}{filter unused option list}%
526 <latexrelease>\let\@remove@eq@value\@undefined
527 <latexrelease>\def\OptionNotUsed{%
528 <latexrelease> \ifx\@current\@clsextension
529 <latexrelease> \xdef\@unusedoptionlist{%
530 <latexrelease> \ifx\@unusedoptionlist\@empty\else\@unusedoptionlist,\fi
531 <latexrelease> \CurrentOption}%
532 <latexrelease> \fi}
533 <latexrelease>\EndIncludeInRelease
534 <*2ekernel>
535 \@onlypreamble\OptionNotUsed

```

(End of definition for \OptionNotUsed and \@remove@eq@value.)

\default@ds The default option code. Set by **\@onefilewithoptions** to either **\OptionNotUsed** for classes, or **\@unknownoptionerror** for packages. This may be reset in either case with **\DeclareOption***.

```
536 % \let\default@ds\OptionNotUsed
```

(End of definition for \default@ds.)

\ProcessOptions **\ProcessOptions*** **\ProcessOptions** calls **\ds@option** for each known package option, then calls **\default@ds** for each option on the local options list. Finally resets all the declared options to **\relax**. The empty option does nothing, this has to be reset on the off chance it's set to **\relax** if an empty element gets into the **\@declaredoptions** list.

The star form is similar but executes options given in the order specified in the document, not the order they are declared in the file. In the case of packages, global options are executed before local ones.

```

537 \def\ProcessOptions{%
538 \let\ds@\@empty
539 \protected@edef\@curroptions{\@optionlist{\@currname.\@current}}%
540 \@ifstar\@xprocessoptions\@processoptions}
541 \@onlypreamble\ProcessOptions
542 \def\@processoptions{%
543 \@for\CurrentOption:=\@declaredoptions\do{%
544 \ifx\CurrentOption\@empty\else
545 \@expandtwoargs\in@{,\CurrentOption,}%
546 ,\ifx\@current\@clsextension\else\@classoptionslist,\fi
547 \@curroptions,}%
548 \ifin@
549 \@use@option
550 \expandafter\let\csname ds@\CurrentOption\endcsname\@empty
551 \fi
552 \fi}%
553 \@processoptions}
554 \@onlypreamble\@processoptions
555 </2ekernel>
556 <latexrelease>\IncludeInRelease{2021/06/01}%
557 <latexrelease> \{\@xprocessoptions}{safer \@xprocessoptions}%
558 <*2ekernel | latexrelease>
559 \def\@xprocessoptions{%

```

```

560 \ifx\@current\@clsextension\else
561 \ifx\@classoptionslist\relax\else
562 \@for\CurrentOption:=\@classoptionslist\do{%
563 \ifx\CurrentOption\@empty\else
564 \@ifundefined{ds@\detokenize\expandafter{\CurrentOption}}{ }{%
565 \@use@option
566 \expandafter\let\csname ds@\CurrentOption\endcsname\@empty
567 }%
568 \fi}%
569 \fi
570 \fi
571 \@process@ptions}
572 </2ekernel | latexrelease>
573 <latexrelease>\EndIncludeInRelease
574 <latexrelease>\IncludeInRelease{0000/00/00}%
575 <latexrelease> \@xprocess@ptions}{safer @xprocess@ptions}%
576 <latexrelease>\let\@remove@eq@value\@undefined

577 <latexrelease>\def\@xprocess@ptions{%
578 <latexrelease> \ifx\@current\@clsextension\else
579 <latexrelease> \@for\CurrentOption:=\@classoptionslist\do{%
580 <latexrelease> \ifx\CurrentOption\@empty\else
581 <latexrelease> \@expandtwoargs\in@{,\CurrentOption,}{,\@declaredoptions,}%
582 <latexrelease> \ifin@
583 <latexrelease> \@use@option
584 <latexrelease> \expandafter\let\csname ds@\CurrentOption\endcsname\@empty
585 <latexrelease> \fi
586 <latexrelease> \fi}%
587 <latexrelease> \fi
588 <latexrelease> \@process@ptions}
589 <latexrelease>\EndIncludeInRelease
590 <*2ekernel>

591 \@onlypreamble\@xprocess@ptions

The common part of \ProcessOptions and \ProcessOptions*.
592 </2ekernel>
593 <*2ekernel | latexrelease>
594 <latexrelease>\IncludeInRelease{2020/10/01}%
595 <latexrelease> \@process@ptions}{Unused options issue}%
596 \def\@process@ptions{%
597 \@for\CurrentOption:=\@curoptions\do{%
598 \@ifundefined{ds@\detokenize\expandafter{\CurrentOption}}{
599 \@use@option
600 \default@ds}%

There should not be any non-empty definition of \CurrentOption at this point, as all the
declared options were executed earlier. This is for compatibility with 2.09 styles which use
\def\ds@... directly, and so have options which do not appear in \@declaredoptions.
601 \@use@option}%

Clear all the definitions for option code. First set all the declared options to \relax,
then reset the ‘default’ and ‘empty’ options. and the lst of declared options.
602 \@for\CurrentOption:=\@declaredoptions\do{%
603 \expandafter\let\csname ds@\CurrentOption\endcsname\relax}%

```

```

604 \let\CurrentOption\@empty
605 \let\@fileswith@pti@ns\@fileswith@pti@ns
606 \AtEndOfPackage{\expandafter\let
607                 \csname unprocessedoptions-\@currname.\@current\endcsname
608                 \relax}}
609 \onlypreamble\@process@pti@ns
610 </2ekernel | latexrelease>
611 <latexrelease>\EndIncludeInRelease
612 <latexrelease>\IncludeInRelease{0000/00/00}%
613 <latexrelease>           {\@process@pti@ns}{Unused options issue}%
614 <latexrelease>
615 <latexrelease>\def\@process@pti@ns{%
616 <latexrelease>  \for\CurrentOption:=\@curroptions\do{%
617 <latexrelease>    \@ifundefined{ds@\CurrentOption}%
618 <latexrelease>      {\use@option
619 <latexrelease>        \default@ds}%
620 <latexrelease>      \@use@option}%
621 <latexrelease>  \for\CurrentOption:=\@declaredoptions\do{%
622 <latexrelease>    \expandafter\let\csname ds@\CurrentOption\endcsname\relax}%
623 <latexrelease>  \let\CurrentOption\@empty
624 <latexrelease>  \let\@fileswith@pti@ns\@fileswith@pti@ns
625 <latexrelease>  \AtEndOfPackage{\let\@unprocessedoptions\relax}}
626 <latexrelease>\EndIncludeInRelease
627 <*2ekernel>

```

(End of definition for \ProcessOptions and \ProcessOptions.)*

\@options \@options is a synonym for \ProcessOptions* for upward compatibility with L^AT_EX 2.09 style files.

```

628 \def\@options{\ProcessOptions*}
629 \onlypreamble\@options

```

(End of definition for \@options.)

\@use@option Execute the code for the current option.

```

630 </2ekernel>
631 <latexrelease>\IncludeInRelease{2021/06/01}%
632 <latexrelease>           {\@use@option}{filter unused option list}%
633 <*2ekernel | latexrelease>
634 \def\@use@option{%
635   \@expandtwoargs\@removeelement
636     {\expandafter\@remove@eq@value\CurrentOption=\@nil}%
637   \@unusedoptionlist\@unusedoptionlist
638   \csname ds@\detokenize\expandafter{\CurrentOption}\endcsname}
639 </2ekernel | latexrelease>
640 <latexrelease>\EndIncludeInRelease
641 <latexrelease>\IncludeInRelease{0000/00/00}%
642 <latexrelease>           {\@use@option}{filter unused option list}%
643 <latexrelease>\def\@use@option{%
644 <latexrelease>  \@expandtwoargs\@removeelement\CurrentOption
645 <latexrelease>  \@unusedoptionlist\@unusedoptionlist
646 <latexrelease>  \csname ds@\CurrentOption\endcsname}
647 <latexrelease>\EndIncludeInRelease
648 <*2ekernel>

```

```
649 \@onlypreamble\@use@option
```

(End of definition for \@use@option.)

`\ExecuteOptions` `\ExecuteOptions{<option-list>}` executes the code declared for each option.

```
650 </2ekernel>
651 <latexrelease>\IncludeInRelease{2017/01/01}%
652 <latexrelease>          {\ExecuteOptions}{Spaces in \ExecuteOptions}%
653 <*2ekernel | latexrelease>
654 \def\ExecuteOptions#1{%
```

Use \@fortmp here as it is anyway cleared during \@for loop so does not change any existing names.

```
655   \edef\@fortmp{\zap@space#1 \@empty}%
656   \def\reserved@a##1\@nil{%
657     \for\CurrentOption:=\@fortmp\do
658       {\csname ds@\CurrentOption\endcsname}%
659     \edef\CurrentOption{##1}}%
660   \expandafter\reserved@a\CurrentOption\@nil}
661 </2ekernel | latexrelease>
662 <latexrelease>\EndIncludeInRelease
663 <latexrelease>\IncludeInRelease{0000/00/00}%
664 <latexrelease>          {\ExecuteOptions}{Spaces in \ExecuteOptions}%
665 <latexrelease>\def\ExecuteOptions#1{%
666 <latexrelease>   \def\reserved@a##1\@nil{%
667 <latexrelease>     \for\CurrentOption:=#1\do
668 <latexrelease>       {\csname ds@\CurrentOption\endcsname}%
669 <latexrelease>     \edef\CurrentOption{##1}}%
670 <latexrelease>   \expandafter\reserved@a\CurrentOption\@nil}
671 <latexrelease>\EndIncludeInRelease
672 <*2ekernel>
```

```
673 \@onlypreamble\ExecuteOptions
```

(End of definition for \ExecuteOptions.)

The top-level commands, which just set some parameters then call the internal command, \@fileswithoptions.

`\documentclass` The main new-style class declaration.

```
674 \def\documentclass{%
675   \let\documentclass@twoclasseserror
676   \if@compatibility\else\let\usepackage\RequirePackage\fi
677   \@fileswithoptions\@clsextension}
678 \@onlypreamble\documentclass
```

(End of definition for \documentclass.)

`\documentstyle` 2.09 style class ‘style’ declaration.

```
679 \def\documentstyle{%
680   \makeatletter\input{latex209.def}\makeatother
681   \documentclass}
682 \@onlypreamble\documentstyle
```

(End of definition for \documentstyle.)

`\RequirePackage` Load package if not already loaded.

```
683 \def\RequirePackage{%
684   \@fileswithoptions\@pkgextension}
685 \@onlypreamble\RequirePackage
```

(End of definition for \RequirePackage.)

`\LoadClass` Load class.

```
686 \def\LoadClass{%
687   \ifx\@current\@pkgextension
688     \@latex@error
689     {\noexpand\LoadClass in package file}%
690     {You may only use \noexpand\LoadClass in a class file.}%
691   \fi
692   \@fileswithoptions\@clsextension}
693 \@onlypreamble\LoadClass
```

(End of definition for \LoadClass.)

`\@loadwithoptions` Pass the current option list on to a class or package. #1 is `\@cls-or-pkgextension`, #2 is `\RequirePackage` or `\LoadClass`, #3 is the class or package to be loaded.

```
694 </2ekernel>
695 <latexrelease>\IncludeInRelease{2021/06/01}%
696 <latexrelease>          {\@loadwithoptions}{Raw option lists load with options}%
697 <*2ekernel | latexrelease>
698 \def\@loadwithoptions#1#2#3{%
699   \expandafter\let\csname opt@#3.#1\expandafter\endcsname
700     \csname opt@\@currname.\@current\endcsname
701   \expandafter\let\csname @raw@opt@#3.#1\expandafter\endcsname
702     \csname @raw@opt@\@currname.\@current\endcsname
703     #2{#3}}
704 </2ekernel | latexrelease>
705 <latexrelease>\EndIncludeInRelease
706 <latexrelease>\IncludeInRelease{0000/00/00}
707 <latexrelease>          {\@loadwithoptions}{Raw option lists load with options}%
708 <latexrelease>\def\@loadwithoptions#1#2#3{%
709 <latexrelease>   \expandafter\let\csname opt@#3.#1\expandafter\endcsname
710 <latexrelease>     \csname opt@\@currname.\@current\endcsname
711 <latexrelease>     #2{#3}}
712 <latexrelease>\EndIncludeInRelease
713 <*2ekernel>
714 \@onlypreamble\@loadwithoptions
```

(End of definition for \@loadwithoptions.)

`\LoadClassWithOptions` Load class ‘#1’ with the current option list.

```
715 \def\LoadClassWithOptions{%
716   \@loadwithoptions\@clsextension\LoadClass}
717 \@onlypreamble\LoadClassWithOptions
```

(End of definition for \LoadClassWithOptions.)

`\RequirePackageWithOptions` Load package ‘#1’ with the current option list.

```

718 </2ekernel>
719 <*2ekernel | latexrelease>
720 <latexrelease>\IncludeInRelease{2020/10/01}%
721 <latexrelease>          {\RequirePackageWithOptions}{Unused options issue}%
722 \def\RequirePackageWithOptions{%

```

The resetting of the unprocessed options is now done on a par package basis.

```

723   \AtEndOfPackage{\expandafter\let
724                   \csname unprocessedoptions-\@currname.\@current\endcsname
725                   \relax}%
726   \@loadwithoptions\@pkgextension\RequirePackage}
727 \@onlypreamble\RequirePackageWithOptions
728 </2ekernel | latexrelease>
729 <latexrelease>\EndIncludeInRelease

730 <latexrelease>\IncludeInRelease{0000/00/00}%
731 <latexrelease>          {\RequirePackageWithOptions}{Unused options issue}%
732 <latexrelease>
733 <latexrelease>\def\RequirePackageWithOptions{%
734 <latexrelease>   \AtEndOfPackage{\let\@unprocessedoptions\relax}%
735 <latexrelease>   \@loadwithoptions\@pkgextension\RequirePackage}
736 <latexrelease>\EndIncludeInRelease
737 <*2ekernel>

```

(End of definition for \RequirePackageWithOptions.)

`\usepackage` To begin with, `\usepackage` produces an error. This is reset by `\documentclass`.

```

738 \def\usepackage#1{%
739   \@latex@error
740   {\noexpand \usepackage before \string\documentclass}%
741   {\noexpand \usepackage may only appear in the document
742    preamble, i.e.,\MessageBreak
743    between \noexpand\documentclass and
744    \string\begin{document}.}%
745   \@gobble}
746 \@onlypreamble\usepackage

```

(End of definition for \usepackage.)

`\NeedsTeXFormat` Check that the document is running on the correct system.

```

747 \def\NeedsTeXFormat#1{%
748   \def\reserved@a{#1}%
749   \ifx\reserved@a\fmtname
750     \expandafter\@needsformat
751   \else
752     \@latex@error{This file needs format ‘\reserved@a’%
753     \MessageBreak but this is ‘\fmtname’}%
754     The current input file will not be processed
755     further,\MessageBreak
756     because it was written for some other flavor of
757     TeX.\MessageBreak\@ehd}%

```

If the file is not meant to be processed by L^AT_EX 2_ε we stop inputting it, but we do not end the run. We just end inputting the current file.

```

758     \endinput \fi}
759 \onlypreamble\NeedsTeXFormat

760 \def\@needsformat{%
761   \@ifnextchar [%]
762     \@needsformat
763   {}}
764 \onlypreamble\@needsformat

765 \def\@needsformat[#1]{%
766   \ifl@t@r\fmtversion{#1}-{}%
767   {\@latex@warning@no@line
768     {You have requested release ‘#1’ of LaTeX,\MessageBreak
769     but only release ‘\fmtversion’ is available}}
770 \onlypreamble\@needsformat

```

(End of definition for \NeedsTeXFormat.)

`\zap@space` `\zap@space foo(space)\@empty` removes all spaces from `foo` that are not protected by `{ }` groups.

```

771 \def\zap@space#1 #2{%
772   #1%
773   \ifx#2\@empty\else\expandafter\zap@space\fi
774   #2}

```

(End of definition for \zap@space.)

`\@fileswithoptions` The common part of `\documentclass` and `\usepackage`.

```

775 </2ekernel)
776 <latexrelease>\IncludeInRelease{2024/06/01}%
777 <latexrelease>           {\@fileswithoptions}{Check Group}%
778 <*2ekernel | latexrelease)
779 \def\@fileswithoptions#1{%
780   \ifnum\currentgrouplevel>\z@
781     \@latex@error
782     {Loading a class or package in a group}%
783     {Classes and packages should only be loaded at the top level}%
784   \fi
785   \@ifnextchar [%]
786     {\@fileswithoptions#1}%
787     {\@fileswithoptions#1[]}}
788 </2ekernel | latexrelease)
789 <latexrelease>\EndIncludeInRelease
790 <latexrelease>\IncludeInRelease{0000/00/00}%
791 <latexrelease>           {\@fileswithoptions}{Check Group}%
792 <latexrelease>\def\@fileswithoptions#1{%
793 <latexrelease>   \@ifnextchar [%]
794 <latexrelease>     {\@fileswithoptions#1}%
795 <latexrelease>     {\@fileswithoptions#1[]}}
796 <latexrelease>\EndIncludeInRelease
797 <*2ekernel)
798 \onlypreamble\@fileswithoptions

```



```

799 \def\@fileswithoptions#1[#2]#3{%
800   \@ifnextchar[%
801     {\@fileswithoptions#1[#2]#3}%
802     {\@fileswithoptions#1[#2]#3[]}}
803 \@onlypreamble\@fileswithoptions

```

Then we do some work.

First of all, we define the global variables. Then we look to see if the file has already been loaded. If it has, we check that it was first loaded with at least the current options. If it has not, we add the current options to the package options, set the default version to be 0000/00/00, and load the file if we can find it. Then we check the version number.

Finally, we restore the old file name, reset the default option, and we set the catcode of @.

For classes, we can immediately process the file. For other types, #2 could be a comma separated list, so loop through, processing each one separately.

```

804 </2ekernel)
805 <latexrelease>\IncludeInRelease{2020/10/01}%
806 <latexrelease>   {\@fileswithoptions}{\ifx tests in \@fileswithoptions}%
807 <*2ekernel | latexrelease>
808 \def\@fileswithoptions#1[#2]#3[#4]{%
809   \ifx#1\@clsextension
810     \ifx\@classoptionslist\relax
811       \protected@xdef\@classoptionslist{\zap@space#2 \@empty}%

```

Save raw class list.

```

812   \gdef\@raw@classoptionslist{#2}%
813   \def\reserved@a{%
814     \@onefilewithoptions#3[#{#2}][#{#4}]#1%
815     \@documentclasshook}%
816   \else
817     \def\reserved@a{%
818       \@onefilewithoptions#3[#{#2}][#{#4}]#1%
819     \fi
820   \else

```

build up a list of calls to \@onefilewithoptions (one for each package) without thrashing the parameter stack.

```

821   \def\reserved@b##1,{%

```

If #1 is \@nnil we have reached the end of the list (older version used \@nil here but \@nil is undefined so \ifx equal to all undefined commands)

```

822     \ifx\@nnil##1\relax\else

```

If \ifx\@nnil##1\@nnil is true then #1 is (presumably) empty (Older code used \relax which is slightly easier to get into #1 by mistake, which would spoil this test.)

```

823     \ifx\@nnil##1\@nnil\else
824       \noexpand\@onefilewithoptions##1[{\unexpanded{#2}}][#{#4}]%
825       \noexpand\@pkgextension
826     \fi
827     \expandafter\reserved@b
828   \fi}%
829   \edef\reserved@a{\zap@space#3 \@empty}%
830   \edef\reserved@a{\expandafter\reserved@b\reserved@a,\@nnil,}%

```

```

831 \fi
832 \reserved@a}
833 </2kernel | latexrelease>
834 <latexrelease>\EndIncludeInRelease
835 <latexrelease>\IncludeInRelease{2017/01/01}%
836 <latexrelease> \{\@fileswith@pti@ns}{ifx tests in \@fileswith@pti@ns}%
837 <latexrelease>\def\@fileswith@pti@ns#1[#2]#3[#4]{%
838 <latexrelease> \ifx#1\@clsextension
839 <latexrelease> \ifx\@classoptionslist\relax
840 <latexrelease> \xdef\@classoptionslist{\zap@space#2 \@empty}%
841 <latexrelease> \def\reserved@a{%
842 <latexrelease> \@onefilewithoptions#3[#{#2}][#{#4}]#1%
843 <latexrelease> \@documentclasshook}%
844 <latexrelease> \else
845 <latexrelease> \def\reserved@a{%
846 <latexrelease> \@onefilewithoptions#3[#{#2}][#{#4}]#1}%
847 <latexrelease> \fi
848 <latexrelease> \else
849 <latexrelease> \def\reserved@b##1,{%
850 <latexrelease> \ifx\@nnil##1\relax\else
851 <latexrelease> \ifx\@nnil##1\@nnil\else
852 <latexrelease> \noexpand\@onefilewithoptions##1[#{#2}][#{#4}]%
853 <latexrelease> \noexpand\@pkgextension
854 <latexrelease> \fi
855 <latexrelease> \expandafter\reserved@b
856 <latexrelease> \fi}%
857 <latexrelease> \edef\reserved@a{\zap@space#3 \@empty}%
858 <latexrelease> \edef\reserved@a{\expandafter\reserved@b\reserved@a,\@nnil,}%
859 <latexrelease> \fi
860 <latexrelease> \reserved@a}

861 <latexrelease>\EndIncludeInRelease
862 <latexrelease>\IncludeInRelease{0000/00/00}%
863 <latexrelease> \{\@fileswith@pti@ns}{ifx tests in \@fileswith@pti@ns}%
864 <latexrelease>\def\@fileswith@pti@ns#1[#2]#3[#4]{%
865 <latexrelease> \ifx#1\@clsextension
866 <latexrelease> \ifx\@classoptionslist\relax
867 <latexrelease> \xdef\@classoptionslist{\zap@space#2 \@empty}%
868 <latexrelease> \def\reserved@a{%
869 <latexrelease> \@onefilewithoptions#3[#{#2}][#{#4}]#1%
870 <latexrelease> \@documentclasshook}%
871 <latexrelease> \else
872 <latexrelease> \def\reserved@a{%
873 <latexrelease> \@onefilewithoptions#3[#{#2}][#{#4}]#1}%
874 <latexrelease> \fi
875 <latexrelease> \else
876 <latexrelease> \def\reserved@b##1,{%
877 <latexrelease> \ifx\@nil##1\relax\else
878 <latexrelease> \ifx\relax##1\relax\else
879 <latexrelease> \noexpand\@onefilewithoptions##1[#{#2}][#{#4}]%
880 <latexrelease> \noexpand\@pkgextension
881 <latexrelease> \fi
882 <latexrelease> \expandafter\reserved@b
883 <latexrelease> \fi}%
884 <latexrelease> \edef\reserved@a{\zap@space#3 \@empty}%

```

```

885 <latexrelease> \edef\reserved@a{%
886 <latexrelease> \expandafter\reserved@b\reserved@a,\@nil,}%
887 <latexrelease> \fi
888 <latexrelease> \reserved@a}
889 <latexrelease>\EndIncludeInRelease
890 <*2ekernel)
891 \@onlypreamble\@fileswith@ptions

```

This macro is used when loading packages or classes.

`\load@onefilewithoptions` Have the main argument as #1, so we only need one `\expandafter` above.

```

892 </2ekernel)
893 <*2ekernel | latexrelease)
894 <latexrelease>\IncludeInRelease{2020/10/01}%
895 <latexrelease> {\@onefilewithoptions}{Hooks and unused options issue}%

```

Here this macro is called `\@onefilewithoptions`, but further ahead in this file it is renamed to `\load@onefilewithoptions`, and `\@onefilewithoptions` becomes a wrapper around this, used for bookkeeping when rolling back. Therefore, when in `latexrelease`, we need to define `\load@onefilewithoptions` instead, thus the extra guarded `\def` line below:

```

896 <*2ekernel)
897 \def\@onefilewithoptions#1[#2][#3]#4{%
898 </2ekernel)
899 <latexrelease>\def\load@onefilewithoptions#1[#2][#3]#4{%

```

We have to sanitise file names, so that something like

```

\usepackage{some/local/path/array}
\usepackage{array}

```

won't load `array.sty` twice. It is remotely possible that those are two different files, but as a matter of principles, we will consider that the base file name uniquely identifies a package, regardless of where it lives. This assumption already holds for file hooks, for example, which address the hook to a file by its base name only.

We'll use `\@expl@@@filehook@set@curr@file@@nNN{#1.#4}\reserved@a\reserved@b` to parse the file name and return the `<path>` and `<base+ext>` in separate token lists. Further ahead, most operations use `\@currname` which doesn't have a path attached to it; only few actions prepend `\@currpath` to `\@currname` (namely loading, as we have to respect the given path).

A file substitution isn't followed just yet because at this point we are parsing user input, so the file is still what the user asked for, and not the file actually loaded.

```

900 \@expl@@@filehook@set@curr@file@@nNN{#1.#4}\reserved@a\reserved@b
901 \edef\reserved@c{\def\noexpand\reserved@c####1%
902 \detokenize\expandafter{\expanded{.#4}}}%
903 \noexpand\@nil{\def\noexpand\reserved@a{####1}}\reserved@c
904 \expandafter\reserved@c\reserved@a\@nil
905 \@pushfilename
906 \xdef\@currname{\string@makeletter\reserved@a}%
907 \xdef\@currpath{\ifx\reserved@b\@empty\else\reserved@b/\fi}%
908 \global\let\@currext#4%

```

The command `\ver@<file>.<ext>` is used to signal that a package is already loaded, either because it is in fact loaded, or because its loading was suppressed. In minimal installations, said package may not exist but still have its loading suppressed with `\ver@<file>.<ext>`, so before checking if the file exists we have to check that we do need

to load it with `\@ifl@aded`. If we don't, then there's no point in checking for a typo or load-disabling.

```
909 \@ifl@aded\@current\@currname
```

In the current preferred approach, a key family name will exist for processing using `ltkeys`. In that case, we replace the previous package options with the new ones, then call the key handler. Otherwise, we use the more classical clash handler.

```
910 {%
911 \@ifundefined{opt@handler@\@currname.\@current}
912 {\@onefilewithoptions@clashchk{#2}}
913 {%
914 \expandafter\protected@edef
915 \csname opt@\@currname.\@current\endcsname
916 {\zap@space#2 \@empty}%
917 \expandafter\def
918 \csname @raw@opt@\@currname.\@current\expandafter\endcsname
919 \expandafter{#2}%
920 \@nameuse{opt@handler@\@currname.\@current}%
921 }%
922 }%
923 {\makeatletter
```

The next line seems to be necessary for 2.09 compatibility (the way the code is written there) This seems questionable and should be look at as in 2e it is definitely unnecessary at this point!

```
924 \@reset@options
```

First we take the `<name>` and `<ext>` given in the argument and check if the file exists, and issue an error otherwise asking for a correction with `\@missingfileerror`. For checking if the file exists we use `\@currpath` (usually empty) before `\@currname`.

```
925 \IfFileExists{\@currpath\@currname.\@current}{}%
926 {\@missing@onefilewithoptions{#2}}%
```

If `\@currname` is empty (the user replied to the “Enter file name” prompt with `<RETURN>`), so stop here (do `\@popfilename` to pop the item just added above).

This `\@gobble` omits the date check at the end.

```
927 \ifx\@currname\@empty
928 \expandafter\@gobble
929 \else
```

If the file exists, check if it was load-prevented, and otherwise do the bookkeeping with `\@filehook@file@push` then call `\set@curr@file` to set `\@curr@file` (and do any required substitution), then actually load the class/package with `\load@onefile@withoptions`. `\set@curr@file` also needs the file path.

```
930 \@disable@packageload@do{\@currname.\@current}%
931 {\@expl@@@filehook@file@push@@
932 \set@curr@file{\@currpath\@currname.\@current}%
933 \@filehook@set@CurrentFile
```

The `\set@curr@file` line above might have replaced the file, so `\@currname` and `\@current` may no longer hold the actual package being loaded, so in that case we need to update these two token lists (`\@curr@file` holds the file name after replacement, so we parse that).

The requested file is saved in `\@currpkg@reqd` to be used in `\InputIfFileExists` later: if the updated `\@currname` and `\@currentext` are used we lose track of the substitution, so `\CurrentFile` and `\CurrentFileUsed` will be (incorrectly) the same.

```
934     \expandafter\@swaptwoargs\expandafter
935     {\expandafter{\@currpkg@reqd}}%
936     {% <
```

`\@currpkg@reqd` doesn't take a path because it is used later to assign `\opt@...` and `\ver@...`

```
937     \edef\@currpkg@reqd{\@currname.\@currentext}%
938     \ifx\CurrentFile\CurrentFileUsed
939     \else
940     \filename@parse\@curr@file
941     \edef\@currpath{\string@makeletter\filename@area}%
942     \edef\@currname{\string@makeletter\filename@base}%
943     \edef\@currentext{\string@makeletter\filename@ext}%
944     \fi
945     \load@onefile@withoptions{#2}%
946     \def\@currpkg@reqd%{\@currpkg@reqd}
947     }% >
```

Now just clean up and exit.

```
948     \@expl@@@filehook@file@pop@@}%
949     \expandafter\@firstofone
950     \fi}%
```

Except in the case where `\@currname` is empty, the date is checked against the date marked in the package file:

```
951     {\@ifl@ter\@currentext{\@currname}{#3}-}%
952     {\@latex@warning@no@line
953     {You have requested,\@on@line,
954     version\MessageBreak
955     '#3' of \@cls@pkg\space \@currname,\MessageBreak
956     but only version\MessageBreak
957     '\csname ver@\@currname.\@currentext\endcsname'\MessageBreak
958     is available}}}%
959     \ifx\@currentext\@clsextension\let\LoadClass\@twoloadclasserror\fi}%
960     \@popfilename
961     \@reset@options}
```

If the package is already loaded, check that there were no option clashes.

```
\@onefilewithoptions@clashchk
962 \def\@onefilewithoptions@clashchk#1{%
963 \@if@options\@currentext{\@currname}{#1}-}%
964 {\@latex@error
965 {Option clash for \@cls@pkg\space \@currname}%
966 {The package \@currname\space has already been loaded
967 with options:\MessageBreak
968 \space\space[\@optionlist{\@currname.\@currentext}]\MessageBreak
969 There has now been an attempt to load it
970 with options\MessageBreak
971 \space\space[#1]\MessageBreak
972 Adding the global options:\MessageBreak
973 \space\space
```

```

974         \optionlist{\@currname.\@current},#1\MessageBreak
975         to your \noexpand\documentclass declaration may fix this.%
976         \MessageBreak
977         Try typing \space <return> \space to proceed.}}%
978     \@firstofone}

```

```

979 \let\@currpkg@reqd\@empty
980 \onlypreamble\@onefilewithoptions
    The kernel no longer uses \@unprocessedoptions
981 \let\@unprocessedoptions\@undefined

```

Now the action taken when a file is not found. Path must be included here as it eventually leads to a file lookup.

```

\@missing@onefilewithoptions 982 \def\@missing@onefilewithoptions#1{%
983     \@missingfileerror{\@currpath\@currname}\@current
984     \global\let\@currpath\@missingfile@area
985     \global\let\@currname\@missingfile@base
986     \global\let\@current\@missingfile@ext}

```

Now the code that actually does the file loading:

```

\load@onefile@withoptions 987 \def\load@onefile@withoptions#1{%
988     \let\CurrentOption\@empty
989     \@reset@options

```

Grab everything in a macro, so the parameter stack is popped before any processing begins.

```

990     \def\reserved@a{%
991         \@pass@options\@current{#1}{\@currname}%
992     \expandafter\let
993         \csname opt@\@currpkg@reqd\expandafter\endcsname
994         \csname opt@\@currname.\@current\endcsname
995     \expandafter\let
996         \csname @raw@opt@\@currpkg@reqd\expandafter\endcsname
997         \csname @raw@opt@\@currname.\@current\endcsname
998     \global\expandafter
999     \let\csname ver@\@currname.\@current\endcsname\@empty

```

We initialize `\dots-h@k` here and only if we load the file so that it remains undefined otherwise.

```

1000     \expandafter\let\csname\@currname.\@current-h@k\endcsname\@empty

```

When the current extension is `\@pkgextension` we are loading a package otherwise, if it is `\@clsextension`, a class, so depending on that we execute different hooks. If the extension is neither, then it is another type of file without special hooks.

```

1001 %-----
1002     \ifx\@current\@pkgextension
1003         \UseHook{package/before}%
1004         \UseOneTimeHook{package/\@currname/before}%
1005     \else
1006         \ifx\@current\@clsextension
1007             \UseHook{class/before}%
1008             \UseOneTimeHook{class/\@currname/before}%
1009         \fi
1010     \fi

```

Now actually load the file (at this point we are certain it exists, but use `\InputIfFileExists` so that file hooks are executed). `\@currpath` is needed here too.

```

1011 \InputIfFileExists{\@currpath\@currpkg@reqd}{}%
1012   {\@latex@error
1013     {The \@cls@pkg\space\@currpkg@reqd\space failed to load}\@ehd}%
1014 %-----

```

In older versions of the code `\@unprocessedoptions` would generate an error for each specified option in a package unless a `\ProcessOptions` has appeared in the package file.

This has changed in 2020. We now use a separate macro per package to avoid interference in case of nested packages. The whole code for handling this issue (GitHub 22) was provided by Hironobu Yamashita, thanks for that.

```

1015 \expandafter\let\csname unprocessedoptions-\@currname.\@current\endcsname
1016   \@unprocessedoptions
1017 \csname\@currname.\@current-h@k\endcsname
1018 \expandafter\let\csname\@currname.\@current-h@k\endcsname
1019   \@undefined

```

Catch the case where the packages has handled the options and redefined `\@unprocessedoptions` to `\relax` (old interface). In that case no error should be produced.

```

1020 \ifx\@unprocessedoptions\relax
1021   \let\@unprocessedoptions\@undefined

```

Otherwise run the per package set of unused options.

```

1022 \else
1023   \csname unprocessedoptions-\@currname.\@current\endcsname
1024 \fi

```

In either case we drop the macro afterwards as it is no longer needed.

```

1025 \expandafter\let
1026   \csname unprocessedoptions-\@currname.\@current\endcsname
1027   \@undefined

```

And same procedure, James, when we are finished loading, except that the hook order is now reversed.

```

1028 %-----
1029 \ifx\@current\@pkgextension
1030   \UseOneTimeHook{package/\@currname/after}%
1031   \UseHook{package/after}%
1032 \else
1033   \ifx\@current\@clsextension
1034     \UseOneTimeHook{class/\@currname/after}%
1035     \UseHook{class/after}%
1036   \fi
1037 \fi}%
1038 %-----
1039 \@ifl@aded\@current\@currname{}{\reserved@a}

```

Now declare the non-generic package and class hooks used above:

```

1040 \NewHook{package/before}
1041 \NewHook{class/before}
1042 \NewReversedHook{package/after}
1043 \NewReversedHook{class/after}

```

```

1044 </2ekernel | latexrelease>
1045 <latexrelease>\EndIncludeInRelease
1046 <latexrelease>\IncludeInRelease{0000/00/00}%
1047 <latexrelease>      {\@onefilewithoptions}{Hooks and unused options issue}%
1048 <latexrelease>

```

Because of the way \@onefilewithoptions is changed for rollback handling below we have to define \load@onefilewithoptions when rolling back!

```

1049 <latexrelease>\def\load@onefilewithoptions#1[#2][#3]#4{%
1050 <latexrelease>  \@pushfilename
1051 <latexrelease>  \xdef\currname{#1}%
1052 <latexrelease>  \global\let\currentt#4%
1053 <latexrelease>  \let\CurrentOption\@empty
1054 <latexrelease>  \@resetoptions
1055 <latexrelease>  \makeatletter
1056 <latexrelease>  \def\reserved@a{%
1057 <latexrelease>    \@ifl@aded\currentt{#1}%
1058 <latexrelease>    {\@ifoptions\currentt{#1}{#2}{}}%
1059 <latexrelease>    {\@latex@error
1060 <latexrelease>      {Option clash for \@cls@pkg\space #1}%
1061 <latexrelease>      {The package #1 has already been loaded
1062 <latexrelease>        with options:\MessageBreak
1063 <latexrelease>        \space\space[\@optionlist{#1.\@currentt}]\MessageBreak
1064 <latexrelease>        There has now been an attempt to load it
1065 <latexrelease>        with options:\MessageBreak
1066 <latexrelease>        \space\space[#2]\MessageBreak
1067 <latexrelease>        Adding the global options:\MessageBreak
1068 <latexrelease>        \space\space
1069 <latexrelease>        \@optionlist{#1.\@currentt},#2\MessageBreak
1070 <latexrelease>        to your \noexpand\documentclass declaration may fix this.%
1071 <latexrelease>        \MessageBreak
1072 <latexrelease>        Try typing \space <return> \space to proceed.}}}%
1073 <latexrelease>  {\@passoptions\currentt{#2}{#1}%
1074 <latexrelease>    \global\expandafter
1075 <latexrelease>    \let\csname ver@\currname.\@currentt\endcsname\@empty
1076 <latexrelease>    \expandafter\let\csname\currname.\@currentt-h@@k\endcsname\@empty
1077 <latexrelease>    \InputIfFileExists
1078 <latexrelease>      {\currname.\@currentt}%
1079 <latexrelease>      {}%
1080 <latexrelease>      {\@missingfileerror\currname\@currentt}%
1081 <latexrelease>    \let\@unprocessedoptions\@unprocessedoptions
1082 <latexrelease>    \csname\currname.\@currentt-h@@k\endcsname
1083 <latexrelease>    \expandafter\let\csname\currname.\@currentt-h@@k\endcsname
1084 <latexrelease>      \undefined
1085 <latexrelease>    \@unprocessedoptions}%
1086 <latexrelease>    \@ifl@ter\currentt{#1}{#3}{}}%
1087 <latexrelease>    {\@latex@warning@no@line
1088 <latexrelease>      {You have requested,\on@line,
1089 <latexrelease>        version\MessageBreak
1090 <latexrelease>        ‘#3’ of \@cls@pkg\space #1,\MessageBreak
1091 <latexrelease>        but only version\MessageBreak
1092 <latexrelease>        ‘\csname ver@#1.\@currentt\endcsname’\MessageBreak
1093 <latexrelease>        is available}}}%
1094 <latexrelease>    \ifx\currentt\@clsextension\let\LoadClass\@twoloadclasserror\fi

```



```

1095 <latexrelease> \popfilename
1096 <latexrelease> \resetoptions}%
1097 <latexrelease> \reserved@a}
1098 <latexrelease>
1099 <latexrelease>\let \load@onefile@withoptions \undefined
1100 <latexrelease>\let \missing@onefile@withoptions \undefined
1101 <latexrelease>
1102 <latexrelease>\EndIncludeInRelease
1103 <*2kernel>

```

(End of definition for \@fileswithoptions and others.)

\@@fileswith@pti@ns Save the definition (for error checking).

```

1104 \let\@@fileswith@pti@ns\@fileswith@pti@ns
1105 \onlypreamble\@@fileswith@pti@ns

```

(End of definition for \@@fileswith@pti@ns.)

\@reset@ptions Reset the default option, and clear lists of declared options.

```

1106 \def\@reset@ptions{%
1107   \global\ifx\@current\@clsextension
1108     \let\default@ds\OptionNotUsed
1109   \else
1110     \let\default@ds\unknownoptionerror
1111   \fi
1112   \global\let\ds@\@empty
1113   \global\let\@declaredoptions\@empty}
1114 \onlypreamble\@reset@ptions

```

(End of definition for \@reset@ptions.)

4.1 Hooks

Allow code to be saved to be executed at specific later times.

Here we save things in macros. I considered using toks registers (and `\addto@hook` from the NFSS code), but that would require stacking the contents in the case of required packages, so just generate a new macro for each package.

\@begin@documenthook Stuff to appear at the beginning or end of the document.

```

\@end@documenthook 1115 \ifx\@begin@documenthook\undefined
1116   \let\@begin@documenthook\@empty
1117 \fi
1118 \let\@end@documenthook\@empty

```

(End of definition for \@begin@documenthook and \@end@documenthook.)

\@AtEndOfPackage The access functions.

```

\@AtEndOfClass 1119 \def\@AtEndOfPackage{%
\@AtBeginDocument 1120   \expandafter\g@addto@macro\csname\@currname.\@current-h@k\endcsname}
\@AtEndDocument 1121 \let\@AtEndOfClass\@AtEndOfPackage
1122 \onlypreamble\@AtEndOfPackage
1123 \onlypreamble\@AtEndOfClass

```

```

1124 </2ekernel>
1125 <*2ekernel | latexrelease>
1126 <latexrelease>\IncludeInRelease{2020/10/01}%
1127 <latexrelease>                {\AtBeginDocument}{Use hook system}%
1128 \DeclareRobustCommand\AtBeginDocument{\AddToHook{begindocument}}
1129 \DeclareRobustCommand\AtEndDocument  {\AddToHook{enddocument}}
1130 %\DeclareRobustCommand\AtEndDocument {\AddToHook{env/document/end}} % alternative impl
1131 </2ekernel | latexrelease>
1132 <latexrelease>\EndIncludeInRelease
1133 <latexrelease>\IncludeInRelease{0000/00/00}%
1134 <latexrelease>                {\AtBeginDocument}{Use hook system}%
1135 <latexrelease>
1136 <latexrelease>\DeclareRobustCommand\AtBeginDocument{\g@addto@macro\@begindocumenthook}
1137 <latexrelease>\DeclareRobustCommand\AtEndDocument{\g@addto@macro\@enddocumenthook}
1138 <latexrelease>
1139 <latexrelease>\EndIncludeInRelease
1140 <*2ekernel>

```

In its initial implementation (not using the hook system) `\AtBeginDocument` was made `\@onlypreamble` because using it later had no effect whatsoever, thus was most certainly an unintended programming error. With the reimplementaion, using the `begindocument` hook internally, this has changed because adding to a onetime hook after it has already been used simply executes the additional code immediately. We therefore no longer generate an error if it is used inside the document so that `\AddToHook{begindocument}` and `\AtBeginDocument` are truly equivalent (as claimed in the hook documentation).

```
1141 %\@onlypreamble\AtBeginDocument
```

(End of definition for \AtEndOfPackage and others.)

`\@cls@pkg` The current file type.

```

1142 </2ekernel>
1143 <*2ekernel | latexrelease>
1144 <latexrelease>\IncludeInRelease{2024/11/01}%
1145 <latexrelease>                {\@cls@pkg}{Allow for more extensions}%
1146 \def\@cls@pkg{%
1147   \ifx\@current\@clsextension
1148     document class%
1149   \else
1150     \ifx\@current\@pkgextension
1151       package%
1152     \else
1153       file%
1154     \fi
1155   \fi}
1156 </2ekernel | latexrelease>
1157 <latexrelease>\EndIncludeInRelease
1158 <latexrelease>\IncludeInRelease{0000/00/00}%
1159 <latexrelease>                {\@cls@pkg}{Allow for more extensions}%
1160 <latexrelease>
1161 <latexrelease>\def\@cls@pkg{%
1162 <latexrelease>  \ifx\@current\@clsextension
1163 <latexrelease>    document class%

```

```

1164 <latexrelease> \else
1165 <latexrelease> package%
1166 <latexrelease> \fi}
1167 <latexrelease>\EndIncludeInRelease
1168 <*2ekernel>

```

```

1169 \@onlypreamble\@cls@pkg
(End of definition for \@cls@pkg.)

```

\@unknownoptionerror Bad option.

```

1170 \def\@unknownoptionerror{%
1171   \latex@error
1172   {Unknown option ‘\CurrentOption’ for \@cls@pkg\space‘\@currname’}%
1173   {The option ‘\CurrentOption’ was not declared in
1174   \@cls@pkg\space‘\@currname’, perhaps you\MessageBreak
1175   misspelled its name.
1176   Try typing \space <return>
1177   \space to proceed.}}
1178 \@onlypreamble\@unknownoptionerror

```

(End of definition for \@unknownoptionerror.)

\@@unprocessedoptions Declare an error for each option, unless a \ProcessOptions occurred.

```

1179 \def\@@unprocessedoptions{%
1180   \ifx\@current\@pkgextension
1181     \protected@edef\@curroptions{\@optionlist{\@currname.\@current}}%
1182     \@for\CurrentOption:=\@curroptions\do{%
1183       \ifx\CurrentOption\@empty\else\@unknownoptionerror\fi}%
1184   \fi}
1185 \@onlypreamble\@unprocessedoptions
1186 \@onlypreamble\@@unprocessedoptions

```

(End of definition for \@@unprocessedoptions.)

\@badrequireerror \RequirePackage or \LoadClass occurs in the options section.

```

1187 \def\@badrequireerror#1[#2]#3[#4]{%
1188   \latex@error
1189   {\noexpand\RequirePackage or \noexpand\LoadClass
1190   in Options Section}%
1191   {The \@cls@pkg\space ‘\@currname’ is defective.\MessageBreak
1192   It attempts to load ‘#3’ in the options section, i.e.,\MessageBreak
1193   between \noexpand\DeclareOption and \string\ProcessOptions.}}
1194 \@onlypreamble\@badrequireerror

```

(End of definition for \@badrequireerror.)

\@twoloadclasserror Two \LoadClass in a class.

```

1195 \def\@twoloadclasserror{%
1196   \latex@error
1197   {Two \noexpand\LoadClass commands}%
1198   {You may only use one \noexpand\LoadClass in a class file}}
1199 \@onlypreamble\@twoloadclasserror

```

(End of definition for \@twoloadclasserror.)

```

\@twoclasseserror Two \documentclass or \documentstyle.
1200 \def\@twoclasseserror#1#{%
1201   \latex@error
1202     {Two \noexpand\documentclass or \noexpand\documentstyle commands}%
1203     {The document may only declare one class.}\@gobble}
1204 \@onlypreamble\@twoclasseserror

(End of definition for \@twoclasseserror.)

```

4.2 Providing shipment

```

\two@digits Prefix a number less than 10 with '0'.
1205 \def\two@digits#1{\ifnum#1<10 0\fi\number#1}

(End of definition for \two@digits.)

```

```

\filecontents* This environment implements inline files. The star-form does not write extra comments
\endfilecontents into the file.
1206 </2kernel>
1207 <*2kernel | latexrelease>
1208 <latexrelease>\IncludeInRelease{2020/10/01}%
1209 <latexrelease>          {\filecontents*}{Define \q@curr@file directly (gh/220)}%
1210 %

```

We use @tempswa to mean no preamble writing and reuse @filesw to indicate no overwriting:

```

1211 \def\filecontents{\@tempswatrue\@fileswtrue
1212   \ifnextchar[\filecontents@opt\filecontents
1213   }
1214 \@namedef{filecontents*}{\@tempswafalse\@fileswtrue
1215   \ifnextchar[\filecontents@opt\filecontents
1216   }

```

To handle the optional argument we execute for each option the command \filecontents@OPTION if it exist or complain about unknown option.

```

1217 \def\filecontents@opt[#1]{%
1218   \edef\@fortmp{\zap@space#1 \@empty}%
1219   \@for\reserved@a:=\@fortmp\do{%
1220     \ifcsname filecontents@\reserved@a\endcsname
1221       \csname filecontents@\reserved@a\endcsname
1222     \else
1223       \latex@error{Unknown filecontents option \reserved@a}%
1224       {Valid options are force (or overwrite), nosearch, noheader, nowarn}%
1225     \fi}%
1226 \filecontents
1227 }

```

Option force (or overwrite) changes the overwriting switch

```

1228 \let\filecontents@force\@fileswfalse
1229 \let\filecontents@overwrite\@fileswfalse % alternative name

```

and option noheader the preamble switch (which is equivalent to using the star form of the environment).

```

1230 \let\filecontents@noheader\@tempswafalse

```

Option `nosearch` only checks the current directory not the whole \TeX tree for the existence of the file to write.

```
1231 \def\filec@ntents@nosearch{%
1232   \let\filec@ntents@checkdir\@currdir
1233   \def\filec@ntents@where{in current directory}}
```

By default we search the whole tree:

```
1234 \let\filec@ntents@checkdir\@empty
1235 \def\filec@ntents@where{exists on the system}
```

Option `nowarn` does not show any warning on the terminal but still writes it to the `.log`.

```
1236 \def\filec@ntents@nowarn{%
1237   \let\filec@ntents@warning\@latex@note@no@line
1238 }
```

By default we show terminal warnings.

```
1239 \let\filec@ntents@warning\@latex@warning@no@line
```

```
1240 \begingroup%
1241 \@tempcnta=1
1242 \loop
1243   \catcode\@tempcnta=12 %
1244   \advance\@tempcnta\@ne %
1245   \ifnum\@tempcnta<32 %
1246     \repeat %
1247   \catcode'\*=11 %
1248   \catcode'\^^M\active%
1249   \catcode'\^^L\active\let^^L\relax%
1250   \catcode'\^^I\active%
```

```
1251 \gdef\filec@ntents#1{%
1252   \set@curr@file{\filec@ntents@checkdir#1}%
1253   \edef\q@curr@file{"\@curr@file"}}
```

Lua \TeX has more writes (and 18 is safe here).

```
1254 \chardef\reserved@c\ifx\directlua\@undefined 15 \else 127 \fi%
1255 \openin\@inputcheck\q@curr@file \space %
1256 \ifeof\@inputcheck%
1257   \@latex@note@no@line%
1258   {Writing file '\@currdir\@curr@file'}%
1259   \ch@ck7\reserved@c\write\relax%
1260   \immediate\openout\reserved@c\q@curr@file\relax%
1261 \else%
1262   \if@filesw%
1263     \@latex@note@no@line%
1264     {File '\@curr@file' already \filec@ntents@where.\MessageBreak%
1265     Not generating it from this source}%
1266     \let\write\@gobbletwo%
1267     \let\closeout\@gobble%
1268   \else%
```

If we are overwriting, we try to make sure that the user is not by mistake overwriting the input file (`\jobname`). Of course, this only works for input files ending in `.tex`. If a different extension is used there is no way to see that we are overwriting ourselves!

```

1269     \edef\reserved@b{\detokenize\expandafter{\jobname}}%
1270     \ifx\@curr@file\reserved@b%
1271         \@fileswtrue%
1272     \else%
1273         \edef\reserved@b{\reserved@b\detokenize{.tex}}%
1274         \ifx\@curr@file\reserved@b
1275             \@fileswtrue%
1276         \fi%
1277     \fi%

```

We allocate a write channel but we open it only if it is (hopefully) safe. If not opened that means we are going to write on the terminal.

```

1278     \ch@ck7\reserved@c\write\relax%
1279     \if@filesw% % Foul ... trying to overwrite \jobname!
1280     \@latex@error{Trying to overwrite '\jobname.tex'}{You can't %
1281         write to the file you are reading from!\MessageBreak%
1282         Data is written to screen instead.}%
1283     \else%
1284         \filec@ntents@warning%
1285         {Writing or overwriting file '\@curr@file'}%
1286         \immediate\openout\reserved@c@q@curr@file\relax%
1287     \fi%
1288 \fi%
1289 \fi%

```

Closing the `\@inputcheck` is done here to avoid having to do this in each branch.

```

1290 \closein\@inputcheck%
1291 \if@tempswa%

1292     \immediate\write\reserved@c{%
1293         \@percentchar\@percentchar\space%
1294         \expandafter\@gobble\string\LaTeX2e file '\@curr@file'^^J%
1295         \@percentchar\@percentchar\space generated by the %
1296         '\@currenvir' \expandafter\@gobblefour\string\newenvironment^^J%
1297         \@percentchar\@percentchar\space from source '\jobname' on %
1298         \number\year/\two@digits\month/\two@digits\day.^^J%
1299         \@percentchar\@percentchar}%
1300 \fi%
1301 \let\do\@makeother\dospecials%

```

If there are active characters in the upper half (e.g., from `inputenc`) there would be confusion so we render everything harmless.

```

1302     \count@ 128\relax%
1303     \loop%
1304         \catcode\count@ 11\relax%
1305         \advance\count@ \@ne%
1306         \ifnum\count@<\@cclvi%
1307             \repeat%

1308     \edef\E{\@backslashchar end\string{\@currenvir\string}}%
1309     \edef\reserved@b{%
1310         \def\noexpand\reserved@b%

```

```

1311     #####1\E####2\E####3\relax}%
1312 \reserved@b{%
1313     \ifx\relax##3\relax%

There was no \end{filecontents}

1314     \immediate\write\reserved@c{##1}%
1315     \else%

There was a \end{filecontents}, so stop this time.

1316     \edef^^M{\noexpand\end{\@currentvir}}%
1317     \ifx\relax##1\relax%
1318     \else%

Text before the \end, write it with a warning.

1319     \@latex@warning{Writing text '##1' before %
1320     \string\end{\@currentvir}\MessageBreak
1321     as last line of \curr@file}%
1322     \immediate\write\reserved@c{##1}%
1323     \fi%
1324     \ifx\relax##2\relax%
1325     \else%

Text after the \end, ignore it with a warning.

1326     \@latex@warning{%
1327     Ignoring text '##2' after \string\end{\@currentvir}}%
1328     \fi%
1329     \fi%
1330     ^^M}%

1331 \catcode'\^^L\active%
1332 \let\L\@undefined%
1333 \def^^L{\expandafter\ifx\curname L\endcurname\relax\fi ^^J^^J}%
1334 \catcode'\^^I\active%
1335 \let\I\@undefined%
1336 \def^^I{\expandafter\ifx\curname I\endcurname\relax\fi\space}%
1337 \catcode'\^^M\active%
1338 \edef^^M##1^^M{%
1339     \noexpand\reserved@b##1\E\E\relax}}%
1340 \endgroup%

1341 </2kernel | latexrelease>
1342 <latexrelease>\EndIncludeInRelease
1343 <latexrelease>\IncludeInRelease{2019/10/01}%
1344 <latexrelease>    {\filec@ntents}{Spaces in file names + optional arg}%
1345 <latexrelease>
1346 <latexrelease>\def\filecontents{\@tempwatrue\@fileswtrue
1347 <latexrelease>  \@ifnextchar[\filec@ntents@opt\filec@ntents
1348 <latexrelease>}
1349 <latexrelease>\@namedef{filecontents*}{\@tempwafalse\@fileswtrue
1350 <latexrelease>  \@ifnextchar[\filec@ntents@opt\filec@ntents
1351 <latexrelease>}
1352 <latexrelease>\def\filec@ntents@opt[#1]{%
1353 <latexrelease>  \edef\@fortmp{\zap@space#1 \@empty}%
1354 <latexrelease>  \@for\reserved@a:=\@fortmp\do{%
1355 <latexrelease>    \ifcurname filec@ntents@\reserved@a\endcurname
1356 <latexrelease>    \curname filec@ntents@\reserved@a\endcurname

```

```

1357 <latexrelease> \else
1358 <latexrelease> \latex@error{Unknown filecontents option \reserved@a}%
1359 <latexrelease> {Valid options are force (or overwrite), nosearch, noheader}%
1360 <latexrelease> \fi}%
1361 <latexrelease> \filec@ntents
1362 <latexrelease>}
1363 <latexrelease>\let\filec@ntents@force\@fileswfalse
1364 <latexrelease>\let\filec@ntents@overwrite\@fileswfalse % alternative name
1365 <latexrelease>\let\filec@ntents@noheader\@tempwafalse
1366 <latexrelease>\def\filec@ntents@nosearch{%
1367 <latexrelease> \let\filec@ntents@checkdir\@currdir
1368 <latexrelease> \def\filec@ntents@where{in current directory}}
1369 <latexrelease>\let\filec@ntents@checkdir\@empty
1370 <latexrelease>\def\filec@ntents@where{exists on the system}
1371 <latexrelease>\begingroup%
1372 <latexrelease>\@tempc@nta=1
1373 <latexrelease>\loop
1374 <latexrelease> \catcode\@tempc@nta=12 %
1375 <latexrelease> \advance\@tempc@nta\@ne %
1376 <latexrelease>\ifnum\@tempc@nta<32 %
1377 <latexrelease>\repeat %
1378 <latexrelease>\catcode'\*=11 %
1379 <latexrelease>\catcode'\^^M\active%
1380 <latexrelease>\catcode'\^^L\active\let^^L\relax%
1381 <latexrelease>\catcode'\^^I\active%
1382 <latexrelease>\gdef\filec@ntents#1{%
1383 <latexrelease> \set@curr@file{\filec@ntents@checkdir#1}%
1384 <latexrelease> \edef\q@curr@file{\expandafter\quote@name\expandafter{\@curr@file}}%
1385 <latexrelease> \chardef\reserved@c\ifx\directlua\@undefined 15 \else 127 \fi%
1386 <latexrelease> \openin\@inputcheck\q@curr@file \space %
1387 <latexrelease> \ifeof\@inputcheck%
1388 <latexrelease> \latex@warning@no@line%
1389 <latexrelease> {Writing file '\@currdir\@curr@file'}%
1390 <latexrelease> \ch@ck7\reserved@c\write\relax%
1391 <latexrelease> \immediate\openout\reserved@c\q@curr@file\relax%
1392 <latexrelease> \else%
1393 <latexrelease> \if@filesw%
1394 <latexrelease> \latex@warning@no@line%
1395 <latexrelease> {File '\@curr@file' already \filec@ntents@where.\MessageBreak%
1396 <latexrelease> Not generating it from this source}%
1397 <latexrelease> \let\write\@gobbletwo%
1398 <latexrelease> \let\closeout\@gobble%
1399 <latexrelease> \else%
1400 <latexrelease> \edef\reserved@a{#1}%
1401 <latexrelease> \edef\reserved@a{\detokenize\expandafter{\reserved@a}}%
1402 <latexrelease> \edef\reserved@b{\detokenize\expandafter{\jobname}}%
1403 <latexrelease> \ifx\reserved@a\reserved@b%
1404 <latexrelease> \@fileswtrue%
1405 <latexrelease> \else%
1406 <latexrelease> \edef\reserved@b{\reserved@b\detokenize{.tex}}%
1407 <latexrelease> \ifx\reserved@a\reserved@b
1408 <latexrelease> \@fileswtrue%
1409 <latexrelease> \fi%
1410 <latexrelease> \fi%

```



```

1411 <latexrelease> \ch@ck7\reserved@c\write\relax%
1412 <latexrelease> \if@filesw% % Foul ... trying to overwrite \jobname!
1413 <latexrelease> \latex@error{Trying to overwrite '\jobname.tex'}{You can't %
1414 <latexrelease> write to the file you are reading from!\MessageBreak%
1415 <latexrelease> Data is written to screen instead.}%
1416 <latexrelease> \else%
1417 <latexrelease> \latex@warning@no@line%
1418 <latexrelease> {Writing or overwriting file '\currdir\curr@file'}%
1419 <latexrelease> \immediate\openout\reserved@c\curr@file\relax%
1420 <latexrelease> \fi%
1421 <latexrelease> \fi%
1422 <latexrelease> \fi%
1423 <latexrelease> \closein\@inputcheck%
1424 <latexrelease> \if@tempswa%
1425 <latexrelease> \immediate\write\reserved@c{%
1426 <latexrelease> \@percentchar\@percentchar\space%
1427 <latexrelease> \expandafter\@gobble\string\LaTeX2e file '\curr@file'^^J%
1428 <latexrelease> \@percentchar\@percentchar\space generated by the %
1429 <latexrelease> '\@currentvir' \expandafter\@gobblefour\string\newenvironment^^J%
1430 <latexrelease> \@percentchar\@percentchar\space from source '\jobname' on %
1431 <latexrelease> \number\year/\two@digits\month/\two@digits\day.^^J%
1432 <latexrelease> \@percentchar\@percentchar}%
1433 <latexrelease> \fi%
1434 <latexrelease> \let\do\makeother\dospecials%
1435 <latexrelease> \count@ 128\relax%
1436 <latexrelease> \loop%
1437 <latexrelease> \catcode\count@ 11\relax%
1438 <latexrelease> \advance\count@ \@ne%
1439 <latexrelease> \ifnum\count@<\ccclvi%
1440 <latexrelease> \repeat%
1441 <latexrelease> \edef\E{\@backslashchar end\string{\@currentvir\string}}%
1442 <latexrelease> \edef\reserved@b{%
1443 <latexrelease> \def\noexpand\reserved@b%
1444 <latexrelease> #####1\E###2\E###3\relax}%
1445 <latexrelease> \reserved@b{%
1446 <latexrelease> \ifx\relax###3\relax%
1447 <latexrelease> \immediate\write\reserved@c{##1}%
1448 <latexrelease> \else%
1449 <latexrelease> \edef^^M{\noexpand\end{\@currentvir}}%
1450 <latexrelease> \ifx\relax##1\relax%
1451 <latexrelease> \else%
1452 <latexrelease> \latex@warning{Writing text '##1' before %
1453 <latexrelease> \string\end{\@currentvir}\MessageBreak as last line of \curr@file}%
1454 <latexrelease> \immediate\write\reserved@c{##1}%
1455 <latexrelease> \fi%
1456 <latexrelease> \ifx\relax##2\relax%
1457 <latexrelease> \else%
1458 <latexrelease> \latex@warning{%
1459 <latexrelease> Ignoring text '##2' after \string\end{\@currentvir}}%
1460 <latexrelease> \fi%
1461 <latexrelease> \fi%
1462 <latexrelease> ^^M}%
1463 <latexrelease> \catcode'\^^L\active%
1464 <latexrelease> \let\L\@undefined%

```

```

1465 <latexrelease> \def^^L{\expandafter\ifx\csname L\endcsname\relax\fi ^^J^^J}%
1466 <latexrelease> \catcode'\^^I\active%
1467 <latexrelease> \let\I\@undefined%
1468 <latexrelease> \def^^I{\expandafter\ifx\csname I\endcsname\relax\fi\space}%
1469 <latexrelease> \catcode'\^^M\active%
1470 <latexrelease> \edef^^M##1^^M{%
1471 <latexrelease> \noexpand\reserved@b##1\E\E\relax}}%
1472 <latexrelease>\endgroup%
1473 <latexrelease>\EndIncludeInRelease
1474 <latexrelease>\IncludeInRelease{0000/00/00}%
1475 <latexrelease> {\filec@ntents}{Spaces in file names + optional arg}%
1476 <latexrelease>
1477 <latexrelease>\let\filec@ntents@opt \@undefined
1478 <latexrelease>\let\filec@ntents@force \@undefined
1479 <latexrelease>\let\filec@ntents@overwrite \@undefined
1480 <latexrelease>\let\filec@ntents@noheader \@undefined
1481 <latexrelease>\let\filec@ntents@nosearch \@undefined
1482 <latexrelease>\let\filec@ntents@checkdir \@undefined
1483 <latexrelease>\let\filec@ntents@where \@undefined
1484 <latexrelease>
1485 <latexrelease>\begingroup%
1486 <latexrelease>\@tempcnta=1
1487 <latexrelease>\loop
1488 <latexrelease> \catcode\@tempcnta=12 %
1489 <latexrelease> \advance\@tempcnta\@ne %
1490 <latexrelease>\ifnum\@tempcnta<32 %
1491 <latexrelease>\repeat %
1492 <latexrelease>\catcode'\*=11 %
1493 <latexrelease>\catcode'\^^M\active%
1494 <latexrelease>\catcode'\^^L\active\let^^L\relax%
1495 <latexrelease>\catcode'\^^I\active%
1496 <latexrelease>
1497 <latexrelease>\gdef\filec@ntents#1{%
1498 <latexrelease> \openin\@inputcheck#1 %
1499 <latexrelease> \ifeof\@inputcheck%
1500 <latexrelease> \@latex@warning@no@line%
1501 <latexrelease> {Writing file '\currdir#1'}%
1502 <latexrelease> \chardef\reserved@c15 %
1503 <latexrelease> \ch@ck7\reserved@c\write%
1504 <latexrelease> \immediate\openout\reserved@c#1\relax%
1505 <latexrelease> \else%
1506 <latexrelease> \closein\@inputcheck%
1507 <latexrelease> \@latex@warning@no@line%
1508 <latexrelease> {File '#1' already exists on the system.\MessageBreak%
1509 <latexrelease> Not generating it from this source}%
1510 <latexrelease> \let\write\@gobbletwo%
1511 <latexrelease> \let\closeout\@gobble%
1512 <latexrelease> \fi%
1513 <latexrelease> \if@tempswa%
1514 <latexrelease> \immediate\write\reserved@c{%
1515 <latexrelease> \@percentchar\@percentchar\space%
1516 <latexrelease> \expandafter\@gobble\string\LaTeX2e file '#1'^^J%
1517 <latexrelease> \@percentchar\@percentchar\space generated by the %
1518 <latexrelease> '\@currenvir' \expandafter\@gobblefour\string\newenvironment^^J%

```

```

1519 <latexrelease> \@percentchar\@percentchar\space from source ‘\jobname’ on %
1520 <latexrelease> \number\year/\two@digits\month/\two@digits\day.^^J%
1521 <latexrelease> \@percentchar\@percentchar}%
1522 <latexrelease> \fi%
1523 <latexrelease> \let\do\@makeother\dospecials%
1524 <latexrelease> \count@ 128\relax%
1525 <latexrelease> \loop%
1526 <latexrelease> \catcode\count@ 11\relax%
1527 <latexrelease> \advance\count@ \one%
1528 <latexrelease> \ifnum\count@<\ccclvi%
1529 <latexrelease> \repeat%
1530 <latexrelease> \edef\E{\@backslashchar end\string{\@currentvir\string}}%
1531 <latexrelease> \edef\reserved@b{%
1532 <latexrelease> \def\noexpand\reserved@b%
1533 <latexrelease> #####1\E####2\E####3\relax}%
1534 <latexrelease> \reserved@b{%
1535 <latexrelease> \ifx\relax##3\relax%
1536 <latexrelease> \immediate\write\reserved@c{##1}%
1537 <latexrelease> \else%
1538 <latexrelease> \edef^^M{\noexpand\end{\@currentvir}}%
1539 <latexrelease> \ifx\relax##1\relax%
1540 <latexrelease> \else%
1541 <latexrelease> \@latex@warning{Writing text ‘##1’ before %
1542 <latexrelease> \string\end{\@currentvir}\MessageBreak as last line of #1}%
1543 <latexrelease> \immediate\write\reserved@c{##1}%
1544 <latexrelease> \fi%
1545 <latexrelease> \ifx\relax##2\relax%
1546 <latexrelease> \else%
1547 <latexrelease> \@latex@warning{%
1548 <latexrelease> Ignoring text ‘##2’ after \string\end{\@currentvir}}%
1549 <latexrelease> \fi%
1550 <latexrelease> \fi%
1551 <latexrelease> ^^M}%
1552 <latexrelease>
1553 <latexrelease> \catcode‘^^L\active%
1554 <latexrelease> \let\L\@undefined%
1555 <latexrelease> \def^^L{\expandafter\ifx\csname L\endcsname\relax\fi ^^J^^J}%
1556 <latexrelease> \catcode‘^^I\active%
1557 <latexrelease> \let\I\@undefined%
1558 <latexrelease> \def^^I{\expandafter\ifx\csname I\endcsname\relax\fi\space}%
1559 <latexrelease> \catcode‘^^M\active%
1560 <latexrelease> \edef^^M##1^^M{%
1561 <latexrelease> \noexpand\reserved@b##1\E\E\relax}}%
1562 <latexrelease> \endgroup%
1563 <latexrelease> \EndIncludeInRelease
1564 <*2ekernel>

1565 \begingroup
1566 \catcode‘|= \catcode‘\%
1567 \catcode‘\%=12
1568 \catcode‘\*=11
1569 \gdef\@percentchar{%}
1570 \gdef\endfilecontents{|
1571 \immediate\closeout\reserved@c
1572 \def\T##1##2##3{|

```

```

1573 \ifx##1\@undefined\else
1574 \@latex@warning@no@line{##2 has been converted to Blank ##3e}|
1575 \fi}|
1576 \T\L{Form Feed}{Lin}|
1577 \T\I{Tab}{Spac}|
1578 \immediate\write\@unused{}}
1579 \global\let\endfilecontents*\endfilecontents

```

We no longer prevent the code to be used after begin document (no rollback needed for this change).

```

1580 %\@onlypreamble\filecontents
1581 %\@onlypreamble\endfilecontents
1582 %\@onlypreamble\filecontents*
1583 %\@onlypreamble\endfilecontents*
1584 \endgroup
1585 %\@onlypreamble\filec@ntents

```

(End of definition for \filecontents and \endfilecontents.)

5 Package/class rollback mechanism

```

1586 </2kernel>
1587 <*2kernel | latexreleasefirst>

```

`\pkgcls@debug` For testing we have a few extra lines of code that by default do nothing but one can set `\pkgcls@debug` to `\typeout` to get extra info. Sometime in the future this will be dropped.

```

1588 <*tracerollback>
1589 %\let\pkgcls@debug\typeout
1590 \let\pkgcls@debug\@gobble
1591 </tracerollback>

```

(End of definition for \pkgcls@debug.)

`\requestedLaTeXdate` The macro (!) `\requestedLaTeXdate` holds the globally requested rollback date (via `latexrelease`) or zero if no such request was made.

```

1592 \def\requestedLaTeXdate{0}

```

(End of definition for \requestedLaTeXdate.)

`\pkgcls@targetdate` If a rollback for a package or class is requested then `\pkgcls@targetdate` holds the requested date as a number YYYYMMDD (if there was one, otherwise the value of `\requestedLaTeXdate`) and `\pkgcls@targetlabel` will be empty. If there was a request for a named version then `\pkgcls@targetlabel` holds the version name and `\pkgcls@targetdate` is set to 1.

`\pkgcls@targetdate=0` is used to indicate that there was no rollback request. While loading an old release `\pkgcls@targetdate` is also reset to zero so that `\DeclareRelease` declarations are bypassed.

In contrast `\pkgcls@innerdate` will always hold the requested date (in a macro not a counter) if there was one, otherwise, e.g., if there was no request or a request to a version name it will contain T_EX largest legal number. While loading a file this can be used to provide conditionals that select code based on the request.

```

1593 \ifx\pkgcls@targetdate\@undefined

```

```

1594 \newcount\pkgcls@targetdate
1595 \fi
1596 \let\pkgcls@targetlabel\@empty
1597 \def\pkgcls@innerdate{\maxdimen}

```

(End of definition for \pkgcls@targetdate, \pkgcls@targetlabel, and \pkgcls@innerdate.)

`\pkgcls@candidate` When looping through the `\DeclareRelease` declarations we record if the release is the best candidate we have seen so far. This is recorded in `\pkgcls@candidate` and we update it whenever we see a better one.

In `\pkgcls@releasedate` we keep track of the release date of that candidate.

```

1598 \let\pkgcls@candidate\@empty
1599 \let\pkgcls@releasedate\@empty

```

(End of definition for \pkgcls@candidate and \pkgcls@releasedate.)

`\load@onefilewithoptions` the best place to add the rollback code is at the point where `\@onefilewithoptions` is called to load a single class or package.

`\@onefilewithoptions`

To make things easy we save the old definition as `\load@onefilewithoptions` and then provide a new interface.

Important: as this code is also unconditionally placed into `latexrelease` we can only do this name change once otherwise both macros will contain the same code.

```

1600 \ifx\load@onefilewithoptions\@undefined
1601 \let\load@onefilewithoptions\@onefilewithoptions
1602 \def\@onefilewithoptions#1[#2][#3]#4{%

```

First a bit of tracing normally disabled.

```

1603 (*traceroollback)
1604 \pkgcls@debug{--- File loaded request (\noexpand\usepackage or ...)}%
1605 \pkgcls@debug{\@spaces 1: #1}%
1606 \pkgcls@debug{\@spaces 2: #2}%
1607 \pkgcls@debug{\@spaces 3: #3}%
1608 \pkgcls@debug{\@spaces 4: #4}%
1609 </traceroollback>

```

Three of the arguments are needed later on in error/warning messages so we save them.

```

1610 \def\pkgcls@name{#1}% % for info message
1611 \def\pkgcls@arg {#3}% % for info message
1612 \edef\pkgcls@ext{%
1613 \ifx#4\@clsextension document class\else
1614 \ifx#4\@pkgextension package\else
1615 file
1616 \fi
1617 \fi
1618 }% % for info message

```

then we parse the final optional argument to determine if there is a specific rollback request for the current file. This will set `\pkgcls@targetdate`, `\pkgcls@targetlabel` and `\pkgcls@mindate`.

```

1619 \pkgcls@parse@date@arg{#3}%

```

When determining the correct release to load we keep track of candidates in `\pkgcls@candidate` and initially we don't have any:

```

1620 \let\pkgcls@candidate\@empty

```

If we had a rollback request then #3 may contain data but not necessarily a “minimal date” so instead of passing it on we pass on the content of `\pkgcls@mindate`. We need to pass the value not the command, otherwise nested packages may pick up the wrong information.

```

1621 \begingroup
1622 \edef\reserved@a{%
1623   \endgroup
1624   \unexpanded{\load@onefilewithoptions#1[#2]}%
1625   [\pkgcls@mindate]%
1626   \unexpanded{#4}}%
1627 \reserved@a
1628 }
1629 \fi

```

(End of definition for `\load@onefilewithoptions` and `\@onefilewithoptions`.)

`\pkgcls@parse@date@arg` The `\pkgcls@parse@date@arg` command parses the second optional argument of `\usepackage`, `\RequirePackage` or `\documentclass` for a rollback request setting the values of `\pkgcls@targetdate` and `\pkgcls@targetlabel`.

This optional argument has a dual purpose: If it just contains a date string then this means that the package should have at least that date (to ensure that a certain feature is actually available, or a certain bug has been fixed). When the package gets loaded the information in `\Provides...` will then be checked against this request.

But if it starts with an equal sign followed by a date string or followed by a version name then this means that we should roll back to the state of the package at that date or to the version with the requested name.

If there was no optional argument or the optional argument does not start with “=” then the `\pkgcls@targetdate` is set to the date of the overall rollback request (via `latexrelease`) or if that was not given it is set to 0. In either case `\pkgcls@targetlabel` will be made empty.

If the argument doesn’t start with “=” then it is supposed to be a “minimal date” and we therefore save the value in `\pkgcls@mindate`, otherwise this macro is made empty.

So in summary we have:

Input	<code>\pkgcls@targetdate</code>	<code>\pkgcls@targetlabel</code>	<code>\pkgcls@mindate</code>
<code><empty></code>	<code><global-rollbackdate-as-number></code>	<code><empty></code>	<code><empty></code>
<code><date></code>	<code><global-rollbackdate-as-number></code>	<code><empty></code>	<code><date></code>
<code>=<date></code>	<code><date-as-number></code>	<code><empty></code>	<code><empty></code>
<code>=<version></code>	1	<code><version></code>	<code><empty></code>
<code><other></code>	<code><global-rollbackdate-as-number></code>	<code><empty></code>	<code><other></code>

where `<global-rollbackdate-as-number>` is a date request given via `latexrelease` or if there wasn’t one 0.

```

1630 \def\pkgcls@parse@date@arg #1{%

```

If the argument is empty we use the rollback date from `latexrelease` which has the value of zero if there was no rollback request. The label and the minimal date is made empty in that case.

```

1631   \ifx\@nil#1\@nil
1632     \pkgcls@targetdate\requestedLaTeXdate\relax
1633     \let\pkgcls@targetlabel\@empty
1634     \let\pkgcls@mindate\@empty

```

Otherwise we parse the argument further, checking for a = as the first character. We append a = at the end so that there is at least one such character in the argument.

```

1635   \else
1636     \pkgcls@parse@date@arg@#1=\@nil\relax
1637   \fi
1638 }

```

The actual parsing work then happens in `\pkgcls@parse@date@arg@`:

```

1639 \def\pkgcls@parse@date@arg@#1=#2\@nil{%

```

We set `\pkgcls@targetdate` depending on the parsing result; the code is expandable so we can do the parsing as part of the assignment.

```

1640   \pkgcls@targetdate

```

If a = was in first position then #1 will be empty. In that case #2 will be the original argument with a = appended.

This can be parsed with `\@parse@version`, the trailing character is simply ignored. This macro returns the parsed date as a number (or zero if it wasn't a date) and accepts both YYYY/MM/DD and YYYY-MM-DD formats.

```

1641     \ifx\@nil#1\@nil
1642       \@parse@version0#2//00\@nil\relax

```

Whatever is returned is thus assigned to `\pkgcls@targetdate` and therefore we can now test its value. If the value is zero we assume that the remaining argument string represents a version and change `\pkgcls@targetdate` and set `\pkgcls@targetlabel` to the version name (after stripping off the trailing =).

```

1643     \ifnum \pkgcls@targetdate=\z@
1644       \pkgcls@targetdate\@ne
1645       \def\pkgcls@innerdate{\maxdimen}%
1646       \pkgcls@parse@date@arg@version#2%
1647     \else
1648       \edef\pkgcls@innerdate{\the\pkgcls@targetdate}%
1649     \fi
1650     \let\pkgcls@mindate\@empty
1651   \else

```

If #1 was not empty then there wasn't a = character in first position so we are dealing either with a "minimum date" or with some incorrect data. We assume the former and make the following assignments (the first one finishing the assignment of `\pkgcls@targetdate`):

```

1652     \requestedLaTeXdate\relax
1653     \let\pkgcls@targetlabel\@empty
1654     \def\pkgcls@innerdate{\maxdimen}%
1655     \def\pkgcls@mindate{#1}%

```

If the min-date is after the requested rollback date (if there is any, i.e., if it is not zero) then we have a conflict and therefore issue a warning.

```

1656     \ifnum \pkgcls@targetdate > \z@
1657       \ifnum \@parse@version0#1//00\@nil > \pkgcls@targetdate
1658         \@latex@warning@no@line{Suspicious rollback/min-date date given\MessageBreak
1659           A minimal date of #1 has been specified for
1660           \pkgcls@ext\MessageBreak '\pkgcls@name'.\MessageBreak
1661           But this is in conflict
1662           with a rollback request to \requestedpatchdate}
1663     \fi

```

```

1664     \fi
1665     \fi
1666 }

```

Strip off the trailing = and assign the version name to `\pkgcls@targetlabel`.

```

1667 \def\pkgcls@parse@date@arg@version#1={%
1668   \def\pkgcls@targetlabel{#1}}

```

(End of definition for `\pkgcls@parse@date@arg`.)

`\DeclareRelease` First argument is the “name” of the release and it can be left empty if one doesn’t like to give a name to the release. The second argument is that from which on this release was available (or should be used in case of minor updates). The final argument is the external file name of this release, by convention this should be `<pkg/cls-name>-<date>.<extension>` but this is not enforced and through this argument one can overwrite it.

```

1669 \def\DeclareRelease#1#2#3{%
1670   \ifnum\pkgcls@targetdate>\z@ % some sort of rollback request
1671   \*tracerollback
1672   \pkgcls@debug{---\string\DeclareRelease:}%
1673   \pkgcls@debug{\@spaces 1: #1}%
1674   \pkgcls@debug{\@spaces 2: #2}%
1675   \pkgcls@debug{\@spaces 3: #3}%
1676   \*tracerollback

```

If the date argument #2 is empty we are dealing with a special release that should be only accessible via its name; a typical use case would be a “beta” release. So if we are currently processing a date request we ignore it and otherwise we check if we can match the name and if so load the corresponding release file.

```

1677     \ifx\@nil#2\@nil
1678       \ifnum\pkgcls@targetdate=\@ne % named request
1679         \def\reserved@a{#1}%
1680         \ifx\pkgcls@targetlabel\reserved@a
1681           \pkgcls@use@this@release{#3}{}%
1682         \*tracerollback
1683       \else
1684         \pkgcls@debug{Label doesn't match}%
1685       \*tracerollback
1686     \fi
1687   \*tracerollback
1688   \else
1689     \pkgcls@debug{Date request: ignored}%
1690   \*tracerollback
1691   \fi
1692   \else

```

If the value of `\pkgcls@targetdate` is greater than 1 (or in reality greater than something like 19930101) we are dealing with a rollback request to a specific date.

```

1693     \ifnum\pkgcls@targetdate>\@ne % a real request

```

So we parse the date of this release to check if it is before or after the request date.

```

1694     \ifnum\@parse@version#2//00\@nil
1695       >\pkgcls@targetdate

```


If it is after we have to distinguish between two cases: If there was an earlier candidate we use that one because the other is too late, but if there wasn't one (i.e., if current release is the oldest that exists) we use it as the best choice. However in that case something is wrong (as there shouldn't be a rollback to a date when a package used didn't yet exists). So we make a complained to the user.

```

1696     \ifx\pkgcls@candidate\@empty
1697         \pkgcls@rollbackdate@error{#2}%
1698         \pkgcls@use@this@release{#3}{#2}%
1699     \else
1700         \pkgcls@use@this@release\pkgcls@candidate
1701             \pkgcls@releasedate
1702     \fi
1703 \else

```

Otherwise, if the release date of this version is before the target rollback and we record it as a candidate. But we don't use it yet as there may be another release which is still before the target rollback.

```

1704     \def\pkgcls@candidate{#3}%
1705     \def\pkgcls@releasedate{#2}%
1706 < *tracereollback)
1707     \pkgcls@debug{New candidate: #3}%
1708 < /tracereollback)
1709     \fi
1710 \else

```

If we end up in this branch we have a named version request. So we check if `\pkgcls@targetlabel` matches the current name and if yes we use this release immediately, otherwise we do nothing as a later declaration may match it.

```

1711     \def\reserved@a{#1}%
1712     \ifx\pkgcls@targetlabel\reserved@a
1713         \pkgcls@use@this@release{#3}{#2}%
1714 < *tracereollback)
1715     \else
1716         \pkgcls@debug{Label doesn't match}%
1717 < /tracereollback)
1718     \fi
1719 \fi
1720 \fi
1721 \fi
1722 }

```

(End of definition for \DeclareRelease.)

`\pkgcls@use@this@release` If a certain release has been selected (stored in the external file given in #1) we need to input it and afterwards stop reading the current file.

```

1723 \def\pkgcls@use@this@release#1#2{%

```

Before that we record the selection made inside the transcript.

```

1724     \pkgcls@show@selection{#1}{#2}%

```

We then set the `\pkgcls@targetdate` to zero so that any `\DeclareRelease` or `\DeclareCurrentRelease` in the file we now load are bypassed⁵¹ and then we finally load the correct release.

⁵¹The older release may also have such declarations inside if it was a simply copy of the `.sty` or `.cls` file current at that date. Removing these declarations would make the file load a tiny bit faster, but this way it works in any case.

After loading that file we need to stop reading the current file so we issue `\endinput`. Note that the `\relax` before that is essential to ensure that the `\endinput` is only happening after the file has been fully processed, otherwise it would act after the first line of the `\@@input!`

```
1725 \pkgcls@targetdate\z@
1726 \@addtofilelist{#1}%
1727 \@@input #1\relax
1728 \endinput
1729 }
```

(End of definition for \pkgcls@use@this@release.)

`\pkgcls@show@selection` This command records what selection was made. As that is needed in two places (and it is rather lengthy) it was placed in a separate command. The first argument is the name of the external file that is being loaded and is only needed for debugging. The second argument is the date that corresponds to this file and it is used as part of the message.

```
1730 \def\pkgcls@show@selection#1#2{%
1731 <*tracerollback>
1732 \pkgcls@debug{Result: use #1}%
1733 </tracerollback>
1734 \GenericInfo
1735 {\@spaces\@spaces\space}{Rollback for
1736 \@cls@pkg\space'\@currname' requested ->
1737 \ifnum\pkgcls@targetdate>\@ne
1738 date
1739 \ifnum\requestedLaTeXdate=\pkgcls@targetdate
1740 \requestedpatchdate
1741 \else
1742 \expandafter\@gobble\pkgcls@arg
1743 \fi.\MessageBreak
```

Instead of “best approximation” we could say that we have been able to exactly match the date (if it is exact), but that would mean extra tests without much gain, so not done.

```
1744 Best approximation is
1745 \else
1746 version '\pkgcls@targetlabel'.\MessageBreak
1747 This corresponds to
1748 \fi
1749 \ifx\@nil#2\@nil
1750 a special release%
1751 \else
1752 the release introduced on #2%
1753 \fi
1754 \@gobble}%
1755 }
```

(End of definition for \pkgcls@show@selection.)

`\pkgcls@rollbackdate@error` This is called if the requested rollback date is earlier than the earliest known release of a package or class.

A similar error is given if global rollback date and min-date on a specific package conflict with each other, but that case is happens only once so it is inlined.

```
1756 \def\pkgcls@rollbackdate@error#1{%
1757 \latex@error{Suspicious rollback date given}%
```

```

1758     {The \cls@pkg\space'\@currname' has no rollback data
1759     before #1 which\MessageBreak
1760     is after your requested rollback date --- so
1761     something may be wrong here.\MessageBreak
1762     Continue and we use the earliest known release.}}

```

(End of definition for \pkgcls@rollbackdate@error.)

\DeclareCurrentRelease This declares the date (and possible name) of the current version of a package or class.

```

1763 \def\DeclareCurrentRelease#1#2{%

```

First we test if \pkgcls@targetdate is greater than zero, otherwise this code is bypassed (as there is no rollback request).

```

1764   \ifnum\pkgcls@targetdate>\z@ % some sort of rollback request
1765   {*tracerollback)
1766     \pkgcls@debug{---DeclareCurrentRelease}%
1767     \pkgcls@debug{ 1: #1}%
1768     \pkgcls@debug{ 2: #2}%
1769   (/tracerollback)

```

If the value is greater than 1 we have to deal with a date request, so we parse #2 as a date and compare it with \pkgcls@targetdate.

```

1770     \ifnum\pkgcls@targetdate>\@ne % a date request
1771     \ifnum\@parse@version#2//00\@nil
1772       >\pkgcls@targetdate

```

If it is greater that means the release date if this file is later than the requested rollback date. Again we have two cases: If there was a previous candidate release we use that one as the current release is too young, but if there wasn't we have to use this release nevertheless as there isn't any alternative.

However this case can only happen if there is a \DeclareCurrentRelease but no declared older releases (so basically the use of the declaration is a bit dubious).

```

1773     \ifx\pkgcls@candidate\@empty
1774     \pkgcls@rollbackdate@error{#2}%
1775     \else
1776     \pkgcls@use@this@release\pkgcls@candidate
1777     \pkgcls@releasedate
1778     \fi

```

Otherwise the current file is the right release, so we record that in the transcript and then carry on.

```

1779     \else
1780     \pkgcls@show@selection{current version}{#2}%
1781     \fi
1782   \else % a label request

```

Otherwise we have a rollback request to a named version so we check if that fits the current name and if not give an error as this was the last possible opportunity.

```

1783     \def\reserved@a{#1}%
1784     \ifx\pkgcls@targetlabel\reserved@a
1785     \pkgcls@show@selection{current version}{#2}%
1786     \else
1787     \@latex@error{Requested version '\pkgcls@targetlabel' for
1788     \cls@pkg\space'\@currname' is unknown}\@ehc
1789     \fi

```

```

1790     \fi
1791 \fi
1792 }

```

(End of definition for `\DeclareCurrentRelease`.)

\IfTargetDateBefore This enables a simple form of conditional code inside a class or package file. If there is a date request and the request date is earlier than the first argument the code in the second argument is processed otherwise the code in the third argument is processed. If there was no date request then we also execute the third argument, i.e., we will get the “latest” version of the file.

Most often the second argument (before-date-code) will be empty.

```

1793 \DeclareRobustCommand\IfTargetDateBefore[1]{%
1794   \ifnum\pkgcls@innerdate <%
1795     \expandafter\@parse@version\expandafter0#1//00\@nil
1796     \typeout{Exclude code introduced on #1}%
1797     \expandafter\@firstoftwo
1798   \else
1799     \typeout{Include code introduced on #1}%
1800     \expandafter\@secondoftwo
1801   \fi
1802 }

```

(End of definition for `\IfTargetDateBefore`.)

```

1803 </2ekernel | latexreleasefirst)

```

6 After Preamble

Finally we declare a package that allows all the commands declared above to be `\@onlypreamble` to be used after `\begin{document}`.

```

1804 <*afterpreamble)
1805 \NeedsTeXFormat{LaTeX2e}
1806 \ProvidesPackage{pkgindoc}
1807     [2020-08-08 v1.3m Package Interface in Document (DPC)]
1808 \def\reserved@a#1\do\@classoptionslist#2\do\filec@ntents#3\relax{%
1809   \gdef\@preamblecmds{#1#3}}
1810 \expandafter\reserved@a\@preamblecmds\relax
1811 </afterpreamble)

```

File 51

ltkeys.dtx

1 Creating and using keyval options

As with any key–value input, using key–value pairs as package or class options has two parts: creating the key options and setting (using) them. Options created in this way *may* be used after package loading as general key–value settings: this will depend on the nature of the underlying code.

```
\DeclareKeys \DeclareKeys [family] {declarations}
```

Creates a series of options from a comma-separated *declarations* list. Each entry in this list is a key–value pair, with the *key* having one or more *properties*. A small number of “basic” *properties* are described below. The full range of properties, provided by `l3keys`, can also be used for more powerful processing. See `interface3` for the full details.

The basic properties provided here are

- `.code` — execute arbitrary code
- `.if` — sets a `\TeX \if...` switch
- `.ifnot` — sets an inverted `\TeX \if...` switch
- `.pass-to-packages` — for class options, this specifies whether the option should be treated “global” (read by packages from the global list); for package options this property has no effect
- `.store` — stores a value in a macro
- `.usage` — defines whether the option can be given only when loading (`load`), in the preamble (`preamble`) or has no limitation on scope (`general`)

The part of the *key* before the *property* is the *name*, with the *value* working with the *property* to define the behaviour of the option.

For example, with

```
\DeclareKeys [mypkg]
{
  draft.if          = @mypkg@draft      ,
  draft.usage       = preamble          ,
  name.store        = \@mypkg@name      ,
  name.usage        = load              ,
  second-name.store = \@mypkg@other@name
}
```

three options would be create. The option `draft` can be given anywhere in the preamble, and will set a switch called `\if@mypkg@draft`. The option `name` can only be given during package loading, and will save whatever value it is given in `\@mypkg@name`. Finally, the option `second-name` can be given anywhere, and will save its value in `\@mypkg@other@name`.

Keys created *before* the use of `\ProcessKeyOptions` act as package options.

`\DeclareUnknownKeyHandler` `\DeclareUnknownKeyHandler [<family>] {<code>}`

The function `\DeclareUnknownKeyHandler` may be used to define the behavior when an undefined key is encountered. The `<code>` will receive the unknown key name as `#1` and the value as `#2`. These can then be processed as appropriate, e.g. by forwarding to another package.

`\ProcessKeyOptions` `\ProcessKeyOptions [<family>]`

The `\ProcessKeyOptions` function is used to check the current option list against the keys defined for `<family>`. Global (class) options and local (package) options are checked when this function is called in a package.

`\SetKeys` `\SetKeys [<family>] {<keyvals>}`

Sets (applies) the explicit list of `<keyvals>` for the `<family>`: if the latter is not given, the value of `\@currname` is used. This command may be used within a package to set options before or after using `\ProcessKeyOptions`.

1.1 Implementation of `ltkeys`

```
1 <@@=keys>
2 <*2kernel>
3 \ExplSyntaxOn
```

1.2 Key properties

```
.code
.if
4 \group_begin:
5 \cs_set_protected:Npn \__keys_tmp:nn #1#2
6 {
7 \quark_if_recursion_tail_stop:n {#1}
8 \cs_new_eq:cc
9 { \c__keys_props_root_str . #2 }
10 { \c__keys_props_root_str . #1 }
11 \__keys_tmp:nn
12 }
13 \__keys_tmp:nn
14 { code:n } { code }
15 { legacy_if_set:n } { if }
16 { legacy_if_set_inverse:n } { ifnot }
17 { tl_set:N } { store }
18 { usage:n } { usage }
19 { \q_recursion_tail } { }
20 \q_recursion_stop
21 \group_end:
```

(End of definition for `.code` and others.)

`.pass-to-packages` Used to force options to be global: as this property (uniquely) has an *optional* value, there is a bit of work to do.

```
\__keys_scope:n
\__keys_scope:N
22 \cs_new_protected:cpn { \c__keys_props_root_str .pass-to-packages }
23 {
24 \bool_if:NTF \l__keys_no_value_bool
```

```

25     { \_keys_scope:n { true } }
26     { \_keys_scope:n }
27   }
28 \cs_new_protected:Npn \_keys_scope:n #1
29   {
30     \str_case:nnF {#1}
31     {
32       { true }
33       { \_keys_scope:N \clist_put_right:NV }
34       { false }
35       { \_keys_scope:N \clist_remove_all:NV }
36     }
37     {
38       \msg_error:nnnn { keys }
39       { choice-unknown }
40       { .pass-to-packages }
41       {#1}
42     }
43   }
44 \cs_new_protected:Npn \_keys_scope:N #1
45   {
46     \exp_after:wN \_keys_find_key_module:wNN
47     \l_keys_path_str \s_keys_stop
48     \l_keys_key_tl \l_keys_key_str
49     #1 \l_keys_forced_global_clist \l_keys_key_str
50   }

```

(End of definition for `.pass-to-packages`, `_keys_scope:n`, and `_keys_scope:N`.)

1.3 Main mechanism

```

51 \cs_generate_variant:Nn \clist_if_in:NnT { Ne }
52 \cs_generate_variant:Nn \clist_if_in:NnTF { Ne }
53 \cs_generate_variant:Nn \clist_put_right:Nn { Nv }

```

`\l_keys_class_only_clist` Used to track class-only options.

```
54 \clist_new:N \l_keys_class_only_clist
```

(End of definition for `\l_keys_class_only_clist`.)

`\l_keys_forced_global_clist` Used to force options to be global.

```
55 \clist_new:N \l_keys_forced_global_clist
```

(End of definition for `\l_keys_forced_global_clist`.)

`\l_keys_options_clist` A single list is used for all options, into which they are collected before processing.

```
56 \clist_new:N \l_keys_options_clist
```

(End of definition for `\l_keys_options_clist`.)

`\l_keys_options_loading_bool`

Used to indicate we are in the loading phase: controls the outcome of warnings.

```
57 \bool_new:N \l_keys_options_loading_bool
```

```

    \_keys_options:n The main function calls functions to collect up the global and local options into \l_
\_keys_options_aux:n keys_options_clist before calling the underlying functions to actually do the pro-
\_keys_options_end: cessing. So that a suitable message is produced if the option is unknown, the special
                    unknown key is set if it does not already exist for the current family, and is cleaned up
                    afterwards if required. To allow the LATEX 2ε layer to know this mechanism is active, and
                    to deal with the key family not matching the file name, we store the family in all cases.
                    Global options are only considered the first time a package is loaded: this is tracked using
                    opt@handler@\@currname.\@current, as this is defined once keyval processing has been
                    applied for the first time.
58 \cs_new_protected:Npn \_keys_options:n #1
59   { \_keys_options_expand_module:Nn \_keys_options_aux:n {#1} }
60 \cs_new_protected:Npn \_keys_options_aux:n #1
61   {
62     \cs_set_protected:Npn \_keys_option_end: { }
63     \clist_clear:N \l_keys_options_clist
64     \cs_if_exist:cF { opt@handler@ \@currname . \@current }
65       {
66         \_keys_options_global:n {#1}
67         \cs_gset_protected:cpn { opt@handler@ \@currname . \@current }
68           { \ProcessKeyOptions [ #1 ] }
69       }
70     \_keys_options_local:
71     \keys_if_exist:nnF {#1} { unknown }
72     {
73       \keys_define:nn {#1}
74       {
75         unknown .code:n =
76         {
77           \msg_error:nnxx { keys } { option-unknown }
78           { \l_keys_key_str } { \@currname }
79         }
80       }
81       \cs_set_protected:Npn \_keys_option_end:
82         { \keys_define:nn {#1} { unknown .undefine: } }
83     }
84     \bool_set_true:N \l_keys_options_loading_bool
85     \clist_map_variable:NNn \l_keys_options_clist \CurrentOption
86       { \keys_set:nV {#1} \CurrentOption }
87     \bool_set_false:N \l_keys_options_loading_bool
88     \AtEndOfPackage { \cs_set_eq:NN \@unprocessedoptions \scan_stop: }
89     \_keys_option_end:
90     \_keys_options_loaded:n {#1}
91   }

92 \msg_new:nnnn { keys } { option-unknown }
93   { Unknown~option~'#1'~for~package~#2. }
94   {
95     LaTeX~has~been~asked~to~set~an~option~called~'#1'~
96     but~the~package~" \msg_module_name:n {#2}"~has~not~created~an~option~with~this~name.
97   }

```

(End of definition for _keys_options:n, _keys_options_aux:n, and _keys_options_end:.)

`__keys_options_global:n` Global (class) options are handled differently for L^AT_EX 2_ε packages and classes. Hence this function is essentially a check on the current file type. The initial test is needed as L^AT_EX 2_ε allows variables to be equal to `\scan_stop:`, which is usually forbidden in expl3 code.

```

98 \cs_new_protected:Npn \__keys_options_global:n #1
99   {
100     \cs_if_eq:NNF \@raw@classoptionslist \scan_stop:
101     {
102       \cs_if_eq:NNTF \@currentt \clsextension
103       { \__keys_options_class:n {#1} }
104       { \__keys_options_package:n {#1} }
105     }
106   }

```

(End of definition for __keys_options_global:n)

`__keys_options_class:n` For classes, each option (stripped of any content after =) is checked for existence as a key.
`__keys_options_class:nnn` If found, the option is added to the combined list for processing. On the other hand,
`__keys_options_class:nn` unused options are stored up in `\@unusedoptionlist`. An earlier version of this code checked for the `unknown` key just once and if found short-cuttet the loop: that though makes handling more complex situations harder, so we take the performance hit instead. Options used by classes are tracked but the catch-all `unknown` is excluded (hence not using a lazy evaluation for the key testing).

```

107 \cs_new_protected:Npn \__keys_options_class:n #1
108   {
109     \cs_if_free:cF { \@raw@opt@ \@currname . \@currentt }
110     {
111       \clist_map_inline:cn { \@raw@opt@ \@currname . \@currentt }
112       {
113         \exp_args:Ne \__keys_options_class:nnn
114         { \tl_trim_spaces:e { \__keys_remove_equals:n {##1} } }
115         {##1} {#1}
116       }
117     }
118   }
119 \cs_new_protected:Npn \__keys_options_class:nnn #1#2#3
120   {
121     \keys_if_exist:nnTF {#3} {#1}
122     {
123       \__keys_options_class:nn {#1} {#2}
124       \clist_put_right:Ne \l__keys_class_only_clist { \tl_to_str:n {#1} }
125     }
126     {
127       \keys_if_exist:nnTF {#3} { unknown }
128       { \__keys_options_class:nn {#1} {#2} }
129       {
130         \clist_if_in:NnF \@unusedoptionlist {#1}
131         { \clist_put_right:Nn \@unusedoptionlist {#1} }
132       }
133     }
134   }
135 \cs_new_protected:Npn \__keys_options_class:nn #1#2
136   {

```

```

137   \clist_remove_all:Nn \@unusedoptionlist {#1}
138   \clist_put_right:Nn \l__keys_options_clist {#2}
139 }

```

(End of definition for `__keys_options_class:n`, `__keys_options_class:nnn`, and `__keys_options_class:nn`.)

`__keys_options_package:n` For global options when processing a package, the tasks are slightly different from those for a class. The check is the same, but here there is nothing to do if the option is not applicable. Each valid option also needs to be removed from `\@unusedoptionlist`.

```

140 \cs_new_protected:Npn \__keys_options_package:n #1
141 {
142   \clist_map_inline:Nn \@raw@classoptionslist
143   {
144     \exp_args:Ne \__keys_options_package:nnn
145     { \tl_trim_spaces:e { \__keys_remove_equals:n {##1} } }
146     {##1} {#1}
147   }
148 }

```

The forced-global test here needs to use `\tl_to_str:n` as the data come from a key name, which is always a string.

```

149 \cs_new_protected:Npn \__keys_options_package:nnn #1#2#3
150 {
151   \keys_if_exist:nnT {#3} {#1}
152   {
153     \clist_if_in:NeTF \l__keys_class_only_clist { \tl_to_str:n {#1} }
154     {
155       \clist_if_in:NeT \l__keys_forced_global_clist { \tl_to_str:n {#1} }
156       { \__keys_options_package:nn {#1} {#2} }
157     }
158     { \__keys_options_package:nn {#1} {#2} }
159   }
160 }
161 \cs_new_protected:Npn \__keys_options_package:nn #1#2
162 {
163   \clist_put_right:Nn \l__keys_options_clist {#2}
164   \clist_remove_all:Nn \@unusedoptionlist {#1}
165 }

```

(End of definition for `__keys_options_package:n`, `__keys_options_package:nnn`, and `__keys_options_package:nn`.)

`__keys_options_local:` If local options are found, they are added to the processing list. L^AT_EX 2_ε stores options for each file in a macro which may or may not exist, hence the need to use `\cs_if_exist:c`.

```

166 \cs_new_protected:Npn \__keys_options_local:
167 {
168   \cs_if_eq:NNF \@currentt \@clsextension
169   {
170     \cs_if_exist:cT { @raw@opt@ \@currname . \@currentt }
171     {
172       \clist_put_right:Nv \l__keys_options_clist
173       { @raw@opt@ \@currname . \@currentt }
174     }
175   }
176 }

```

(End of definition for `_keys_options_local:`.)

`_keys_remove_equals:n` As the name suggests, this is a simple function to remove an equals sign from the input.
`_keys_remove_equals:w` This is all wrapped up in an `n` function so that there will always be a sign available.

```
177 \cs_new:Npn \_keys_remove_equals:n #1
178   { \_keys_remove_equals:w #1 = \s_keys_stop }
179 \cs_new:Npn \_keys_remove_equals:w #1 = #2 \s_keys_stop { \exp_not:n {#1} }
```

(End of definition for `_keys_remove_equals:n` and `_keys_remove_equals:w`.)

1.4 The document interfaces

```
180 \cs_generate_variant:Nn \keys_define:nn { nx }
```

`_keys_options_expand_module:Nn` To deal with active characters inside the module argument whilst also expanding that
`_keys_options_expand_module:nN` argument, we use a combination of `c`- and `f`-type expansion. This works as the definitions
for active UTF-8 bytes contain an `\ifincsname` test.

```
181 \cs_new_protected:Npn \_keys_options_expand_module:Nn #1#2
182   {
183     \cs:w \_keys_options_expand_module:nN \use:e { \cs_end: {#2} } #1
184   }
185 \cs_new_protected:Npn \_keys_options_expand_module:nN #1#2
186   { #2 {#1} }
```

(End of definition for `_keys_options_expand_module:Nn` and `_keys_options_expand_module:nN`.)

`\DeclareKeys` Defining key options is quite straight-forward: we have an intermediate function to allow
for potential set-up steps.

```
187 \NewDocumentCommand \DeclareKeys { 0 { \@currname } +m }
188   { \_keys_options_expand_module:Nn \keys_define:nn {#1} {#2} }
```

(End of definition for `\DeclareKeys`. This function is documented on page 1091.)

`\DeclareUnknownKeyHandler`

```
189 \NewDocumentCommand \DeclareUnknownKeyHandler { 0 { \@currname } +m }
190   {
191     \cs_set_protected:cpn { \_keys_unknown_handler_ #1 :nn } ##1##2 {#2}
192     \_keys_options_expand_module:Nn \keys_define:nx {#1}
193     {
194       unknown .code:n =
195         \exp_not:N \exp_args:NV
196         \exp_not:c { \_keys_unknown_handler_ #1 :nn }
197         \exp_not:N \l_keys_key_str {###1}
198     }
199   }
```

(End of definition for `\DeclareUnknownKeyHandler`. This function is documented on page 1092.)

`\ProcessKeyOptions` We need to deal with the older interface from `l3keys2e` here: it had a mandatory argument.
We can mop that up using a look-ahead, and then exploit that information to determine
whether the package option handling is set up for the new approach for clash handling.

```
200 \NewDocumentCommand \ProcessKeyOptions { 0 { \@currname } }
201   { \_keys_options:n {#1} }
202 \@onlypreamble \ProcessKeyOptions
```

(End of definition for `\ProcessKeyOptions`. This function is documented on page 1092.)

1.5 Option usage scope

`_keys_options_loaded:n` Indicates that the load-time options for a package have been processed: once this has happened, make them unavailable either with a warning or an error.

```

203 \cs_new_protected:Npn \_keys_options_loaded:n #1
204   {
205     \prop_get:NnNT \l_keys_usage_load_prop {#1} \l_keys_tmpa_tl
206     {
207       \clist_map_inline:Nn \l_keys_tmpa_tl
208       {
209         \keys_define:nn {#1}
210         {
211           ##1 .code:n =
212             \_keys_options_loaded:nn {#1} {##1}
213         }
214       }
215     }
216   }
217 \cs_new_protected:Npn \_keys_options_loaded:nn #1#2
218   {
219     \bool_if:NTF \l_keys_options_loading_bool
220     { \msg_warning:nnnn { keys } { load-option-ignored } }
221     { \msg_error:nnnn { keys } { load-only } }
222     {#1} {#2}
223   }
224 \msg_new:nnn { keys } { load-option-ignored }
225   {
226     Package~"\msg_module_name:n {#1}"~has~already~been~loaded:~
227     ignoring~load-time~option~"#2".
228   }
229 \msg_new:nnnn { keys } { load-only }
230   {
231     Key~"#2"~may~only~be~used~during~loading~of~package~
232     "\msg_module_name:n {#1}".
233   }
234   {
235     LaTeX~was~asked~to~set~a~key~called~"#2",~but~this~is~only~allowed~
236     in~the~optional~argument~when~loading~package~"\msg_module_name:n{#1}".
237   }

```

(End of definition for `_keys_options_loaded:n` and `_keys_options_loaded:nn`.)

Disable all preamble options in one shot.

```

238 \tl_gput_left:Nn \@kernel@after@begindocument
239   {
240     \prop_map_inline:Nn \l_keys_usage_preamble_prop
241     {
242       \clist_map_inline:nn {#2}
243       {
244         \keys_define:nn {#1}
245         {
246           ##1 .code:n =
247             \msg_error:nnn { keys } { preamble-only } {##1}
248         }

```

```

249     }
250   }
251 }
252 \msg_new:nnon { keys } { preamble-only }
253 { Key~"#1"~may~only~be~used~in~the~preamble. }
254 {
255   LaTeX~was~asked~to~set~a~key~called~"#1",~but~this~is~only~allowed~
256   before~\begin{document}.~You~will~need~to~set~the~key~earlier.
257 }

```

1.6 General key setting

\SetKeys A simple wrapper.

```

258 \NewDocumentCommand \SetKeys { 0 { \@currname } +m }
259 { \_keys_options_expand_module:Nn \keys_set:nn {#1} {#2} }

```

(End of definition for \SetKeys. This function is documented on page 1092.)

```

260 \ExplSyntaxOff
261 </2ekernel>

```

File 52

lfilehook.dtx

1 Introduction

1.1 Provided hooks

The code offers a number of hooks into which packages (or the user) can add code to support different use cases. Many hooks are offered as pairs (i.e., the second hook is reversed). Also important to know is that these pairs are properly nested with respect to other pairs of hooks.

There are hooks that are executed for all files of a certain type (if they contain code), e.g., for all “include files” or all “packages”, and there are also hooks that are specific to a single file, e.g., do something after the package `foo.sty` has been loaded.

1.2 General hooks for file reading

There are four hooks that are called for each file that is read using document-level commands such as `\input`, `\include`, `\usepackage`, etc. They are not called for files read using internal low-level methods, such as `\@input` or `\openin`.

<code>file/before</code>	These are:
<code>file/.../before</code>	
<code>file/.../after</code>	file/before, file/⟨file-name⟩/before These hooks are executed in that order just
<code>file/after</code>	before the file is loaded for reading. The code of the first hook is used with every file,

while the second is executed only for the file with matching `⟨file-name⟩` allowing you to specify code that only applies to one file.

file/⟨file-name⟩/after, file/after These hooks are after the file with name `⟨file-name⟩` has been fully consumed. The order is swapped (the specific one comes first) so that the `/before` and `/after` hooks nest properly, which is important if any of them involve grouping (e.g., contain environments, for example). Furthermore both hooks are reversed hooks to support correct nesting of different packages adding code to both `/before` and `/after` hooks.

So the overall sequence of hook processing for any file read through the user interface commands of L^AT_EX is:

```
\UseHook{file/before}
\UseHook{file/⟨file name⟩/before}
  ⟨file contents⟩
\UseHook{file/⟨file name⟩/after}
\UseHook{file/after}
```

The file hooks only refer to the file by its name and extension, so the `⟨file name⟩` should be the file name as it is on the filesystem with extension (if any) and without paths. Different from `\input` and similar commands, the `.tex` extension is not assumed in hook `⟨file name⟩`, so `.tex` files must be specified with their extension to be recognized. Files within subfolders should also be addressed by their name and extension only.

Extensionless files also work, and should then be given without extension. Note however that \TeX prioritizes `.tex` files, so if two files `foo` and `foo.tex` exist in the search path, only the latter will be seen.

When a file is input, the $\langle file\ name \rangle$ is available in `\CurrentFile`, which is then used when accessing the `file/\langle file\ name \rangle/before` and `file/\langle file\ name \rangle/after`.

`\CurrentFile` The name of the file about to be read (or just finished) is available to the hooks through `\CurrentFile` (there is no `expl3` name for it for now). The file is always provided with its extension, i.e., how it appears on your hard drive, but without any specified path to it. For example, `\input{sample}` and `\input{app/sample.tex}` would both have `\CurrentFile` being `sample.tex`.

`\CurrentFilePath` The path to the current file (complement to `\CurrentFile`) is available in `\CurrentFilePath` if needed. The paths returned in `\CurrentFilePath` are only user paths, given through `\input@path` (or `expl3`'s equivalent `\l_file_search_path_seq`) or by directly typing in the path in the `\input` command or equivalent. Files located by `kpsewhich` get the path added internally by the \TeX implementation, so at the macro level it looks as if the file were in the current folder, so the path in `\CurrentFilePath` is empty in these cases (package and class files, mostly).

`\CurrentFileUsed`
`\CurrentFilePathUsed` In normal circumstances these are identical to `\CurrentFile` and `\CurrentFilePath`. They will differ when a file substitution has occurred for `\CurrentFile`. In that case, `\CurrentFileUsed` and `\CurrentFilePathUsed` will hold the actual file name and path loaded by \LaTeX , while `\CurrentFile` and `\CurrentFilePath` will hold the names that were *asked for*. Unless doing very specific work on the file being read, `\CurrentFile` and `\CurrentFilePath` should be enough.

1.3 Hooks for package and class files

Commands to load package and class files (e.g., `\usepackage`, `\RequirePackage`, `\LoadPackageWithOptions`, etc.) offer the hooks from section 1.2 when they are used to load a package or class file, e.g., `file/array.sty/after` would be called after the `array` package got loaded. But as packages and classes form a special group of files, there are some additional hooks available that only apply when a package or class is loaded.

<code>package/before</code>	These are:
<code>package/after</code>	
<code>package/.../before</code>	<code>package/before</code>, <code>package/after</code> These hooks are called for each package being loaded.
<code>package/.../after</code>	
<code>class/before</code>	<code>package/\langle name \rangle/before</code>, <code>package/\langle name \rangle/after</code> These hooks are additionally called if the package name is $\langle name \rangle$ (without extension).
<code>class/after</code>	
<code>class/.../before</code>	<code>class/before</code>, <code>class/after</code> These hooks are called for each class being loaded.
<code>class/.../after</code>	
	<code>class/\langle name \rangle/before</code>, <code>class/\langle name \rangle/after</code> These hooks are additionally called if the class name is $\langle name \rangle$ (without extension).

All `/after` hooks are implemented as reversed hooks.

The overall sequence of execution for `\usepackage` and friends is:

```
\UseHook{package/before}
\UseOneTimeHook{package/<package name>/before}

  \UseHook{file/before}
  \UseHook{file/<package name>.sty/before}
  <package contents>
  \UseHook{file/<package name>.sty/after}
  \UseHook{file/after}

  code from \AtEndOfPackage if used inside the package

\UseOneTimeHook{package/<package name>/after}
\UseHook{package/after}
```

and similar for class file loading, except that `package/` is replaced by `class/` and `\AtEndOfPackage` by `\AtEndOfClass`.

If a package or class is not loaded none of the hooks are executed!

All class or package hooks involving the name of the class or package are implemented as one-time hooks, whereas all other such hooks are normal hooks. This allows for the following use case

```
\AddToHook{package/varioreref/after}
{ ... apply my customizations if the package gets
  loaded (or was loaded already) ... }
```

without the need to first test if the package is already loaded.

1.4 Hooks for `\include` files

To manage `\include` files, \LaTeX issues a `\clearpage` before and after loading such a file. Depending on the use case one may want to execute code before or after these `\clearpages` especially for the one that is issued at the end.

Executing code before the final `\clearpage`, means that the code is processed while the last page of the included material is still under construction. Executing code after it means that all floats from inside the include file are placed (which might have added further pages) and the final page has finished.

Because of these different scenarios we offer hooks in three places.⁵² None of the hooks are executed when an `\include` file is bypassed because of an `\includeonly` declaration. They are, however, all executed if \LaTeX makes an attempt to load the `\include` file (even if it doesn't exist and all that happens is “No file `<filename>.tex`”).

⁵²If you want to execute code before the first `\clearpage` there is no need to use a hook—you can write it directly in front of the `\include`.

<code>include/before</code> <code>include/.../before</code> <code>include/end</code> <code>include/.../end</code> <code>include/after</code> <code>include/.../after</code>	These are: <code>include/before, include/<name>/before</code> These hooks are executed (in that order) after the initial <code>\clearpage</code> and after <code>.aux</code> file is changed to use <code><name>.aux</code> , but before the <code><name>.tex</code> file is loaded. In other words they are executed at the very beginning of the first page of the <code>\include</code> file.
--	--

`include/<name>/end, include/end` These hooks are executed (in that order) after \LaTeX has stopped reading from the `\include` file, but before it has issued a `\clearpage` to output any deferred floats.

`include/<name>/after, include/after` These hooks are executed (in that order) after \LaTeX has issued the `\clearpage` but before it has switched back writing to the main `.aux` file. Thus technically we are still inside the `\include` and if the hooks generate any further typeset material including anything that writes to the `.aux` file, then it would be considered part of the included material and bypassed if it is not loaded because of some `\includeonly` statement.⁵³

`include/excluded, include/<name>/excluded` The above hooks for `\include` files are only executed when the file is loaded (or more exactly the load is attempted). If, however, the `\include` file is explicitly excluded (through an `\includeonly` statement) the above hooks are bypassed and instead the `include/excluded` hook followed by the `include/<name>/excluded` hook are executed. This happens after \LaTeX has loaded the `.aux` file for this include file, i.e., after \LaTeX has updated its counters to pretend that the file was seen.

All `include` hooks involving the name of the included file are implemented as one-time hooks (whereas all other such hooks are normal hooks).

If you want to execute code that is run for every `\include` regardless of whether or not it is excluded, use the `cmd/include/before` or `cmd/include/after` hooks.

1.5 High-level interfaces for \LaTeX

We do not provide any additional wrappers around the hooks (like `filehook` or `scrfile` do) because we believe that for package writers the high-level commands from the hook management, e.g., `\AddToHook`, etc. are sufficient and in fact easier to work with, given that the hooks have consistent naming conventions.

⁵³For that reason another `\clearpage` is executed after these hooks which normally does nothing, but starts a new page if further material got added this way.

1.6 Kernel, class, and package interfaces for L^AT_EX

```
\declare@file@substitution \declare@file@substitution  {\file} {\replacement-file}  
\undeclare@file@substitution \undeclare@file@substitution {\file}
```

If $\langle file \rangle$ is requested for loading replace it with $\langle replacement-file \rangle$. `\CurrentFile` remains pointing to $\langle file \rangle$ but `\CurrentFileUsed` will show the file actually loaded.

The main use case for this declaration is to provide a corrected version of a package that can't be changed (due to its license) but no longer functions because of L^AT_EX kernel changes, for example, or to provide a version that makes use of new kernel functionality while the original package remains available for use with older releases. As such it is mainly meant for use in the L^AT_EX kernel but other use cases are conceivable.

The `\undeclare@file@substitution` declaration undoes a substitution made earlier.

Please do not misuse this functionality and replace a file with another unless if really needed and only if the new version is implementing the same functionality as the original one!

```
\disable@package@load \disable@package@load  {\package} {\alternate-code}  
\reenable@package@load \reenable@package@load {\package}
```

If $\langle package \rangle$ is requested, do not load it but instead run $\langle alternate-code \rangle$ which could issue a warning, error or any other code.

The main use case is for classes that want to restrict the set of supported packages or contain code that make the use of some packages impossible. So rather than waiting until the document breaks they can set up informative messages why certain packages are not available.

The function is only implemented for packages not for arbitrary files and again it should only be applied if there are good reasons for doing this.⁵⁴

1.7 A sample package for structuring the log output

As an application we provide the package `structuredlog` that adds lines to the `.log` when a file is opened and closed for reading keeping track of nesting level as well. For example, for the current document it adds the lines

```
= (LEVEL 1 START) t1lmr.fd  
= (LEVEL 1 STOP) t1lmr.fd  
= (LEVEL 1 START) supp-pdf.mkii  
= (LEVEL 1 STOP) supp-pdf.mkii  
= (LEVEL 1 START) nameref.sty  
== (LEVEL 2 START) refcount.sty  
== (LEVEL 2 STOP) refcount.sty  
== (LEVEL 2 START) gettitlestring.sty  
== (LEVEL 2 STOP) gettitlestring.sty  
= (LEVEL 1 STOP) nameref.sty  
= (LEVEL 1 START) ltfilehook-doc.out
```

⁵⁴Just to be sure: “I don't like this package by somebody else” is not a good one :-)

```

= (LEVEL 1 STOP) ltfilehook-doc.out
= (LEVEL 1 START) ltfilehook-doc.out
= (LEVEL 1 STOP) ltfilehook-doc.out
= (LEVEL 1 START) ltfilehook-doc.hd
= (LEVEL 1 STOP) ltfilehook-doc.hd
= (LEVEL 1 START) ltfilehook.dtx
== (LEVEL 2 START) otllmr.fd
== (LEVEL 2 STOP) otllmr.fd
== (LEVEL 2 START) omllmm.fd
== (LEVEL 2 STOP) omllmm.fd
== (LEVEL 2 START) omslmsy.fd
== (LEVEL 2 STOP) omslmsy.fd
== (LEVEL 2 START) omxlmex.fd
== (LEVEL 2 STOP) omxlmex.fd
== (LEVEL 2 START) umsa.fd
== (LEVEL 2 STOP) umsa.fd
== (LEVEL 2 START) umsb.fd
== (LEVEL 2 STOP) umsb.fd
== (LEVEL 2 START) tsllmr.fd
== (LEVEL 2 STOP) tsllmr.fd
== (LEVEL 2 START) tllmss.fd
== (LEVEL 2 STOP) tllmss.fd
= (LEVEL 1 STOP) ltfilehook.dtx

```

Thus if you inspect an issue in the .log it is easy to figure out in which file it occurred, simply by searching back for LEVEL and if it is a STOP then remove 1 from the level value and search further for LEVEL with that value which should then be the START level of the file you are in.

2 The Implementation

```

1 <*2kernel>
2 <@@=filehook>

```

2.1 Document and package-level commands

\CurrentFile User-level macros that hold the current file name and file path. These are used internally as well because the code takes care to protect against a possible redefinition of these macros in the loaded file (it's necessary anyway to make hooks work with nested \input).
\CurrentFilePath The versions \...Used hold the *actual* file name and path that is loaded by L^AT_EX, whereas the other two hold the name as requested. They will differ in case there's a file substitution.
\CurrentFileUsed
\CurrentFilePathUsed

```

3 </2kernel>
4 <*2kernel | latexrelease>
5 <latexrelease>\IncludeInRelease{2020/10/01}%
6 <latexrelease>          {\CurrentFile}{Hook management file}%
7 \ExplSyntaxOn
8 \tl_new:N \CurrentFile
9 \tl_new:N \CurrentFilePath
10 \tl_new:N \CurrentFileUsed
11 \tl_new:N \CurrentFilePathUsed

```

```

12 \ExplSyntaxOff
13 </2ekernel | latexrelease>
14 <latexrelease>\EndIncludeInRelease

15 <latexrelease>\IncludeInRelease{0000/00/00}%
16 <latexrelease>          {\CurrentFile}{Hook management file}%
17 <latexrelease>
18 <latexrelease>\let \CurrentFile          \@undefined
19 <latexrelease>\let \CurrentFilePath      \@undefined
20 <latexrelease>\let \CurrentFileUsed     \@undefined
21 <latexrelease>\let \CurrentFilePathUsed \@undefined
22 <latexrelease>
23 <latexrelease>\EndIncludeInRelease
24 <*2ekernel>

```

(End of definition for `\CurrentFile` and others. These functions are documented on page 1101.)

2.2 expl3 helpers

```

25 </2ekernel>
26 <*2ekernel | latexrelease>
27 <latexrelease>\IncludeInRelease{2020/10/01}%
28 <latexrelease>          {\_filehook_file_parse_full_name:nN}{File helpers}%
29 \ExplSyntaxOn

```

`_filehook_file_parse_full_name:nN`
`_filehook_full_name:nn`

A utility macro to trigger expl3’s file-parsing and lookup, and return a normalized representation of the file name. If the queried file doesn’t exist, no normalization takes place. The output of `_filehook_file_parse_full_name:nN` is passed on to the #2—a 3-argument macro that takes the `<path>`, `<base>`, and `<ext>` parts of the file name.

```

30 \cs_new:Npn \_filehook_file_parse_full_name:nN #1
31   {
32     \exp_args:Nf \file_parse_full_name_apply:nN
33       {
34         \exp_args:Nf \_filehook_full_name:nn
35           { \file_full_name:n {#1} } {#1}
36       }
37   }
38 \cs_new:Npn \_filehook_full_name:nn #1 #2
39   {
40     \tl_if_empty:nTF {#1}
41       { \tl_trim_spaces:n {#2} }
42       { \tl_trim_spaces:n {#1} }
43   }

```

(End of definition for `_filehook_file_parse_full_name:nN` and `_filehook_full_name:nn`.)

`_filehook_if_no_extension:nTF`
`_filehook_drop_extension:N`

Some actions depend on whether the file extension was explicitly given, and sometimes the extension has to be removed. The macros below use `_filehook_file_parse_full_name:nN` to split up the file name and either check if `<ext>` (#3) is empty, or discard it.

```

44 \cs_new:Npn \_filehook_if_no_extension:nTF #1
45   {
46     \exp_args:Ne \tl_if_empty:nTF
47       { \file_parse_full_name_apply:nN {#1} \use_iii:nnn }
48   }

```

```

49 \cs_new_protected:Npn \__filehook_drop_extension:N #1
50 {
51   \tl_gset:Nx #1
52   {
53     \exp_args:NV \__filehook_file_parse_full_name:nN #1
54     \__filehook_drop_extension_aux:nnn
55   }
56 }
57 \cs_new:Npn \__filehook_drop_extension_aux:nnn #1 #2 #3
58 { \tl_if_empty:nF {#1} { #1 / } #2 }

```

(End of definition for __filehook_if_no_extension:nTF and __filehook_drop_extension:N.)

\g__filehook_input_file_seq
 \l__filehook_internal_tl
 __filehook_file_push:
 __filehook_file_pop:
 __filehook_file_pop_assign:nnnn

Yet another stack, to keep track of \CurrentFile and \CurrentFilePath with nested \inputs. At the beginning of \InputIfFileExists, the current value of \CurrentFilePath and \CurrentFile is pushed to \g__filehook_input_file_seq, and at the end, it is popped and the value reassigned. Some other places don't use \InputIfFileExists directly (\include) or need \CurrentFile earlier (\@onefilewithoptions), so these are manually used elsewhere as well.

```

59 \tl_new:N \l__filehook_internal_tl
60 \seq_if_exist:NF \g__filehook_input_file_seq
61 { \seq_new:N \g__filehook_input_file_seq }
62 \cs_new_protected:Npn \__filehook_file_push:
63 {
64   \seq_gpush:Nx \g__filehook_input_file_seq
65   {
66     { \CurrentFilePathUsed } { \CurrentFileUsed }
67     { \CurrentFilePath } { \CurrentFile }
68   }
69 }
70 \cs_new_protected:Npn \__filehook_file_pop:
71 {
72   \seq_gpop:NNTF \g__filehook_input_file_seq \l__filehook_internal_tl
73   { \exp_after:wN \__filehook_file_pop_assign:nnnn \l__filehook_internal_tl }
74   {
75     \msg_error:nnn { latex2e } { should-not-happen }
76     { Tried-to-pop-from-an-empty-file-name-stack. }
77   }
78 }
79 \cs_new_protected:Npn \__filehook_file_pop_assign:nnnn #1 #2 #3 #4
80 {
81   \tl_set:Nn \CurrentFilePathUsed {#1}
82   \tl_set:Nn \CurrentFileUsed {#2}
83   \tl_set:Nn \CurrentFilePath {#3}
84   \tl_set:Nn \CurrentFile {#4}
85 }
86 \ExplSyntaxOff

```

(End of definition for \g__filehook_input_file_seq and others.)

```

87 </2kernel | latexrelease>
88 <latexrelease>\EndIncludeInRelease

```

When rolling forward the following expl3 functions may not be defined. If we roll back the code does nothing.

```

89 <latexrelease>\IncludeInRelease{2020/10/01}%
90 <latexrelease>          {\file_parse_full_name_apply:nN}{Roll forward help}%
91 <latexrelease>
92 <latexrelease>\ExplSyntaxOn
93 <latexrelease>\cs_if_exist:NF\file_parse_full_name_apply:nN
94 <latexrelease>{
95 <latexrelease>\cs_new:Npn \file_parse_full_name_apply:nN #1
96 <latexrelease>  {
97 <latexrelease>    \exp_args:Ne \_file_parse_full_name_auxi:nN
98 <latexrelease>      { \_kernel_file_name_sanitize:n {#1} }
99 <latexrelease>  }
100 <latexrelease>\cs_new:Npn \_file_parse_full_name_auxi:nN #1
101 <latexrelease>  {
102 <latexrelease>    \_file_parse_full_name_area:nw { } #1
103 <latexrelease>    / \s__file_stop
104 <latexrelease>  }
105 <latexrelease>\cs_new:Npn \_file_parse_full_name_area:nw #1 #2 / #3 \s__file_stop
106 <latexrelease>  {
107 <latexrelease>    \tl_if_empty:nTF {#3}
108 <latexrelease>      { \_file_parse_full_name_base:nw { } #2 . \s__file_stop {#1} }
109 <latexrelease>      { \_file_parse_full_name_area:nw { #1 / #2 }
110 <latexrelease>        #3 \s__file_stop }
111 <latexrelease>  }
112 <latexrelease>\cs_new:Npn \_file_parse_full_name_base:nw #1 #2 . #3 \s__file_stop
113 <latexrelease>  {
114 <latexrelease>    \tl_if_empty:nTF {#3}
115 <latexrelease>    {
116 <latexrelease>      \tl_if_empty:nTF {#1}
117 <latexrelease>      {
118 <latexrelease>        \tl_if_empty:nTF {#2}
119 <latexrelease>        { \_file_parse_full_name_tidy:nnnN { } { } }
120 <latexrelease>        { \_file_parse_full_name_tidy:nnnN { .#2 } { } }
121 <latexrelease>      }
122 <latexrelease>      { \_file_parse_full_name_tidy:nnnN {#1} { .#2 } }
123 <latexrelease>    }
124 <latexrelease>    { \_file_parse_full_name_base:nw { #1 . #2 }
125 <latexrelease>      #3 \s__file_stop }
126 <latexrelease>  }
127 <latexrelease>\cs_new:Npn \_file_parse_full_name_tidy:nnnN #1 #2 #3 #4
128 <latexrelease>  {
129 <latexrelease>    \exp_args:Nee #4
130 <latexrelease>    {
131 <latexrelease>      \str_if_eq:nnF {#3} { / } { \use_none:n }
132 <latexrelease>      #3 \prg_do_nothing:
133 <latexrelease>    }
134 <latexrelease>    { \use_none:n #1 \prg_do_nothing: }
135 <latexrelease>    {#2}
136 <latexrelease>  }
137 <latexrelease>}
138 <latexrelease>\ExplSyntaxOff
139 <latexrelease>
140 <latexrelease>\EndIncludeInRelease
141 <*2ekernel>
142 <@@=>

```

2.3 Declaring the file-related hooks

These hooks have names with three-parts that start with `file/`, `include/`, `class/` or `package/` and end with `/before` or `/after` (or `/end` in the case of `include/`). They are all generic hooks so will be declared only if code is added to them; this declaration is done for you automatically and, indeed, they should not be declared explicitly.

Those named `.../after` and `include/.../end` are, when code is added, declared as reversed hooks.

2.4 Patching L^AT_EX's `\InputIfFileExists` command

Most of what we have to do is adding `\UseHook` into several L^AT_EX 2_ε core commands, because of some circular dependencies in the kernel we do this only now and not in `ltxfiles`.

```
\InputIfFileExists \InputIfFileExists loads any file if it is available so we have to add the hooks
\@input@file@exists@with@hooks file/before and file/after in the right places. If the file doesn't exist no hooks
\unqu@tefilef@und should be executed.
```

```
143 </2kernel>
144 <latexrelease>\IncludeInRelease{2020/10/01}%
145 <latexrelease>      {\InputIfFileExists}{Hook management (files)}%
146 <*2kernel | latexrelease>

147 \let\InputIfFileExists\@undefined
148 \DeclareRobustCommand \InputIfFileExists[2]{%
149   \IfFileExists{#1}%
150   {%
151     \@expl@@@filehook@file@push@@
152     \@filehook@set@CurrentFile
```

We pre-expand `\@filef@und` so that in case another file is loaded in the true branch of `\InputIfFileExists`, these don't change their value meanwhile. This isn't a worry with `\CurrentFile...` because they are kept in a stack.

```
153   \expandafter\@swaptwoargs\expandafter
154     {\expandafter\@input@file@exists@with@hooks
155       \expandafter{\@filef@und}}%
156     {#2}%
157   \@expl@@@filehook@file@pop@@
158 }%
159 }
160 \def\@input@file@exists@with@hooks#1{%
```

If the file exists then `\CurrentFile` holds its name. But we can't rely on that still being true after the file has been processed. Thus for using the name in the file hooks we need to preserve the name and then restore it for the `file/.../after` hook.

The hook always refers to the file requested by the user. The hook is *always* loaded for `\CurrentFile` which usually is the same as `\CurrentFileUsed`. In the case of a file replacement, the `\CurrentFileUsed` holds the actual file loaded. In any case the file names are normalized so that the hooks work on the real file name, rather than what the user typed in.

`expl3`'s `\file_full_name:n` normalizes the file name (to factor out differences in the `.tex` extension), and then does a file lookup to take into account a possible path from `\l_file_search_path_seq` and `\input@path`. However only the file name and extension

are returned so that file hooks can refer to the file by their name only. The path to the file is returned in `\CurrentFilePath`.

```

161 \edef\reserved@a{%
162   \@expl@@@filehook@file@pop@assign@@nnnn
163   {\CurrentFilePathUsed}%
164   {\CurrentFileUsed}%
165   {\CurrentFilePath}%
166   {\CurrentFile}}%
167 \expandafter\@swaptwoargs\expandafter{\reserved@a}%

```

Before adding to the file list we need to make all (letter) characters catcode 11, because several packages use constructions like

```

\filename@parse{<filename>}
\ifx\filename@ext\@clsextension
...
\fi

```

and that doesn't work if `\filename@ext` is `\detokenized`. Making `\@clsextension` a string doesn't help much because some packages define their own `\<prefix>@someextension` with normal catcodes. This is not entirely correct because packages loaded (somehow) with catcode 12 alphabetic tokens (say, as the result of a `\string` or `\detokenize` command, or from a T_EX string like `\jobname`) will have these character tokens incorrectly turned into letter tokens. This however is rare, so we'll go for the all-letters approach (grepping the packages in T_EX Live didn't bring up any obvious candidate for breaking with this catcode change).

```

168 {\edef\reserved@a{\unqu@tefilef@und#1\@nil}%
169  \@addtofilelist{\string@makeletter\reserved@a}%
170  \UseHook{file/before}%

```

The current file name is available in `\CurrentFile` so we use that in the specific hook.

```

171  \UseHook{file/\CurrentFile/before}%
172  \@input #1% <- trailing space comes from \@filef@und
173  }%

```

And here, `\CurrentFile` is restored (by `\@expl@@@filehook@file@pop@assign@@nnnn`) so we can use it once more.

```

174  \UseHook{file/\CurrentFile/after}%
175  \UseHook{file/after}}
176 \def\unqu@tefilef@und"#1" \@nil{#1}

```

Now declare the non-generic file hooks used above:

```

177 \NewHook{file/before}
178 \NewReversedHook{file/after}
179 \<latexrelease>\EndIncludeInRelease
180 \</2ekernel | latexrelease>

```

Now define `\InputIfFileExists` to input #1 if it seems to exist. Immediately prior to the input, #2 is executed. If the file #1 does not exist, execute '#3'.

```

181 \<latexrelease>\IncludeInRelease{2019/10/01}%
182 \<latexrelease>      {\InputIfFileExists}{Hook management (files)}%
183 \<latexrelease>
184 \<latexrelease>\DeclareRobustCommand \InputIfFileExists[2]{%
185 \<latexrelease>  \IfFileExists{#1}%
186 \<latexrelease>   {%

```



```

187 <latexrelease> \expandafter\@swaptwoargs\expandafter
188 <latexrelease>      {\@filef@und}{#2\@addtofilelist{#1}\@input}}}}
189 <latexrelease>\let\@input@file@exists@with@hooks\@undefined
190 <latexrelease>\let\unqu@tefilef@und\@undefined
191 <latexrelease>\EndIncludeInRelease

192 <latexrelease>\IncludeInRelease{0000/00/00}%
193 <latexrelease>      {\InputIfFileExists}{Hook management (files)}%
194 <latexrelease>\long\def \InputIfFileExists#1#2{%
195 <latexrelease>  \IfFileExists{#1}%
196 <latexrelease>    {#2\@addtofilelist{#1}\@input \@filef@und}}

```

Also undo the internal command as some packages unfortunately test for their existence instead of using `\IfFormatAtLeastTF`.

```

197 <latexrelease>\expandafter\let\csname InputIfFileExists \endcsname\@undefined

198 <latexrelease>\let\@input@file@exists@with@hooks\@undefined
199 <latexrelease>\let\unqu@tefilef@und\@undefined
200 <latexrelease>\EndIncludeInRelease
201 <*2ekernel>

```

(End of definition for `\InputIfFileExists`, `\@input@file@exists@with@hooks`, and `\unqu@tefilef@und`.)

2.5 Declaring a file substitution

```

202 <@=filehook>
203 </2ekernel>
204 <*2ekernel | latexrelease>
205 <latexrelease>\IncludeInRelease{2020/10/01}%
206 <latexrelease>      {\_filehook_subst_add:nn}{Declaring file substitution}%
207 \ExplSyntaxOn

```

`_filehook_subst_add:nn` `_filehook_subst_add:nn` declares a file substitution by doing a (global) definition of the form `\def\@file-subst@{file}{replacement}`. The file names are properly sanitised, and normalized with the same treatment done for the file hooks. That is, a file replacement is declared by using the file name (and extension, if any) only, and the file path should not be given. If a file name is empty it is replaced by `.tex` (the empty csname is used to check that).

```

208 \cs_new_protected:Npn \_filehook_subst_add:nn #1 #2
209   {
210   \group_begin:
211     \cs_set:cpx { } { \exp_not:o { \cs:w\cs_end: } }
212     \int_set:Nn \tex_escapechar:D { -1 }
213     \cs_gset:cpx
214       {
215         @file-subst@
216         \_filehook_subst_file_normalize:Nn \use_ii_iii:nnn {#1}
217       }
218     { \_filehook_subst_file_normalize:Nn \_filehook_file_name_compose:nnn
219       {#2} }
220   \group_end:
221   }
222 \cs_new_protected:Npn \_filehook_subst_remove:n #1
223   {

```

```

224 \group_begin:
225 \cs_set:cpx { } { \exp_not:o { \cs:w\cs_end: } }
226 \int_set:Nn \tex_escapechar:D { -1 }
227 \cs_undefine:c
228 {
229 @file-subst@
230 \__filehook_subst_file_normalize:Nn \use_ii_iii:nnn {#1}
231 }
232 \group_end:
233 }
234 \cs_new:Npn \__filehook_subst_file_normalize:Nn #1 #2
235 {
236 \exp_after:wN \__filehook_subst_empty_name_chk:NN
237 \cs:w \exp_after:wN \cs_end:
238 \cs:w \__filehook_file_parse_full_name:nN {#2} #1 \cs_end:
239 }
240 \cs_new:Npn \__filehook_subst_empty_name_chk:NN #1 #2
241 { \if_meaning:w #1 #2 .tex \else: \token_to_str:N #2 \fi: }

```

(End of definition for `__filehook_subst_add:nn` and others.)

`\use_ii_iii:nnn` A variant of `\use_...` to discard the first of three arguments.

Todo: this should move to `expl3`

```

242 \cs_gset:Npn \use_ii_iii:nnn #1 #2 #3 {#2 #3}

```

(End of definition for `\use_ii_iii:nnn`.)

```

243 \ExplSyntaxOff
244 </2ekernel | latexrelease>
245 <latexrelease>\EndIncludeInRelease
246 <*2ekernel>

```

For two internals we provide L^AT_EX 2_ε names so that we can use them elsewhere in the kernel (and so that they can be used in packages if really needed, e.g., `scrfile`).

```

247 </2ekernel>
248 <*2ekernel | latexrelease>
249 <latexrelease>\IncludeInRelease{2020/10/01}%
250 <latexrelease>          {\declare@file@substitution}{File substitution}%
251 \ExplSyntaxOn
252 \cs_new_eq:NN \declare@file@substitution \__filehook_subst_add:nn
253 \cs_new_eq:NN \undeclare@file@substitution \__filehook_subst_remove:n
254 \ExplSyntaxOff
255 </2ekernel | latexrelease>
256 <latexrelease>\EndIncludeInRelease

```

We are not fully rolling back the file substitutions in case a rollback encounters a package that contains them, but is itself not setup for rollback. So we just bypass them and hope for the best.

```

257 <latexrelease>\IncludeInRelease{0000/00/00}%
258 <latexrelease>          {\declare@file@substitution}{File substitution}%
259 <latexrelease>
260 <latexrelease>\let \declare@file@substitution \@gobbletwo
261 <latexrelease>\let \undeclare@file@substitution \@gobble
262 <latexrelease>

```

263 <latexrelease>\EndIncludeInRelease

264 <*2ekernel>

(End of definition for \declare@file@substitution and \undeclare@file@substitution. These functions are documented on page 1104.)

265 <@@=>

2.6 Selecting a file (\set@curr@file)

\set@curr@file Now we hook into \set@curr@file to resolve a possible file substitution, and add
\set@curr@file@nosearch \@expl@@@filehook@set@curr@file@@nNN at the end, after \@curr@file is set.

\@curr@file A file name is built using \expandafter\string\csname<filename>\endcsname to
\@curr@file@reqd avoid expanding utf8 active characters. The \csname expands the normalization machinery and the routine to resolve a file substitution, returning a control sequence with the same name as the file.

It happens that when <filename> is empty, the generated control sequence is \csname\endcsname, and doing \string on that results in the file csnameendcsname.tex. To guard against that we \ifx-compare the generated control sequence with the empty csname. To do so, \csname\endcsname has to be defined, otherwise it would be equal to \relax and we would have false positives. Here we define \csname\endcsname to expand to itself to avoid it matching the definition of some other control sequence.

266 </2ekernel>

267 <*2ekernel | latexrelease>

268 <latexrelease>\IncludeInRelease{2022/06/01}%

269 <latexrelease> {\set@curr@file}{Setting current file name}%

270 \def\set@curr@file{%

271 \begingroup

272 \set@curr@file@aux}

273 \edef\set@curr@file@nosearch{%

274 \begingroup

275 \let\noexpand\input@path\noexpand\@empty

276 \csname seq_clear:N\endcsname

277 \expandafter\noexpand\csname l_file_search_path_seq\endcsname

278 \noexpand\set@curr@file@aux}

279 \def\set@curr@file@aux#1{%

280 \escapechar\m@ne

281 \let\protect\string

282 \edef~{\string~}%

283 \expandafter\def\csname\expandafter\endcsname

284 \expandafter{\csname\endcsname}%

Two file names are set here: \@curr@file@reqd which is the file requested by the user, and \@curr@file which should be the same, except when we have a file substitution, in which case it holds the actual loaded file. \@curr@file is resolved first, to check if a substitution happens. If it doesn't, \@expl@@@filehook@if@file@replaced@@TF short-cuts and just copies \@curr@file, otherwise the full normalization procedure is executed.

At this stage the file name is parsed and normalized, but if the input doesn't have an extension, the default .tex is *not* added to \@curr@file because for applications other than \input (graphics, for example) the default extension may not be .tex. First check if the input has an extension, then if the input had no extension, call

\@expl@@@filehook@drop@extension@@N. In case of a file substitution, \@curr@file will have an extension.

```
285 \expl@@@filehook@if@no@extension@@nTF{#1}%
286 {\@tempwattrue}{\@tempwafalse}%
287 \@kernel@make@file@csname\@curr@file
288 \expl@@@filehook@resolve@file@subst@@w {#1}%
289 \@expl@@@filehook@if@file@replaced@@TF
290 {\@kernel@make@file@csname\@curr@file@reqd
291 \expl@@@filehook@normalize@file@name@@w{#1}%
292 \if@tempswa \expl@@@filehook@drop@extension@@N\@curr@file@reqd \fi}%
293 {\if@tempswa \expl@@@filehook@drop@extension@@N\@curr@file \fi
294 \global\let\@curr@file@reqd\@curr@file}%
295 \@expl@@@filehook@clear@replacement@flag@@
296 \endgroup}
297 </2kernel | latexrelease>
298 <latexrelease>\EndIncludeInRelease
299 <latexrelease>\IncludeInRelease{2021/06/01}%
300 <latexrelease> {\set@curr@file}{Setting current file name}%
301 <latexrelease>\def\set@curr@file#1{%
302 <latexrelease> \begingroup
303 <latexrelease> \escapechar\m@ne
304 <latexrelease> \let\protect\string
305 <latexrelease> \edef~{\string~}%
306 <latexrelease> \expandafter\def\csname\expandafter\endcsname
307 <latexrelease> \expandafter{\csname\endcsname}%
308 <latexrelease> \@expl@@@filehook@if@no@extension@@nTF{#1}%
309 <latexrelease> {\@tempwattrue}{\@tempwafalse}%
310 <latexrelease> \@kernel@make@file@csname\@curr@file
311 <latexrelease> \expl@@@filehook@resolve@file@subst@@w {#1}%
312 <latexrelease> \@expl@@@filehook@if@file@replaced@@TF
313 <latexrelease> {\@kernel@make@file@csname\@curr@file@reqd
314 <latexrelease> \expl@@@filehook@normalize@file@name@@w{#1}%
315 <latexrelease> \if@tempswa \expl@@@filehook@drop@extension@@N\@curr@file@reqd \fi}%
316 <latexrelease> {\if@tempswa \expl@@@filehook@drop@extension@@N\@curr@file \fi
317 <latexrelease> \global\let\@curr@file@reqd\@curr@file}%
318 <latexrelease> \@expl@@@filehook@clear@replacement@flag@@
319 <latexrelease> \endgroup}
320 <latexrelease>\let\set@curr@file@nosearch\undefined
321 <latexrelease>\EndIncludeInRelease
322 <latexrelease>\IncludeInRelease{2020/10/01}%
323 <latexrelease> {\set@curr@file}{Setting current file name}%
324 <latexrelease>\def\set@curr@file#1{%
325 <latexrelease> \begingroup
326 <latexrelease> \escapechar\m@ne
327 <latexrelease> \expandafter\def\csname\expandafter\endcsname
328 <latexrelease> \expandafter{\csname\endcsname}%
329 <latexrelease> \@expl@@@filehook@if@no@extension@@nTF{#1}%
330 <latexrelease> {\@tempwattrue}{\@tempwafalse}%
331 <latexrelease> \@kernel@make@file@csname\@curr@file
332 <latexrelease> \expl@@@filehook@resolve@file@subst@@w {#1}%
333 <latexrelease> \@expl@@@filehook@if@file@replaced@@TF
334 <latexrelease> {\@kernel@make@file@csname\@curr@file@reqd
335 <latexrelease> \expl@@@filehook@normalize@file@name@@w{#1}%
```

```

336 <latexrelease> \if@tempswa \@expl@@@filehook@drop@extension@@N\@curr@file@reqd \fi}%
337 <latexrelease> {\if@tempswa \@expl@@@filehook@drop@extension@@N\@curr@file \fi
338 <latexrelease> \global\let\@curr@file@reqd\@curr@file}%
339 <latexrelease> \@expl@@@filehook@clear@replacement@flag@@
340 <latexrelease> \endgroup}
341 <latexrelease>\let\set@curr@file@nosearch\@undefined
342 <latexrelease>\EndIncludeInRelease

343 <latexrelease>\IncludeInRelease{2019/10/01}%
344 <latexrelease> {\set@curr@file}{Setting current file name}%
345 <latexrelease>\def\set@curr@file#1{%
346 <latexrelease> \begingroup
347 <latexrelease> \escapechar\m@ne
348 <latexrelease> \xdef\@curr@file{%
349 <latexrelease> \expandafter\expandafter\expandafter\unquote@name
350 <latexrelease> \expandafter\expandafter\expandafter{%
351 <latexrelease> \expandafter\string
352 <latexrelease> \csname\@firstofone#1\@empty\endcsname}}%
353 <latexrelease> \endgroup
354 <latexrelease>}
355 <latexrelease>\let\set@curr@file@nosearch\@undefined
356 <latexrelease>\EndIncludeInRelease

357 <latexrelease>\IncludeInRelease{0000/00/00}%
358 <latexrelease> {\set@curr@file}{Setting current file name}%
359 <latexrelease>\let\set@curr@file\@undefined
360 <latexrelease>\let\set@curr@file@nosearch\@undefined
361 <latexrelease>\EndIncludeInRelease
362 <*2kernel>

```

(End of definition for \set@curr@file and others.)

```

\@filehook@set@CurrentFile
\@kernel@make@file@csname
\@set@curr@file@aux

```

Todo: This should get internalized using @expl@ names

```

363 </2kernel>
364 <*2kernel | latexrelease>
365 <latexrelease>\IncludeInRelease{2020/10/01}%
366 <latexrelease> {\@kernel@make@file@csname}{Make file csname}%
367 \def\@kernel@make@file@csname#1#2#3{%
368 \xdef#1{\expandafter\@set@curr@file@aux
369 \csname\expandafter#2\@firstofone#3\@nil\endcsname}}

```

This auxiliary compares \<filename> with \csname\endcsname to check if the empty .tex file was requested.

```

370 \long\def\@set@curr@file@aux#1{%
371 \expandafter\ifx\csname\endcsname#1%
372 .tex\else\string#1\fi}

```

Then we call \@expl@@@filehook@set@curr@file@@nNN once for \@curr@file to set \CurrentFile(Path)Used and once for \@curr@file@reqd to set \CurrentFile(Path). Here too the slower route is only used if a substitution happened, but here \@expl@@@filehook@if@file@replaced@@TF can't be used because the flag is reset at the \endgroup above, so we check if \@curr@file and \@curr@file@reqd differ. This macro is issued separate from \set@curr@file because it changes \CurrentFile, and side-effects would quickly get out of control.

```

373 \def\@filehook@set@CurrentFile{%

```

```

374 \expl@@@filehook@set@curr@file@@nNN{\@curr@file}%
375 \CurrentFileUsed\CurrentFilePathUsed
376 \ifx\@curr@file@reqd\@curr@file
377 \let\CurrentFile\CurrentFileUsed
378 \let\CurrentFilePath\CurrentFilePathUsed
379 \else
380 \expl@@@filehook@set@curr@file@@nNN{\@curr@file@reqd}%
381 \CurrentFile\CurrentFilePath
382 \fi}
383 </2ekernel | latexrelease>
384 <latexrelease>\EndIncludeInRelease
385 <*2ekernel>

(End of definition for \@filehook@set@CurrentFile, \@kernel@make@file@csname, and
\@set@curr@file@aux.)

386 <@@=filehook>

```

```

\__filehook_set_curr_file:nNN
\__filehook_set_curr_file_assign:nnnNN

```

When inputting a file, `\set@curr@file` does a file lookup (in `\input@path` and `\l_file_search_path_seq`) and returns the actual file name (`(base)` plus `(ext)`) in `\CurrentFileUsed`, and in case there's a file substitution, the requested file in `\CurrentFile` (otherwise both are the same). Only the base and extension are returned, regardless of the input (both `path/to/file.tex` and `file.tex` end up as `file.tex` in `\CurrentFile`). The path is returned in `\CurrentFilePath`, in case it's needed.

```

387 </2ekernel>
388 <*2ekernel | latexrelease>
389 <latexrelease>\IncludeInRelease{2020/10/01}%
390 <latexrelease> \__filehook_set_curr_file:nNN}{Set curr file}%
391 \ExplSyntaxOn
392 \cs_new_protected:Npn \__filehook_set_curr_file:nNN #1
393 {
394 \exp_args:Nf \__filehook_file_parse_full_name:nN {#1}
395 \__filehook_set_curr_file_assign:nnnNN
396 }
397 \cs_new_protected:Npn \__filehook_set_curr_file_assign:nnnNN #1 #2 #3 #4 #5
398 {
399 \str_set:Nn #5 {#1}
400 \str_set:Nn #4 {#2#3}
401 }
402 \ExplSyntaxOff
403 </2ekernel | latexrelease>
404 <latexrelease>\EndIncludeInRelease
405 <*2ekernel>

(End of definition for \__filehook_set_curr_file:nNN and
\__filehook_set_curr_file_assign:nnnNN.)

```

2.7 Replacing a file and detecting loops

```

\__filehook_resolve_file_subst:w
\__filehook_normalize_file_name:w
\__filehook_file_name_compose:nnm

```

Start by sanitizing the file with `__filehook_file_parse_full_name:nN` then do `__filehook_file_subst_begin:nnn{<path>}{<name>}{<ext>}`.

```

406 </2ekernel>
407 <*2ekernel | latexrelease>
408 <latexrelease>\IncludeInRelease{2020/10/01}%

```

```

409 <latexrelease>      {\_filehook_resolve_file_subst:w}{Replace files detect loops}%
410 \ExplSyntaxOn
411 \cs_new:Npn \_filehook_resolve_file_subst:w #1 \@nil
412   { \_filehook_file_parse_full_name:nN {#1} \_filehook_file_subst_begin:nnn }
413 \cs_new:Npn \_filehook_normalize_file_name:w #1 \@nil
414   { \_filehook_file_parse_full_name:nN {#1} \_filehook_file_name_compose:nnn }
415 \cs_new:Npn \_filehook_file_name_compose:nnn #1 #2 #3
416   { \tl_if_empty:nF {#1} { #1 / } #2#3 }

```

```

\_filehook_file_replaced
\_filehook_if_file_replaced:TF
\_filehook_clear_replacement_flag:

```

Since the file replacement is done expandably in a `\csname`, use a flag to remember if a substitution happened. We use this in `\set@curr@file` to short-circuit some of it in case no substitution happened (by far the most common case, so it's worth optimizing). The flag raised during the file substitution algorithm must be explicitly cleared after the `_filehook_if_file_replaced:TF` conditional is no longer needed, otherwise further uses of `_filehook_if_file_replaced:TF` will wrongly return true.

```

417 \flag_new:n { \_filehook_file_replaced }
418 \cs_new:Npn \_filehook_if_file_replaced:TF #1 #2
419   { \flag_if_raised:nTF { \_filehook_file_replaced } {#1} {#2} }
420 \cs_new_protected:Npn \_filehook_clear_replacement_flag:
421   { \flag_clear:n { \_filehook_file_replaced } }

```

```

\_filehook_file_subst_begin:nnn

```

First off, start by checking if the current file (`\langle name \rangle + \langle ext \rangle`) has a declared substitution. If not, then just put that as the name (including a possible `\langle path \rangle` in this case): this is the default case with no substitutions, so it's the first to be checked. The auxiliary `_filehook_file_subst_tortoise_hare:nn` sees that there's no replacement for `#2#3` and does nothing else.

```

422 \cs_new:Npn \_filehook_file_subst_begin:nnn #1 #2 #3
423   {
424     \_filehook_file_subst_tortoise_hare:nn { #2#3 } { #2#3 }
425     { \_filehook_file_name_compose:nnn {#1} {#2} {#3} }
426   }
427 \ExplSyntaxOff
428 </2ekernel | latexrelease>
429 <latexrelease>\EndIncludeInRelease
430 <*2ekernel>

```

2.7.1 The Tortoise and Hare algorithm

```

\_filehook_file_subst_tortoise_hare:nn
\_filehook_file_subst_loop:NN
\_filehook_file_subst_loop:cc

```

If there is a substitution (`\langle true \rangle` in the first `\cs_if_exist:cTF` below), then first check if there is no substitution down the line: this should be the second most common case, of one file replaced by another. In that case just leave the substitution there and the job is done. If any substitution happens, then the `\flag _filehook_file_replaced` is raised (conditionally, because checking if a flag is raised is much faster than raising it over and over again).

If, however there are more substitutions, then we need to check for a possible loop in the substitutions, which would otherwise put \TeX in an infinite loop if just an exhaustive expansion was used.

To detect a loop, the *Tortoise and Hare* algorithm is used. The name of the algorithm is an analogy to Aesop's fable, in which the Hare outruns a Tortoise. The two pointers here are the csnames which contains each file replacement, both of which start at the position zero, which is the file requested. In the inner part of the macro below, `_filehook_file_subst_loop:cc` is called with `\@file-subst@<file>` and

`\@file-subst@\@file-subst@<file>`; that is, the substitution of `<file>` and the substitution of that substitution: the Tortoise walks one step while the Hare walks two.

Within `__filehook_file_subst_loop:NN` the two substitutions are compared, and if they lead to the same file it means that there is a loop in the substitutions. If there's no loop, `__filehook_file_subst_tortoise_hare:nn` is called again with the Tortoise at position 1 and the hare at 2. Again, the substitutions are checked ahead of the Hare pointer to check that it won't run too far; in case there is no loop in the declarations, eventually one of the `\cs_if_exist:cTF` below will go `<false>` and the algorithm will end; otherwise it will run until the Hare reaches the same spot as the tortoise and a loop is detected.

```

431 </2ekernel>
432 <*2ekernel|latexrelease>
433 <latexrelease>\IncludeInRelease{2020/10/01}%
434 <latexrelease> {\__filehook_file_subst_tortoise_hare:nn}{Tortoise and Hare}%
435 \ExplSyntaxOn
436 \cs_new:Npn \__filehook_file_subst_tortoise_hare:nn #1 #2 #3
437 {
438   \cs_if_exist:cTF { @file-subst@ #2 }
439   {
440     \flag_if_raised:nF { __filehook_file_replaced }
441     { \flag_raise:n { __filehook_file_replaced } }
442     \cs_if_exist:cTF { @file-subst@ \use:c { @file-subst@ #2 } }
443     {
444       \__filehook_file_subst_loop:cc
445       { @file-subst@ #1 }
446       { @file-subst@ \use:c { @file-subst@ #2 } }
447     }
448     { \use:c { @file-subst@ #2 } }
449   }
450   { #3 }
451 }

```

This is just an auxiliary to check if a loop was found, and continue the algorithm otherwise. If a loop is found, the `.tex` file is used as fallback and `__filehook_file_subst_cycle_error:cN` is called to report the error.

```

452 \cs_new:Npn \__filehook_file_subst_loop:NN #1 #2
453 {
454   \token_if_eq_meaning:NNTF #1 #2
455   {
456     .tex
457     \__filehook_file_subst_cycle_error:cN { @file-subst@ #1 } #1
458   }
459   { \__filehook_file_subst_tortoise_hare:nn {#1} {#2} {#2} }
460 }
461 \cs_generate_variant:Nn \__filehook_file_subst_loop:NN { cc }

```

Showing this type of error expandably is tricky, as we have a very limited amount of characters to show and a potentially large list. As a work around, several errors are printed, each showing one step of the loop, until all the error messages combined show the loop.

```

462 \cs_new:Npn \__filehook_file_subst_cycle_error:NN #1 #2
463 {
464   \msg_expandable_error:nmff { latex2e } { file-cycle }

```



```

465     {#1} { \use:c { @file-subst@ #1 } }
466     \token_if_eq_meaning:NNF #1 #2
467     { \_filehook_file_subst_cycle_error:cN { @file-subst@ #1 } #2 }
468   }
469 \cs_generate_variant:Nn \_filehook_file_subst_cycle_error:NN { c }
    And the error message:
470 \msg_new:nnn { latex2e } { file-cycle }
471   { File~loop!~#1~replaced~by~#2... }

(End of definition for \_filehook_resolve_file_subst:w and others.)

472 \ExplSyntaxOff
473 </2ekernel | latexrelease>
474 <latexrelease>\EndIncludeInRelease
475 <*2ekernel>
476 <@@=>

```

2.8 Preventing a package from loading

We support the use case of preventing a package from loading but not any other type of files (e.g., classes).

`\disable@package@load` defines `\@pkg-disable@<package>` to expand to some code #2 instead of loading the package.

```

\disable@package@load
\reenable@package@load
\@disable@packageload@do
477 </2ekernel>
478 <*2ekernel | latexrelease>
479 <latexrelease>\IncludeInRelease{2020/10/01}%
480 <latexrelease>          {\disable@package@load}{Disable packages}%
481 \def\disable@package@load#1#2{%
482   \global\namedef{@pkg-disable@#1.\@pkgextension}{#2}}

```

Here we check if a control sequence named `\@pkg-disable@<name>.sty` is defined, and if so don't use the package loading code #2, but use the replacement code stored in that control sequence, write something to the log, and then prevent `\@onefilewithoptions` from sanity-checking the requested package date (the `\expandafter` here triggers one in `\@onefilewithoptions` that ends a conditional there, and the `\@gobbletwo` removes the date checking code from the input stream).

```

483 \def\@disable@packageload@do#1#2{%
484   \ifundefined{@pkg-disable@#1}%
485     {#2}%
486     {\@nameuse{@pkg-disable@#1}%
487       \@latex@info{Package '#1' has been disabled.%
488         \MessageBreak Load request ignored}%
489       \expandafter\@gobbletwo}}

```

`\reenable@package@load` undefines `\@pkg-disable@<package>` to reallow loading a package.

```

490 \def\reenable@package@load#1{%
491   \global\expandafter\let
492   \csname @pkg-disable@#1.\@pkgextension \endcsname \@undefined}

```

```

493 </2ekernel | latexrelease>
494 <latexrelease>\EndIncludeInRelease
495 <latexrelease>\IncludeInRelease{0000/00/00}%
496 <latexrelease>      {\disable@package@load}{Disable packages}%
497 <latexrelease>
498 <latexrelease>\let\disable@package@load  \@undefined
499 <latexrelease>\let\@disable@packageload@do\@undefined
500 <latexrelease>\let\reenable@package@load  \@undefined
501 <latexrelease>\EndIncludeInRelease
502 <*2ekernel>

```

(End of definition for \disable@package@load, \reenable@package@load, and \@disable@packageload@do. These functions are documented on page 1104.)

2.9 High-level interfaces for L^AT_EX

None so far and the general feeling for now is that the hooks are enough. Packages like filehook, etc., may use them to set up their interfaces (samples are given below) but for the now the kernel will not provide any.

2.10 Internal commands needed elsewhere

Here we set up a few horrible (but consistent) L^AT_EX 2_ε names to allow for internal commands to be used outside this module (and in parts that still use L^AT_EX 2_ε syntax. We have to unset the @@ since we want double “at” sign in place of double underscores.

```

503 <@@= >
504 </2ekernel>
505 <*2ekernel | latexrelease>
506 <latexrelease>\IncludeInRelease{2020/10/01}%
507 <latexrelease>      {\expl@@@filehook@if@no@extension@@nTF}{2e tmp interfaces}%
508 \ExplSyntaxOn
509 \cs_new_eq:NN \expl@@@filehook@if@no@extension@@nTF
510                \__filehook_if_no_extension:nTF
511 \cs_new_eq:NN \expl@@@filehook@set@curr@file@@nNN
512                \__filehook_set_curr_file:nNN
513 \cs_new_eq:NN \expl@@@filehook@resolve@file@subst@@w
514                \__filehook_resolve_file_subst:w
515 \cs_new_eq:NN \expl@@@filehook@normalize@file@name@@w
516                \__filehook_normalize_file_name:w
517 \cs_new_eq:NN \expl@@@filehook@if@file@replaced@@TF
518                \__filehook_if_file_replaced:TF
519 \cs_new_eq:NN \expl@@@filehook@clear@replacement@flag@@
520                \__filehook_clear_replacement_flag:
521 \cs_new_eq:NN \expl@@@filehook@drop@extension@@N
522                \__filehook_drop_extension:N
523 \cs_new_eq:NN \expl@@@filehook@file@push@@
524                \__filehook_file_push:
525 \cs_new_eq:NN \expl@@@filehook@file@pop@@
526                \__filehook_file_pop:

```

```

527 \cs_new_eq:NN \@expl@@@filehook@file@pop@assign@@nnnn
528         \__filehook_file_pop_assign:nnnn
529 \ExplSyntaxOff

```

This one specifically has to be undefined because it is left over in the input stream from `\InputIfFileExists` and executed when `latexrelease` is loaded. It cannot be `\let` to `\undefined` otherwise it would error as well, so it is `\let` to `\relax` to be silently ignored when loading `\latexrelease`.

```

530 </2ekernel | latexrelease>
531 <latexrelease>\EndIncludeInRelease
532 <latexrelease>
533 <latexrelease>\IncludeInRelease{0000/00/00}%
534 <latexrelease>    {\@expl@@@filehook@if@no@extension@@nTF}{2e tmp interfaces}%
535 <latexrelease>\let\@expl@@@filehook@file@pop@@\relax
536 <latexrelease>\EndIncludeInRelease
537 <*2ekernel>

```

This ends the kernel code in this file.

```

538 </2ekernel>

```

3 A sample package for structuring the log output

```

539 <*structuredlog>
540 <@@=filehook>
541 \ProvidesExplPackage
542     {structuredlog}{\ltfilehookdate}{\ltfilehookversion}
543     {Structuring the TeX transcript file}

```

`\g_filehook_nesting_level_int` Stores the current package nesting level.

```

544 \int_new:N \g_filehook_nesting_level_int

```

Initialise the counter with the number of files in the `\@currnamestack` (the number of items divided by 3) minus one, because this package is skipped when printing to the log.

```

545 \int_gset:Nn \g_filehook_nesting_level_int
546     { ( \tl_count:N \@currnamestack ) / 3 - 1 }

```

(End of definition for `\g_filehook_nesting_level_int`.)

`__filehook_log_file_record:n` This macro is responsible for increasing and decreasing the file nesting level, as well as printing to the log. The argument is either `STOPTART` or `STOP` and the action it takes on the nesting integer depends on that.

```

547 \cs_new_protected:Npn \__filehook_log_file_record:n #1
548     {
549     \str_if_eq:nnT {#1} {START} { \int_gincr:N \g_filehook_nesting_level_int }
550     \iow_term:x
551     {
552     \prg_replicate:nn { \g_filehook_nesting_level_int } { = } ~
553     ( LEVEL ~ \int_use:N \g_filehook_nesting_level_int \c_space_tl #1 ) ~
554     \CurrentFileUsed

```

If there was a file replacement, show that as well:

```

555     \str_if_eq:NNF \CurrentFileUsed \CurrentFile
556     { ~ ( \CurrentFile \c_space_tl requested ) }
557     \iow_newline:
558   }
559   \str_if_eq:nnT {#1} {STOP} { \int_gdecr:N \g__filehook_nesting_level_int }
560 }

```

Now just hook the macro above in the generic file/before...

```

561 \AddToHook{file/before}{ \__filehook_log_file_record:n { START } }

```

...and file/after hooks. We don't want to install the file/after hook immediately, because that would mean it is the first time executed when the package finishes. We therefore put the declaration inside `\AddToHookNext` so that it gets only installed when we have left this package.

```

562 \AddToHookNext{file/after}
563 { \AddToHook{file/after}{ \__filehook_log_file_record:n { STOP } } }

```

(End of definition for `__filehook_log_file_record:n`.)

```

564 <@@=
565 </structuredlog>

```

4 Package emulations

4.1 Package `atveryend` emulation

With the new hook management and the hooks in `\enddocument` all of `atveryend` is taken care of. We can make an emulation only here after the substitution functionality is available:

```

566 <*2kernel>
567 \declare@file@substitution{atveryend.sty}{atveryend-ltx.sty}
568 </2kernel>

```

Here is the package file we point to:

```

569 <*atveryend-ltx>
570 \ProvidesPackage{atveryend-ltx}
571 [2020/08/19 v1.0a
572 Emulation of the original atveryend package^^Jwith kernel methods]

```

Here are new definitions for its interfaces now pointing to the hooks in `\enddocument`

```

573 \newcommand\AfterLastShipout {\AddToHook{enddocument/afterlastpage}}
574 \newcommand\AtVeryEndDocument {\AddToHook{enddocument/afteraux}}

```

Next one is a bit of a fake, but the result should normally be as expected. If not, one needs to add a rule to sort the code chunks in `enddocument/info`.

```

575 \newcommand\AtEndAfterFileList{\AddToHook{enddocument/info}}
576 \newcommand\AtVeryVeryEnd {\AddToHook{enddocument/end}}

```

`\BeforeClearDocument` This one is the only one we don't implement or rather don't have a dedicated hook in the code.

```

577 \ExplSyntaxOn
578 \newcommand\BeforeClearDocument[1]
579 { \AtEndDocument{#1}
580   \atveryend@DEPRECATED{BeforeClearDocument \tl_to_str:n{#1}}
581 }

```

```
582 \cs_new:Npn\atveryend@DEPRECATED #1
583   {\iow_term:x{=====~DEPRECATED~USAGE~#1~=====}}
584 \ExplSyntaxOff

(End of definition for \BeforeClearDocument.)

585 \atveryend-ltx)
```

File 53

ltshipout.dtx

1 Introduction

The code provides an interface to the `\shipout` primitive of \TeX which is called when a finished pages is finally “shipped out” to the target output file, e.g., the `.dvi` or `.pdf` file. A good portion of the code is based on ideas by Heiko Oberdiek implemented in his packages `atbegshi` and `atenddvi` even though the interfaces are somewhat different.⁵⁵

1.1 Overloading the `\shipout` primitive

`\shipout` With this implementation \TeX 's `shipout` primitive is no longer available for direct use. Instead `\shipout` is running some (complicated) code that picks up the box to be shipped out regardless of how that is done, i.e., as a constructed `\vbox` or `\hbox` or as a box register.

It then stores it in a named box register. This box can then be manipulated through a set of hooks after which it is shipped out for real.

Each shipout that actually happens (i.e., where the material is not discarded for one or the other reason) is recorded and the total number is available in a readonly variable and in a \LaTeX counter.

`\RawShipout` This command implements a simplified shipout that bypasses the foreground and background hooks, e.g., only `shipout/firstpage` and `shipout/lastpage` are executed and the total shipout counters are incremented.

The command doesn't use `\ShipoutBox` but its own private box register so that it can be used inside of shipout hooks to do some additional shipouts while already in the output routine with the current page being stored in `\ShipoutBox`. It does have access to `\ShipoutBox` if it is used in `shipout/before` (or `shipout/after`) and can use its content.

It is safe to use it in `shipout/before` or `shipout/after` but not necessarily in the other `shipout/...` hooks as they are intended for special processing.

⁵⁵Heiko's interfaces are emulated by the kernel code, if a document requests his packages, so older documents will continue to work.

`\ShipoutBox` This box register is called `\ShipoutBox` (alternatively available via the L3 name `\l_`
`\l_shipout_box` `shipout_box`).

This box is a “local” box and assignments to it should be done only locally. Global assignments (as done by some packages with older code where this box is known as 255) may work but they are conceptually wrong and may result in errors under certain circumstances.

During the execution of `shipout/before` this box contains the accumulated material for the page, but not yet any material added by other shipout hooks. During execution of `shipout/after`, i.e., after the shipout has happened, the box also contains any background or foreground material.

Material from the hooks `shipout/firstpage` or `shipout/lastpage` is not included (but only used during the actual shipout) to facilitate reuse of the box data (e.g., `shipout/firstpage` material should never be added to a later page of the output).

`\l_shipout_box_ht_dim`
`\l_shipout_box_dp_dim`
`\l_shipout_box_wd_dim`
`\l_shipout_box_ht_plus_dp_dim`

The shipout box dimensions are available in the L3 registers `\l_shipout_box_ht_dim`, etc. (there are no L^AT_EX 2_ε names).⁵⁶ These variables can be used inside the hook code for `shipout/before`, `shipout/foreground` and `shipout/background` if needed.

1.2 Provided hooks

`shipout/before` The code for `\shipout` offers a number of hooks into which packages (or the user) can
`shipout/after` add code to support different use cases. These are:
`shipout/foreground`

`shipout/background` **shipout/before** This hook is executed after the finished page has been stored in
`shipout/firstpage` `\ShipoutBox / \l_shipout_box`. It can be used to alter that box content or
`shipout/lastpage` to discard it completely (see `\DiscardShipoutBox` below).

You can use `\RawShipout` inside this hook for special use cases. It can make use of `\ShipoutBox` (which doesn’t yet include the background and foreground material).

Note: It is not possible (or say advisable) to try and use this hook to typeset material with the intention to return it to main vertical list, it will go wrong and give unexpected results in many cases—for starters it will appear after the current page not before or it will vanish or the vertical spacing will be wrong!

shipout/background This hook adds a picture environment into the background of the page with the (0,0) coordinate in the top-left corner using a `\unitlength` of `1pt`.

It should therefore only receive `\put` commands or other commands suitable in a `picture` environment and the vertical coordinate values would normally be negative.

⁵⁶Might need changing, but HO’s version as strings is not really helpful I think).

Technically this is implemented by adding a zero-sized `\hbox` as the very first item into the `\ShipoutBox` containing that `picture` environment. Thus the rest of the box content will overprint what ever is typeset by that hook.

shipout/foreground This hook adds a `picture` environment into the foreground of the page with the $(0,0)$ coordinate in the top-left corner using a `\unitlength` of `1pt`. Technically this is implemented by adding a zero-sized `\hbox` as the very last item into the `\ShipoutBox` and raising it up so that it still has its $(0,0)$ point in the top-left corner. But being placed after the main box content it will be typeset later and thus overprints it (i.e., is in the foreground).

shipout This hook is executed after foreground and/or background material has been added, i.e., just in front of the actual shipout operation. Its purpose is to allow manipulation of the finalized box (stored in `\ShipoutBox`) with the extra material also in place (which is not yet the case in `shipout/before`).

It cannot be used to cancel the shipout operation via `\DiscardShipoutBox` (that has to happen in `shipout/before`, if desired!

shipout/firstpage The material from this hook is executed only once at the very beginning of the first output page that is shipped out (i.e., not discarded at the last minute). It should only contain `\special` or similar commands needed to direct post processors handling the `.dvi` or `.pdf` output.⁵⁷

This hook is added to the very first page regardless of how it is shipped out (i.e., with `\shipout` or `\RawShipout`).

shipout/lastpage The corresponding hook to add `\specials` at the very end of the output file. It is only executed on the very last page of the output file — or rather on the page that `LATEX` believes is the last one. Again it is executed regardless of the shipout method.

It may not be possible for `LATEX` to correctly determine which page is the last one without several reruns. If this happens and the hook is non-empty then `LATEX` will add an extra page to place the material and also request a rerun to get the correct placement sorted out.

shipout/after This hook is executed after a shipout has happened. If the shipout box is discarded this hook is not looked at.

You can use `\RawShipout` inside this hook for special use cases and the main `\ShipoutBox` is still available at this point (but in contrast to `shipout/before` it now includes the background and foreground material).

Note: Just like `shipout/before` this hook is not meant to be used for adding typeset material back to the main vertical list—it might vanish or the vertical spacing will be wrong!

As mentioned above the hook `shipout/before` is executed first and can manipulate the prepared shipout box stored in `\ShipoutBox` or set things up for use in `\write` during the actual shipout. It is even run if there was a `\DiscardShipoutBox` request in the document.

The other hooks (except `shipout` and `shipout/after`) are added inside `hboxes` to the box being shipped out in the following order:

⁵⁷In `LATEX 2ε` that was already existing, but implemented using a box register with the name `\@begindvibox`.

<code>shipout/firstpage</code>	only on the first page
<code>shipout/background</code>	
<code><boxed content of \ShipoutBox></code>	
<code>shipout/foreground</code>	
<code>shipout/lastpage</code>	only on the last page

If any of the hooks has no code then the corresponding box is added at that point.

Once the (page) box has got the above extra content it can again be manipulated using the `shipout` hook and then is shipped out for real.

Once the (page) box has been shipped out the `shipout/after` hook is called (while you are still inside the output routine). It is not called if the shipout box was discarded.

In a document that doesn't produce pages, e.g., only makes `\typeouts`, none of the hooks are ever executed (as there is no `\shipout`) not even the `shipout/lastpage` hook.

If `\RawShipout` is used instead of `\shipout` then only the hooks `shipout/firstpage` and `shipout/lastpage` are executed (on the first or last page), all others are bypassed.

1.3 Legacy L^AT_EX commands

`\AtBeginDvi` `\AtBeginDvi` `{(code)}`

`\AtEndDvi`

`\AtBeginDvi` is the existing L^AT_EX 2_ε interface to fill the `shipout/firstpage` hook. This is not really a good name as it is not just supporting `.dvi` but also `.pdf` output or `.xdv`.

`\AtEndDvi` is the counterpart that was not available in the kernel but only through the package `atenddvi`. It fills the `shipout/lastpage` hook.

Neither interface can set a code label but uses the current default label.

As these two wrappers have been available for a long time we continue offering them (but not enhancing them, e.g., by providing support for code labels).

For new code we strongly suggest using the high-level hook management commands directly instead of “randomly-named” wrappers. This will lead to code that is easier to understand and to maintain and it also allows you to set code labels if needed.

For this reason we do not provide any other “new” wrapper commands for the above hooks in the kernel, but only keep the existing ones for backward compatibility.

1.4 Special commands for use inside the hooks

`\DiscardShipoutBox` `\AddToHookNext {shipout/before} {...\DiscardShipoutBox...}`
`\shipout_discard:`

The `\DiscardShipoutBox` declaration (L3 name `\shipout_discard:`) requests that on the next shipout the page box is thrown away instead of being shipped to the `.dvi` or `.pdf` file.

Typical applications wouldn't do this unconditionally, but have some processing logic that decides to use or not to use the page.

Note that if this declaration is used directly in the document it may depend on the placement to which page it applies, given that `LATEX` output routine is called in an asynchronous manner! Thus normally one would use this only as part of the `shipout/before` code.

Todo: Once we have a new mark mechanism available we can improve on that and make sure that the declaration applies to the page that contains it — not done (yet)

`\DiscardShipoutBox` cannot be used in any of the `shipout/...` hooks other than `shipout/before`.

In the `atbegshi` package there are a number of additional commands for use inside the `shipout/before` hook. They should normally not be needed any more as one can instead simply add code to the hooks `shipout/before`, `shipout`, `shipout/background` or `shipout/foreground`.⁵⁸ If `atbegshi` gets loaded then those commands become available as public functions with their original names as given below.

1.5 Provided Lua_T_E_X callbacks

`pre_shipout_filter` Under Lua_T_E_X the `pre_shipout_filter` Lua callback is provided which gets called directly after the `shipout` hook, immediately before the `shipout` primitive gets invoked. The signature is

```
function(<node> head)
  return true
end
```

The `head` is the list node corresponding to the box to be shipped out. The return value should always be `true`.

⁵⁸If that assumption turns out to be wrong it would be trivial to change them to public functions (right now they are private).

1.6 Information counters

<code>\ReadonlyShipoutCounter</code>	<code>\ifnum\ReadonlyShipoutCounter=...</code>
<code>\g_shipout_readonly_int</code>	<code>\int_use:N \g_shipout_readonly_int % expl3 usage</code>

This integer holds the number of pages shipped out up to now (including the one to be shipped out when inside the output routine). More precisely, it is incremented only after it is clear that a page will be shipped out, i.e., after the `shipout/before` hook (because that might discard the page)! In contrast `shipout/after` sees the incremented value.

Just like with the `page` counter its value is only accurate within the output routine. In the body of the document it may be off by one as the output routine is called asynchronously!

Also important: it *must not* be set, only read. There are no provisions to prevent that restriction, but if you manipulate it, chaos will be the result. To emphasize this fact it is not provided as a \LaTeX counter but as a \TeX counter (i.e., a command), so `\Alph{\ReadonlyShipoutCounter}` etc, would not work.

<code>totalpages</code>	<code>\arabic{totalpages}</code>
<code>\g_shipout_totalpages_int</code>	<code>\int_use:N \g_shipout_totalpage_int % expl3 usage</code>

In contrast to `\ReadonlyShipoutCounter`, the `totalpages` counter is a \LaTeX counter and incremented for each shipout attempt including those pages that are discarded for one or the other reason. Again `shipout/before` sees the counter before it is incremented. In contrast `shipout/after` sees the incremented value.

Furthermore, while it is incremented for each page, its value is never used by \LaTeX . It can therefore be freely reset or changed by user code, for example, to additionally count a number of pages that are not build by \LaTeX but are added in a later part of the process, e.g., cover pages or picture pages made externally.

Important: as this is a page-related counter its value is only reliable inside the output routine!

<code>\PreviousTotalPages</code>	<code>\PreviousTotalPages</code>
----------------------------------	----------------------------------

Command that expands to the number of total pages from the previous run. If there was no previous run or if used in the preamble it expands to 0. Note that this is a command and not a counter, so in order to display the number in, say, Roman numerals you have to assign its value to a counter and then use `\Roman` on that counter.

1.7 Debugging shipout code

<code>\DebugShipoutsOn</code>	<code>\DebugShipoutsOn</code>
<code>\DebugShipoutsOff</code>	Turn the debugging of shipout code on or off. This displays changes made to the shipout data structures.
<code>\shipout_debug_on:</code>	
<code>\shipout_debug_off:</code>	

Todo: This needs some rationalizing and may not stay this way.

2 Emulating commands from other packages

The packages in this section are no longer necessary, but as they are used by other packages, they are emulated when they are explicitly loaded with `\usepackage` or `\RequirePackage`.

Please note that the emulation only happens if the package is explicitly requested, i.e., the commands documented below are not automatically available in the L^AT_EX kernel! If you write a new package we suggest to use the appropriate kernel hooks directly instead of loading the emulation.

2.1 Emulating atbegshi

<code>\AtBeginShipoutUpperLeft</code>	<code>\AddToHook {shipout/before} {...\AtBeginShipoutUpperLeft{<code>}...}</code>
<code>\AtBeginShipoutUpperLeftForeground</code>	

This adds a `picture` environment into the background of the shipout box expecting `<code>` to contain `picture` commands. The same effect can be obtained by simply using kernel features as follows:

```
\AddToHook{shipout/background}{<code>}
```

There is one technical difference: if `\AtBeginShipoutUpperLeft` is used several times each invocation is put into its own box inside the shipout box whereas all `<code>` going into `shipout/background` ends up all in the same box in the order it is added or sorted based on the rules for the hook chunks.

`\AtBeginShipoutUpperLeftForeground` is similar with the difference that the `picture` environment is placed in the foreground. To model it with the kernel functions use the hook `shipout/foreground` instead.

<code>\AtBeginShipoutAddToBox</code>	<code>\AddToHook {shipout/before} {...\AtBeginShipoutAddToBox{<code>}...}</code>
<code>\AtBeginShipoutAddToBoxForeground</code>	

These work like `\AtBeginShipoutUpperLeft` and `\AtBeginShipoutUpperLeftForeground` with the difference that `<code>` is directly placed into an `\hbox` inside the shipout box and not surrounded by a `picture` environment.

To emulate them using `shipout/background` or `shipout/foreground` you may have to wrap `<code>` into a `\put` statement but if the code is not doing any typesetting just adding it to the hook should be sufficient.

<code>\AtBeginShipoutBox</code>	This is the name of the shipout box as <code>atbegshi</code> knows it.
---------------------------------	--

<code>\AtBeginShipoutOriginalShipout</code>	
---	--

This is the name of the `\shipout` primitive as `atbegshi` knows it. This bypasses all the mechanisms set up by the L^AT_EX kernel and there are various scenarios in which it can therefore fail. It should only be used to run existing legacy `atbegshi` code but not in newly developed applications.

The kernel alternative is `\RawShipout` which is integrated with the L^AT_EX mechanisms and updates, for example, the `\ReadOnlyShipoutCounter` counter. Please use `\RawShipout` for new code if you want to bypass the before, foreground and background hooks.

`\AtBeginShipoutInit` By default `atbegshi` delayed its action until `\begin{document}`. This command was forcing it in an earlier place. With the new concept it does nothing.

`\AtBeginShipout` `\AtBeginShipout{<code>}` \equiv `\AddToHook{shipout/before}{<code>}`
`\AtBeginShipoutNext` `\AtBeginShipoutNext{<code>}` \equiv `\AddToHookNext{shipout/before}{<code>}`

This is equivalent to filling the `shipout/before` hook by either using `\AddToHook` or `\AddToHookNext`, respectively.

`\AtBeginShipoutFirst` The `atbegshi` names for `\AtBeginDvi` and `\DiscardShipoutBox`.
`\AtBeginShipoutDiscard`

2.2 Emulating `everyshi`

The `everyshi` package is providing commands to run arbitrary code just before the shipout starts. One point of difference: in the new shipout hooks the page is available as `\ShipoutBox` for inspection of change, one should not manipulate box 255 directly inside `shipout/before`, so old code doing this would change to use `\ShipoutBox` instead of 255 or `\@cclv`.

`\EveryShipout` `\EveryShipout{<code>}` \equiv `\AddToHook{shipout/before}{<code>}`

`\AtNextShipout` `\AtNextShipout{<code>}` \equiv `\AddToHookNext{shipout/before}{<code>}`

However, most use cases for `everyshi` are attempts to put some picture or text into the background or foreground of the page and that can be done today simply by using the `shipout/background` and `shipout/foreground` hooks without any need to coding.

2.3 Emulating `atenddvi`

The `atenddvi` package implemented only a single command: `\AtEndDvi` and that is now available out of the box so the emulation makes the package a no-op.

2.4 Emulating `everypage`

This package patched the original `\@begindvi` hook and replaced it with its own version. Its functionality is now covered by the hooks offered by the kernel so that there is no need for such patching any longer.

`\AddEverypageHook` `\AddEverypageHook{<code>}` \equiv
`\AddToHook{shipout/background}{\put(1in,-1in){<code>}}`

`\AddEverypageHook` is adding something into the background of every page at a position of 1in to the right and 1in down from the top left corner of the page. By using the kernel hook directly you can put your material directly to the right place, i.e., use other coordinates in the `\put` statement above.

`\AddThispageHook` `\AddThispageHook{<code>}` \equiv
`\AddToHookNext{shipout/background}{\put(1in,-1in){<code>}}`

The `\AddThispageHook` wrapper is similar but uses `\AddToHookNext`.

3 The Implementation

1 <@@=shipout>

At the moment the whole module rolls back in one go, but if we make any modifications in later releases this will then need splitting.

```
2 <*2ekernel | latexrelease>
3 <latexrelease>\IncludeInRelease{2020/10/01}%
4 <latexrelease>          {\shipout}{Hook management (shipout)}%
5 \ExplSyntaxOn
```

3.1 Debugging

`\g__shipout_debug_bool` Holds the current debugging state.

```
6 \bool_new:N \g__shipout_debug_bool
```

(End of definition for \g__shipout_debug_bool.)

`\shipout_debug_on:` Turns debugging on and off by redefining `__shipout_debug:n`.

```
\shipout_debug_off:
\__shipout_debug:n
\__shipout_debug_gset:
7 \cs_new_eq:NN \__shipout_debug:n \use_none:n
8 \cs_new_protected:Npn \shipout_debug_on:
9   {
10    \bool_gset_true:N \g__shipout_debug_bool
11    \__shipout_debug_gset:
12   }
13 \cs_new_protected:Npn \shipout_debug_off:
14   {
15    \bool_gset_false:N \g__shipout_debug_bool
16    \__shipout_debug_gset:
17   }
18 \cs_new_protected:Npn \__shipout_debug_gset:
19   {
20    \cs_gset_protected:Npx \__shipout_debug:n ##1
21    { \bool_if:NT \g__shipout_debug_bool {##1} }
22   }
```

(End of definition for \shipout_debug_on: and others. These functions are documented on page 1129.)

`\ShipoutBox` The box filled with the page to be shipped out (both L3 and L^AT_EX 2_ε name).

```
\l_shipout_box
23 \box_new:N \l_shipout_box
24 \cs_set_eq:NN \ShipoutBox \l_shipout_box
```

(End of definition for \ShipoutBox and \l_shipout_box. These functions are documented on page 1125.)

`\l__shipout_raw_box` The `\RawShipout` gets its own box but it is internal as there is no hook manipulation for it.

```
25 \box_new:N \l__shipout_raw_box
```

(End of definition for \l__shipout_raw_box.)

`__shipout_finalize_box:` For LuaTeX invoke the `pre_shipout_filter` callback.

```
26 \sys_if_engine luatex:TF
27 {
28   \newprotectedluacmd \__shipout_finalize_box:
29   \exp_args:Nx \everyjob {
30     \exp_not:V \everyjob
31     \exp_not:N \lua_now:n {
32       luatexbase.create_callback('pre_shipout_filter', 'list')
33       local~call, getbox, setbox = luatexbase.call_callback, tex.getbox, tex.setbox~
34       lua.get_functions_table()[\the \allocationnumber] = function()
35         local~head = getbox(\the \l_shipout_box)
36         local~result = call('pre_shipout_filter', head)
37         if~not (result == head) then~
38           setbox(\the \l_shipout_box, result~or~nil)
39         end~
40       end
41     }
42   }
43 } {
44   \cs_set_eq:NN \__shipout_finalize_box: \scan_stop:
45 }
```

(End of definition for __shipout_finalize_box:.)

`__shipout_execute:` This is going to be the code run by `\shipout`. The code follows closely the ideas from atbegshi, so not documenting that here for now.

```
46 \cs_set_protected:Npn \__shipout_execute: {
47   \tl_set:Nx \l__shipout_group_level_tl
48   { \int_value:w \tex_currentgrouplevel:D }
49   \tex_afterassignment:D \__shipout_execute_test_level:
50   \tex_setbox:D \l_shipout_box
51 }
```

(End of definition for __shipout_execute:.)

`\shipout` Overloading the `\shipout` primitive:

```
52 \cs_gset_eq:NN \shipout \__shipout_execute:
```

(End of definition for \shipout. This function is documented on page 1124.)

`\l__shipout_group_level_tl` Helper token list to record the group level at which `__shipout_execute:` is encountered.

```
53 \tl_new:N \l__shipout_group_level_tl
```

(End of definition for \l__shipout_group_level_tl.)

`__shipout_execute_test_level:` If the group level has changed then we are still constructing `\l_shipout_box` and to continue we need to wait until the current group has finished, hence the `\tex_aftergroup:D`.

```
54 \cs_new:Npn \__shipout_execute_test_level: {
55   \int_compare:nNnT
56     \l__shipout_group_level_tl < \tex_currentgrouplevel:D
57     \tex_aftergroup:D \__shipout_execute_cont:
58 }
```

(End of definition for __shipout_execute_test_level:.)

`_shipout_execute_cont:` This does the actual shipout running several hooks as part of it. The code for them is passed as argument #2 to #4 to `_shipout_execute_main_cont:Nnnn`; the first argument is the box to be shipped out.

```

59 \cs_new:Npn \_shipout_execute_cont: {
60   \_shipout_execute_main_cont:Nnnn
61   \l_shipout_box
62   { \hook_use:n {shipout/before} }
63   { \hook_if_empty:nF {shipout/foreground}
64     { \_shipout_add_foreground_picture:n
65       { \hook_use:n {shipout/foreground} } } }

```

If the user hook for the background (`shipout/background`) has no code, there might still code in the kernel hook so we need to test for this too. We only test for the `\@kernel@before@shipout@background` though. If the `\@kernel@after@shipout@background` needs executing even if the user hook is empty then we can add another test (or the kernel could put something into the before hook).

```

66     \bool_lazy_and:nnF
67     { \hook_if_empty_p:n {shipout/background} }
68     { \tl_if_empty_p:N \@kernel@before@shipout@background }
69     { \_shipout_add_background_picture:n
70       { \@kernel@before@shipout@background
71         \hook_use:n {shipout/background}
72         \@kernel@after@shipout@background }
73     }
74   }
75   { \hook_use:n {shipout/after} }
76 }

```

(End of definition for `_shipout_execute_cont:.`)

`_shipout_execute_main_cont:Nnnn` When we have reached this point the shipout box has been processed and is available in `\l_shipout_box` and ready for real ship out (unless it gets discarded during the process).

The three arguments hold hook code that is executed just before the actual shipout (#1), within the shipout adding background and foreground material (#2) and after the shipout has happened (#3). These are passed as arguments because the same code without those hooks is also used when doing a “raw” shipout implemented by `\RawShipout`. The only hook that is always executed is that for the very last page, i.e., `shipout/lastpage`.

First we quickly check if it is void (can’t happen in the standard L^AT_EX output routine but `\shipout` might be called from a package that has some special processing logic). If it is void we aren’t shipping anything out and processing ends.⁵⁹

```

77 \cs_new:Npn \_shipout_execute_main_cont:Nnnn #1#2#3#4 {
78   \box_if_empty:NTF #1
79   { \@latex@warning@no@line{ Ignoring~ void~ shipout~ box } }
80   {

```

Otherwise we assume that we will ship something and prepare for final adjustments (in particular setting the state of `\protect` while we are running the hook code). We also save the current `\protect` state to restore it later.

```

81 %       \bool_gset_false:N \g__shipout_discard_bool % setting this would disable
82                                     % \DiscardShipoutBox on doc-level

```

⁵⁹In that case we don’t reset the deadcycles, that would be up to the OR processing logic to do.


```

83     \cs_set_eq:NN \__shipout_saved_protect: \protect
84     \set@typeset@protect

```

We also store the current shipout box dimension in registers, so that they can be used in the hook code.⁶⁰

```

85     \__shipout_get_box_size:N #1

```

Then we execute the `shipout/before` hook (or nothing in case of `\RawShipout`).

```

86     #2

```

In `\g_shipout_totalpages_int` we count all shipout attempts so we increment that counter already here (the other one is incremented later when we know for sure that we do a `\shipout`).

We increment it after running the above hook so that the values for `\g_shipout_totalpages_int` and `\g_shipout_readonly_int` are in sync while the hook is executed (in the case that `totalpages` isn't manually altered or through discarding pages that is).

```

87     \int_gincr:N \g_shipout_totalpages_int

```

The above hook might contain code that requests the page to be discarded so we now test for it.

```

88     \bool_if:NTF \g__shipout_discard_bool
89     { \@latex@info@no@line{Completed~ page~ discarded}
90     \bool_gset_false:N \g__shipout_discard_bool

```

As we are discarding the page box and not shipping anything out, we need to do some house cleaning and reset TeX's deadcycles so that it doesn't complain about too many calls to the OR without any shipout.

```

91     \tex_deadcycles:D \c_zero_int

```

Todo: In atbegshi the box was dropped but is that actually needed? Or the resetting of \protect to its kernel value?

```

92 %     \group_begin:
93 %     \box_set_eq_drop:NN #1 #1
94 %     \group_end:
95 %     \cs_set_eq:NN \protect \exp_not:N
96     }

```

Even if there was no explicit request to discard the box it is possible that the code for the hook `shipout/before` has voided the box (by mistake or deliberately). We therefore test once more but this time make it a warning, because the best practice way is to use the request mechanism.

```

97     { \box_if_empty:NTF #1
98     { \@latex@warning@no@line { Ignoring~ void~ shipout~ box.
99     \MessageBreak The~ shipout~ box~ was~ voided~ by~ hook~ code }
100     }

```

Finally, if the box is still non-empty we are nearly ready to ship it out. First we increment the total page counter so that we can later test if we have reached the final page according to our available information.⁶¹

```

101     {

```

⁶⁰This is not really necessary as the code could access them via `\box_ht:N`, etc., but it is perhaps convenient.

⁶¹Doing that earlier would be wrong because we might end up with the last page counted but discard and then we have no place to add the final objects into the output file.

```

102         \int_gincr:N \g_shipout_readonly_int
103         \__shipout_debug:n {
104             \typeout{Absolute~ page~ =~ \int_use:N \g_shipout_readonly_int
105                 \space (target:~ \@abspage@last)}
106         }

```

Then we store the box sizes again (as they may have changed) and then look at the hooks `shipout/foreground` and `shipout/background`. If either or both are non-empty we add a `picture` environment to the box (in the foreground and/or in the background) and execute the hook code inside that environment.

```

107         \__shipout_get_box_size:N #1

```

The first hook we run is the `shipout/firstpage` hook. This is only done once, then the `__shipout_run_firstpage_hook:` command redefines itself to do nothing. If the hook contains `\specials` for integration at the top of the page they will be temporarily stored in a safe place and added later with `__shipout_add_firstpage_specials:`.

```

108         \__shipout_run_firstpage_hook:

```

Run the hooks for background and foreground or, if this is called by `\RawShipout`, copy the box `\l__shipout_raw_box` to `\l_shipout_box` so that firstpage and lastpage material gets added if necessary (that is always done to `\l_shipout_box`).

```

109         #3

```

We then run `__shipout_add_firstpage_specials:` that adds the content of the hook `shipout/firstpage` to the start of the first page (if non-empty). It is then redefined to do nothing on later pages.

```

110         \__shipout_add_firstpage_specials:

```

Then we check if we have to add the `shipout/lastpage` hook or the corresponding kernel hook because we have reached the last page. This test will be false for all but one (and hopefully the correct) page.

```

111         \int_compare:nNnT \@abspage@last = \g_shipout_readonly_int
112         { \bool_lazy_and:nnF
113             { \hook_if_empty_p:n {shipout/lastpage} }
114             { \tl_if_empty_p:N \@kernel@after@shipout@lastpage }
115             { \__shipout_debug:n { \typeout{Executing~ lastpage~ hook~
116                 on~ page~ \int_use:N \g_shipout_readonly_int } }
117             \__shipout_add_foreground_box:n
118                 { \UseHook{shipout/lastpage}
119                 \@kernel@after@shipout@lastpage }

```

We record that we have handled the `shipout/lastpage` hook but only if we really did.

```

120         \bool_gset_true:N \g__shipout_lastpage_handled_bool
121     }
122 }

```

```

123     \hook_use:n {shipout}
124     \__shipout_finalize_box:

```

Finally we run the actual `TeX` primitive for `shipout`. As that will expand delayed `\write` statements inside the page in which protected commands should not expand we first change `\protect` to the appropriate definition for that case.

```

125         \cs_set_eq:NN \protect \exp_not:N
126         \tex_shipout:D \box_use:N \l_shipout_box

```

The `\l_shipout_box` may contain the firstpage material if this was the very first shipout. That makes it unsuitable for reuse in another shipout, so as a safety measure the next command resets `\l_shipout_box` to its earlier state if that is necessary. On later pages this is then a no-op.

```
127         \__shipout_drop_firstpage_specials:
```

The `shipout/after` hook (if in #4) needs to run with `\protected` commands again being executed, because that hook will “typeset” material added at the top of the next page.

```
128         \set@typeset@protect
129         #4
130     }
131 }
```

Restore the value of `\protect` in case `\shipout` is called outside of the output routine (where it is automatically restored because of the implicit group).

```
132     \cs_set_eq:NN \protect \__shipout_saved_protect:
133 }
134 }
```

(End of definition for `__shipout_execute_main_cont:Nnnn`.)

`__shipout_execute_raw:` This implements the “raw” shipout which bypasses the before, foreground, background and after hooks. It follows the same pattern than `__shipout_execute_raw:` except that it finally calls `__shipout_execute_main_cont:Nnnn` with three empty arguments. instead of the hook code.

```
135 \cs_set_protected:Npn \__shipout_execute_raw: {
136     \tl_set:Nx \l__shipout_group_level_tl
137     { \int_value:w \tex_currentgrouplevel:D }
138     \tex_afterassignment:D \__shipout_execute_test_level_raw:
139     \tex_setbox:D \l__shipout_raw_box
140 }

141 \cs_new:Npn \__shipout_execute_test_level_raw: {
142     \int_compare:nNnT
143     \l__shipout_group_level_tl < \tex_currentgrouplevel:D
144     \tex_aftergroup:D \__shipout_execute_nohooks_cont:
145 }
```

Well, not totally empty arguments, we add some debugging if we are actually doing a shipout.

```
146 \cs_new:Npn \__shipout_execute_nohooks_cont: {
147     \__shipout_execute_main_cont:Nnnn \l__shipout_raw_box
148     {} { \__shipout_debug:n{ \typeout{Doing~ raw~ shipout~ ...} } }
149     \box_set_eq:NN \l_shipout_box \l__shipout_raw_box } {}
150 }
```

(End of definition for `__shipout_execute_raw:` and `__shipout_execute_test_level_raw:.`)

`\RawShipout` The interface name for raw shipout.

```
151 \cs_get_eq:NN \RawShipout \__shipout_execute_raw:
```

(End of definition for `\RawShipout`. This function is documented on page 1124.)

`__shipout_saved_protect:` Remember the current `\protect` state.

```
152 \cs_new_eq:NN \__shipout_saved_protect: \protect
```

(End of definition for `_shipout_saved_protect:`.)

```
shipout/before Declaring all hooks for the shipout code.
shipout        153 \hook_new:n{shipout/before}
shipout/after  154 \hook_new:n{shipout}
shipout/foreground 155 \hook_new:n{shipout/after}
shipout/background 156 \hook_new:n{shipout/foreground}
shipout/firstpage 157 \hook_new:n{shipout/background}
shipout/lastpage 158 \hook_new:n{shipout/firstpage}
               159 \hook_new:n{shipout/lastpage}
```

(End of definition for `shipout/before` and others. These functions are documented on page 1125.)

```
\@kernel@after@shipout@lastpage And here are the internal kernel hooks going before or after the public ones where needed.
\@kernel@before@shipout@background
\@kernel@after@shipout@background 160 \let\@kernel@after@shipout@lastpage\@empty
                                   161 \let\@kernel@before@shipout@background\@empty
                                   162 \let\@kernel@after@shipout@background\@empty
```

(End of definition for `\@kernel@after@shipout@lastpage`, `\@kernel@before@shipout@background`, and `\@kernel@after@shipout@background`.)

```
\_shipout_run_firstpage_hook: There are three commands to handle the shipout/firstpage hook: \_shipout_run_
firstpage_hook:, \_shipout_add_firstpage_specials: and \_shipout_drop_
firstpage_specials:.
```

That hook is supposed to contain `\specials` and similar material to be placed at the very beginning of the output page and so it needs careful placing to avoid that anything else gets in front of it. And this means we have to wait with this until other hooks such as `shipout/background` have added their bits. It is also important that such `\specials` show up only on the very first page, so if this page gets saved before `\shipout` for later reuse, we have to make sure that they aren't in the saved version.

In addition the hook may also contain code to be executed “first”, e.g., visible from code in `shipout/background` and this conflicts with adding the `\specials` late.

Therefore the processing is split into different parts: `_shipout_run_firstpage_hook:` is done early and checks if there is any material in the hook.

```
163 \cs_new:Npn \_shipout_run_firstpage_hook: {
164   \hook_if_empty:nTF {shipout/firstpage}
```

If not then we define the other two commands to do nothing.

```
165   {
166     \cs_gset_eq:NN \_shipout_add_firstpage_specials: \prg_do_nothing:
167     \cs_gset_eq:NN \_shipout_drop_firstpage_specials: \prg_do_nothing:
168   }
```

If there is material we execute inside a box, which means any `\special` will end up in that box and any other code is executed and can have side effects (as long as they are global).

```
169   {
170     \hbox_set:Nn \l__shipout_firstpage_box { \UseHook{shipout/firstpage} }
171   }
```

Once we are here we change the definition to do nothing next time and we also change the command used to implement `\AtBeginDvi` to become a warning and not add further material to a hook that is never used again.

```

172 \cs_gset_eq:NN \__shipout_run_firstpage_hook: \prg_do_nothing:
173 \cs_gset:Npn \__shipout_add_firstpage_material:Nn ##1 ##2 {
174   \@latex@warning{ First~ page~ is~ already~ shipped~ out,~ ignoring
175                   \MessageBreak \string##1 }
176 }
177 }

```

(End of definition for `__shipout_run_firstpage_hook:`.)

`__shipout_add_firstpage_specials:` The `__shipout_add_firstpage_specials:` then adds the `\specials` stored in `\l__shipout_firstpage_box` to the page to be shipped out when the time is ready. Note that if there was no material in the `shipout/firstpage` hook then this command gets redefined to do nothing. But for most documents there is something, e.g., some PostScript header, or some meta data declaration, etc. so by default we assume there is something to do.

```

178 \cs_new:Npn \__shipout_add_firstpage_specials: {

```

First we make a copy of the `\l_shipout_box` that we can restore it later on.

```

179 \box_set_eq:NN \l__shipout_raw_box \l_shipout_box

```

Adding something to the beginning means adding it to the background as that layer is done first in the output.

```

180 \__shipout_add_background_box:n { \hbox_unpack_drop:N \l__shipout_firstpage_box }

```

After the actual shipout `__shipout_drop_firstpage_specials:` is run to restore the earlier content of `\l_shipout_box` and then redefines itself again to do nothing.

As a final act we change the definition to do nothing next time.

```

181 \cs_gset_eq:NN \__shipout_add_firstpage_specials: \prg_do_nothing:
182 }

```

The `__shipout_drop_firstpage_specials:` is run after the shipout has occurred but before the `shipout/afterpage` hook is executed. That is the point where we have to restore the `\ShipoutBox` to its state without the `shipout/firstpage` material.

```

183 \cs_new:Npn \__shipout_drop_firstpage_specials: {
184   \box_set_eq:NN \l_shipout_box \l__shipout_raw_box

```

If there was no such material then `__shipout_run_firstpage_hook:` will have changed the definition to a no-op already. Otherwise this is what we do here.

```

185 \cs_gset_eq:NN \__shipout_drop_firstpage_specials: \prg_do_nothing:
186 }

```

(End of definition for `__shipout_add_firstpage_specials:` and `__shipout_drop_firstpage_specials:`.)

`\l__shipout_firstpage_box` The box to hold any firstpage `\specials`.

```

187 \box_new:N \l__shipout_firstpage_box

```

(End of definition for `\l__shipout_firstpage_box:`.)

`\g__shipout_lastpage_handled_bool` A boolean to signal if we have already handled the `shipout/lastpage` hook.

```

188 \bool_new:N \g__shipout_lastpage_handled_bool

```

(End of definition for `\g__shipout_lastpage_handled_bool:`.)

`_shipout_add_firstpage_material:N` This command adds material to the `shipout/firstpage` hook. It is used in `\AtBeginDvi`, etc. The first argument is the command through which it is called. Initially this is ignored but once we are passed the first page it can be used to generate a warning message mentioning the right user command.

```
189 \cs_new:Npn \_shipout_add_firstpage_material:Nn #1#2 {
190   \AddToHook{shipout/firstpage}{#2}
191 }
```

(End of definition for `_shipout_add_firstpage_material:Nn`.)

`_shipout_get_box_size:N` Store the box dimensions in dimen registers.

Todo: This could/should perhaps be generalized to set height depth and width given an arbitrary box.

```
192 \cs_new:Npn \_shipout_get_box_size:N #1 {
193   \dim_set:Nn \l_shipout_box_ht_dim { \box_ht:N #1 }
194   \dim_set:Nn \l_shipout_box_dp_dim { \box_dp:N #1 }
195   \dim_set:Nn \l_shipout_box_wd_dim { \box_wd:N #1 }
196   \dim_set:Nn \l_shipout_box_ht_plus_dp_dim
197     { \l_shipout_box_ht_dim + \l_shipout_box_dp_dim }
198 }
```

(End of definition for `_shipout_get_box_size:N`.)

`\l_shipout_box_ht_dim` And here are the variables set by `_shipout_get_box_size:N`.

```
\l_shipout_box_dp_dim 199 \dim_new:N \l_shipout_box_ht_dim
\l_shipout_box_wd_dim 200 \dim_new:N \l_shipout_box_dp_dim
\l_shipout_box_ht_plus_dp_dim 201 \dim_new:N \l_shipout_box_wd_dim
202 \dim_new:N \l_shipout_box_ht_plus_dp_dim
```

(End of definition for `\l_shipout_box_ht_dim` and others. These functions are documented on page 1125.)

`\g_shipout_discard_bool` Indicate whether or not the current page box should be discarded

```
203 \bool_new:N \g_shipout_discard_bool
```

(End of definition for `\g_shipout_discard_bool`.)

`\l_shipout_tmp_box` We need a box for the background and foreground material and a token register to

`\l_shipout_saved_badness_tl` remember badness settings as we disable them during the buildup below.

```
204 \box_new:N \l_shipout_tmp_box
205 \tl_new:N \l_shipout_saved_badness_tl
```

(End of definition for `\l_shipout_tmp_box` and `\l_shipout_saved_badness_tl`.)

`_shipout_add_background_box:n` In standard L^AT_EX the shipout box is always a `\vbox` but here we are allow for other usage as well, in case some package has its own output routine.

```
206 \cs_new:Npn \_shipout_add_background_box:n #1
207 { \_shipout_get_box_size:N \l_shipout_box
```

But we start testing for a vertical box as that should be the normal case.

```
208   \box_if_vertical:NTF \l_shipout_box
209     {
```

Save current values of `\vfuzz` and `\vbadness` then change them to allow box manipulations without warnings.

```

210     \tl_set:Nx \l__shipout_saved_badness_tl
211     { \vfuzz=\the\vfuzz\relax
212       \vbadness=\the\vbadness\relax }
213     \vfuzz=\c_max_dim
214     \vbadness=\c_max_int

```

Then we reconstruct `\l_shipout_box` ...

```

215     \vbox_set_to_ht:Nnn \l_shipout_box \l_shipout_box_ht_plus_dp_dim
216     {

```

... the material in #1 is placed into a horizontal box with zero dimensions.

```

217         \hbox_set:Nn \l__shipout_tmp_box
218         { \l__shipout_saved_badness_tl #1 }
219         \box_set_wd:Nn \l__shipout_tmp_box \c_zero_dim
220         \box_set_ht:Nn \l__shipout_tmp_box \c_zero_dim
221         \box_set_dp:Nn \l__shipout_tmp_box \c_zero_dim

```

The we typeset that box followed by whatever was in `\l_shipout_box` before (unpacked).

```

222         \skip_zero:N \baselineskip
223         \skip_zero:N \lineskip
224         \skip_zero:N \lineskiplimit
225         \box_use:N \l__shipout_tmp_box
226         \vbox_unpack:N \l_shipout_box

```

The `\kern` ensures that the box has no depth which is afterwards explicitly corrected.

```

227         \kern \c_zero_dim
228     }
229     \box_set_ht:Nn \l_shipout_box \l_shipout_box_ht_dim
230     \box_set_dp:Nn \l_shipout_box \l_shipout_box_dp_dim

```

Todo: The whole boxing maneuver looks a bit like overkill to me, but for the moment I leave.

```

231     \l__shipout_saved_badness_tl
232 }
233 {

```

A horizontal box is handled in a similar way. The last case would be a void box in which case we do nothing hence the missing F branch.

```

234     \box_if_horizontal:NT \l_shipout_box
235     {
236         \tl_set:Nx \l__shipout_saved_badness_tl
237         { \hfuzz=\the\hfuzz\relax
238           \hbadness=\the\hbadness\relax }
239         \hfuzz=\c_max_dim
240         \hbadness=\c_max_int
241         \hbox_set_to_wd:Nnn \l_shipout_box \l_shipout_box_wd_dim
242         {
243             \hbox_set:Nn \l__shipout_tmp_box
244             { \l__shipout_saved_badness_tl #1 }
245             \box_set_wd:Nn \l__shipout_tmp_box \c_zero_dim
246             \box_set_ht:Nn \l__shipout_tmp_box \c_zero_dim
247             \box_set_dp:Nn \l__shipout_tmp_box \c_zero_dim

```

```

248         \box_move_up:nn
249         \l_shipout_box_ht_dim
250         { \box_use:N \l__shipout_tmp_box }
251         \hbox_unpack:N \l_shipout_box
252     }
253     \l__shipout_saved_badness_tl
254 }
255 }
256 }

```

(End of definition for __shipout_add_background_box:n.)

_shipout_add_foreground_box:n Foreground boxes are done in the same way, only the order and placement of boxes has to be done differently.

```

257 \cs_new:Npn \__shipout_add_foreground_box:n #1
258 {
259   \box_if_vertical:NTF \l_shipout_box
260   {
261     \tl_set:Nx \l__shipout_saved_badness_tl
262     { \vfuzz=\the\vfuzz\relax
263       \vbadness=\the\vbadness\relax }
264     \vfuzz=\c_max_dim
265     \vbadness=\c_max_int
266     \vbox_set_to_ht:Nnn \l_shipout_box \l_shipout_box_ht_plus_dp_dim
267     {
268       \hbox_set:Nn \l__shipout_tmp_box
269       { \l__shipout_saved_badness_tl #1 }
270       \box_set_wd:Nn \l__shipout_tmp_box \c_zero_dim
271       \box_set_ht:Nn \l__shipout_tmp_box \c_zero_dim
272       \box_set_dp:Nn \l__shipout_tmp_box \c_zero_dim
273       \skip_zero:N \baselineskip
274       \skip_zero:N \lineskip
275       \skip_zero:N \lineskiplimit
276       \vbox_unpack:N \l_shipout_box
277       \kern -\l_shipout_box_ht_plus_dp_dim
278       \box_use:N \l__shipout_tmp_box
279       \kern \l_shipout_box_ht_plus_dp_dim
280     }
281     \l__shipout_saved_badness_tl
282     \box_set_ht:Nn \l_shipout_box \l_shipout_box_ht_dim
283     \box_set_dp:Nn \l_shipout_box \l_shipout_box_dp_dim
284   }
285   {
286     \box_if_horizontal:NT \l_shipout_box
287     {
288       \tl_set:Nx \l__shipout_saved_badness_tl
289       { \hfuzz=\the\hfuzz\relax
290         \hbadness=\the\hbadness\relax }
291       \hfuzz=\c_max_dim
292       \hbadness=\c_max_int
293       \hbox_set_to_wd:Nnn \l_shipout_box \l_shipout_box_wd_dim
294       {
295         \hbox_unpack:N \l_shipout_box
296         \kern -\box_wd:N \l_shipout_box

```



```

297         \hbox_set:Nn \l__shipout_tmp_box
298             { \l__shipout_saved_badness_tl #1 }
299         \box_set_wd:Nn \l__shipout_tmp_box \c_zero_dim
300         \box_set_ht:Nn \l__shipout_tmp_box \c_zero_dim
301         \box_set_dp:Nn \l__shipout_tmp_box \c_zero_dim
302         \box_move_up:nn { \box_ht:N \l__shipout_box }
303             { \box_use:N \l__shipout_tmp_box }
304         \kern \box_wd:N \l__shipout_box
305     }%
306     \l__shipout_saved_badness_tl
307 }
308 }
309 }

```

(End of definition for `__shipout_add_foreground_box:n`.)

`__shipout_init_page_origins:`
`\c__shipout_horigin_tl`
`\c__shipout_vorigin_tl`

Two constants holding the offset of the top-left with respect to the media box.

Setting the constants this way is courtesy of Bruno.

We delay setting the constants to the last possible place as there might be updates in the preamble or even in the `begindocument` hook that affects their setup.

```

310 \cs_new:Npn \__shipout_init_page_origins: {
311   \tl_const:Nx \c__shipout_horigin_tl
312     {
313       \cs_if_exist_use:NTF \pdfvariable { horigin }
314       { \cs_if_exist_use:NF \pdfhorigin { 1in } }
315     }
316   \tl_const:Nx \c__shipout_vorigin_tl
317     {
318       \cs_if_exist_use:NTF \pdfvariable { vorigin }
319       { \cs_if_exist_use:NF \pdfvorigin { 1in } }
320     }

```

After the constants have been set there is no need to execute this command again, in fact it would raise an error, so we redefine it to do nothing.

```

321   \cs_gset_eq:NN \__shipout_init_page_origins: \prg_do_nothing:
322 }

```

(End of definition for `__shipout_init_page_origins:`, `\c__shipout_horigin_tl`, and `\c__shipout_vorigin_tl`.)

`__shipout_picture_overlay:n`

Put the argument into a `picture` environment that doesn't take up any size and uses `1pt` for `\unitlength`.

Todo: Could perhaps be generalized as it might be useful elsewhere. For now it is not.

```

323 \cs_new:Npn \__shipout_picture_overlay:n #1 {

```

The very first time this is executed we have to initialize (and freeze) the origins.

```

324   \__shipout_init_page_origins:
325   \kern -\c__shipout_horigin_tl \scan_stop:
326   \vbox_to_zero:n {
327     \kern -\c__shipout_vorigin_tl \scan_stop:
328     \unitlength 1pt \scan_stop:

```

This mimics a simple zero-sized picture environment. The `\hss` is need in case there is horizontal material (without using `\put` with a positive width).

```

329     \hbox_set_to_wd:Nnn \l__shipout_tmp_box \c_zero_dim
330         { \ignorespaces #1 \hss }
331     \box_set_ht:Nn \l__shipout_tmp_box \c_zero_dim
332     \box_set_dp:Nn \l__shipout_tmp_box \c_zero_dim
333     \box_use:N \l__shipout_tmp_box
334     \tex_vss:D
335 }
336 }

```

(End of definition for `__shipout_picture_overlay:n`.)

`__shipout_add_background_picture:n` Put a `picture` env in the background of the shipout box with its reference point in the top-left corner.

```

337 \cs_new:Npn \__shipout_add_background_picture:n #1 {
338     \__shipout_add_background_box:n { \__shipout_picture_overlay:n {#1} }
339 }

```

(End of definition for `__shipout_add_background_picture:n`.)

`__shipout_add_foreground_picture:n` Put a `picture` env in the foreground of the shipout box with its reference point in the top-left corner.

```

340 \cs_new:Npn \__shipout_add_foreground_picture:n #1 {
341     \__shipout_add_foreground_box:n { \__shipout_picture_overlay:n {#1} }
342 }

```

(End of definition for `__shipout_add_foreground_picture:n`.)

`\shipout_discard:` Request that the next shipout box should be discarded. At the moment this is just setting a boolean, but we may want to augment this behavior that the position of the call is taken into account (in case \LaTeX looks ahead and is not using the position for on the next page).

```

343 \cs_new_protected:Npn \shipout_discard: {
344     \bool_gset_true:N \g__shipout_discard_bool
345 }

```

(End of definition for `\shipout_discard:`. This function is documented on page 1128.)

3.2 Handling the end of job hook

At the moment this is partly solved by using the existing hooks. But rather than putting the code into these hooks it should be moved to the right place directly as we shouldn't prefill hooks with material unless it needs to interact with other code.

`\g_shipout_readonly_int` We count every shipout activity that makes a page (but not those that are discarded) in order to know how many pages got produced.

```

346 \int_new:N \g_shipout_readonly_int

```

For $\LaTeX 2_\epsilon$ it is available as a command (i.e., a \TeX counter only).

```

347 \cs_new_eq:NN \ReadonlyShipoutCounter \g_shipout_readonly_int

```

(End of definition for `\g_shipout_readonly_int` and `\ReadonlyShipoutCounter`. These functions are documented on page 1129.)

`\g_shipout_totalpages_int` We count every shipout attempt (even those that are discarded) in this counter. It is not used in the code but may get used in user code.

```
348 \int_new:N \g_shipout_totalpages_int
```

For L^AT_εX this is offered as a L^AT_εX counter so can be easily typeset inside the output routine to display things like “\thepage/\thetotalpages”, etc.

```
349 \cs_new_eq:NN \c@totalpages \g_shipout_totalpages_int
```

```
350 \cs_new:Npn \thetotalpages { \arabic{totalpages} }
```

(End of definition for `\g_shipout_totalpages_int` and `\c@totalpages`. These functions are documented on page 1129.)

`\@abspage@last` In `\@abspage@last` record the number of pages from the last run. This is written to the `.aux` and this way made available to the next run. In case there is no `.aux` file or the statement is missing from it we initialize it with the largest possible number in T_εX. We use this as the default because then we are inserting the `shipout/lastpage` on the last page (or after the last page) but not on page 1 for a multipage document.

```
351 \xdef\@abspage@last{\number\maxdimen}
```

(End of definition for `\@abspage@last`.)

`\enddocument` Instead of using the hooks `enddocument` and `enddocument/afterlastpage` we add this code to private kernel hooks to be 100% sure when it is executed and to avoid cluttering the hooks with data that is always there.

Inside `\enddocument` there is a `\clearpage`. Just before that we execute this code here. There is a good chance that we are on the last page. Therefore, if we don't know the value from the last run, we assume that the current page is the right one. So we set `\@abspage@last` and as a result the next shipout will run the `shipout/lastpage` code. Of course, if there are floats that still need a placement this guess will be wrong but then rerunning the document will give us the correct value next time around.

```
\@kernel@after@enddocument 352 \g@addto@macro \@kernel@after@enddocument {
353   \int_compare:nNnT \@abspage@last = \maxdimen
354   {
```

We use L^AT_εX coding as `\@abspage@last` is not an L₃ name.

```
355   \xdef\@abspage@last{ \int_eval:n {\g_shipout_readonly_int + 1} }
356   }
357 }
```

`\@kernel@after@enddocument@afterlastpage` Once the `\clearpage` has done its work inside `\enddocument` we know for sure how many pages this document has, so we record that in the `.aux` file for the next run.

```
358 \g@addto@macro \@kernel@after@enddocument@afterlastpage {
```

There is one special case: If no output is produced then there is no point in a) recording the number as 0 will never match the page number of a real page and b) adding an extra page to run the `shipout/lastpage` is pointless as well (as it would remain forever). So we test for this and run the code only if there have been pages.

```
359   \int_compare:nNnF \g_shipout_readonly_int = 0
360   {
```

This ends up in the `.aux` so we use L^AT_εX names here.

Todo: This needs an interface for \nofiles in expl3, doesn't at the moment!

```
361     \if@filesw
362         \iow_now:Nx \@auxout {
363             \gdef\string\@abspage@last {\int_use:N \g_shipout_readonly_int}}
364     \fi
```

But we may have guessed wrongly earlier and have run it too early or we still have to run the `shipout/lastpage` even though there is no page to place it into. If that is the case we make a trivial extra page and put it there. This temporary page will then vanish again on the next run but helps to keep pdf viewers happy. In either case we should put out an appropriate “rerun” warning.

```
365     \bool_if:NTF \g__shipout_lastpage_handled_bool
366     {
```

If the hook was already executed, we have to test if that total shipouts match the shipouts from last run (because that corresponds to the page it was executed). If not we output a warning.

```
367         \int_compare:nNnF \@abspage@last = \g_shipout_readonly_int
368         {
369             \@latex@warning@no@line{Hook~ 'shipout/lastpage'~ executed-
370 on~ wrong~ page~ (\@abspage@last\space not~
371 \int_use:N\g_shipout_readonly_int).\MessageBreak
372 Rerun~ to~ correct~ this}%
373         }
374     }
375     {
```

If the hook was not run, we need to add an extra page and place it there. However, making this extra page in case the hook is actually empty would be forcing a rerun without any reason, so we check that condition and also check if `\@kernel@after@shipout@lastpage` contains any code. If both are empty we omit the page generation.

```
376     \bool_lazy_and:nnF
377     { \hook_if_empty_p:n {shipout/lastpage} }
378     { \tl_if_empty_p:N \@kernel@after@shipout@lastpage }
379     {
380         \tex_shipout:D\vbox to\textheight
381         {
382             \hbox:n { \UseHook{shipout/lastpage}
383                 \@kernel@after@shipout@lastpage }

```

This extra page could be totally empty except for the hook content, but to help the user understanding why it is there we put some text into it.

```
384         \__shipout_excuse_extra_page:
385         \null
386     }
```

At this point we also signal to L^AT_EX's endgame that a rerun is necessary so that an appropriate message can be shown on the terminal. We do this by simply defining a command used as a flag and tested in `\enddocument`.

```
387         \cs_gset_eq:NN \@extra@page@added \relax
388     }
389 }
390 }
391 }
```

(End of definition for `\enddocument`, `\@kernel@after@enddocument`, and `\@kernel@after@enddocument@afterlastpage`.)

```
\_shipout_excuse_extra_page: Say mea culpa ...
392 \cs_new:Npn \_shipout_excuse_extra_page: {
393   \vfil
394   \begin{center}
395     \bfseries Temporary~ page!
396   \end{center}
397   \LaTeX{}~ was~ unable~ to~ guess~ the~ total~ number~ of~ pages~
398   correctly.~ ~ As~ there~ was~ some~ unprocessed~ data~ that~
399   should~ have~ been~ added~ to~ the~ final~ page~ this~ extra~
400   page~ has~ been~ added~ to~ receive~ it.
401   \par
402   If~ you~ rerun~ the~ document~ (without~ altering~ it)~ this~
403   surplus~ page~ will~ go~ away,~ because~ \LaTeX{}~ now~ knows~
404   how~ many~ pages~ to~ expect~ for~ this~ document.
405   \vfil
406 }
```

(End of definition for `_shipout_excuse_extra_page:.`)

`\PreviousTotalPages` In the preamble before the aux file was read `\PreviousTotalPages` is always zero.

`\@kernel@before@begindocument`

```
407 \def\PreviousTotalPages{0}
```

In the aux file there should be an update for `\@abspage@last` recording the number of pages from the previous run. If not that macro holds the value of `\maxdimen`. So we test for it and update `\PreviousTotalPages` if there was a real value. This should happen just before the `begindocument` hook is executed so that the value can be used inside that hook.

```
408 \g@addto@macro\@kernel@before@begindocument
409   {\ifnum\@abspage@last<\maxdimen
410     \xdef\PreviousTotalPages{\@abspage@last}\fi}
```

(End of definition for `\PreviousTotalPages` and `\@kernel@before@begindocument`. These functions are documented on page 1129.)

4 Legacy L^AT_EX 2_ε interfaces

`\DiscardShipoutBox` Request that the next shipout box is to be discarded.

```
411 \cs_new_eq:NN \DiscardShipoutBox \shipout_discard:
```

(End of definition for `\DiscardShipoutBox`. This function is documented on page 1128.)

`\AtBeginDvi` If we roll forward from an earlier kernel `\AtBeginDvi` is defined so we better not use `\cs_new_protected:Npn` here.

```
412 \cs_set_protected:Npn \AtBeginDvi
413   {\_shipout_add_firstpage_material:Nn \AtBeginDvi}
```

(End of definition for `\AtBeginDvi`. This function is documented on page 1127.)

`\DebugShipoutsOn`

`\DebugShipoutsOff`

```
414 \cs_new_eq:NN \DebugShipoutsOn \shipout_debug_on:
415 \cs_new_eq:NN \DebugShipoutsOff \shipout_debug_off:
```

(End of definition for `\DebugShipoutsOn` and `\DebugShipoutsOff`. These functions are documented on page 1129.)

5 Internal commands needed elsewhere

These internal commands use double and triple @ signs so we need to stop getting them translated to the module name.

416 <@=@>

Some internals needed elsewhere.

```

\@expl@@@shipout@add@firstpage@material@@Nn
\@expl@@@shipout@add@background@box@@n
\@expl@@@shipout@add@foreground@box@@n
\@expl@@@shipout@add@background@picture@@n
\@expl@@@shipout@add@foreground@picture@@n
417 \cs_set_eq:NN \@expl@@@shipout@add@firstpage@material@@Nn
418     \__shipout_add_firstpage_material:Nn
419 \cs_set_eq:NN \@expl@@@shipout@add@background@box@@n
420     \__shipout_add_background_box:n
421 \cs_set_eq:NN \@expl@@@shipout@add@foreground@box@@n
422     \__shipout_add_foreground_box:n
423 \cs_set_eq:NN \@expl@@@shipout@add@background@picture@@n
424     \__shipout_add_background_picture:n
425 \cs_set_eq:NN \@expl@@@shipout@add@foreground@picture@@n
426     \__shipout_add_foreground_picture:n

```

(End of definition for \@expl@@@shipout@add@firstpage@material@@Nn and others.)

```

427 \ExplSyntaxOff
428 </2ekernel | latexrelease>
429 <latexrelease>\EndIncludeInRelease

```

Rolling back here doesn't undefine the interface commands as they may be used in packages without rollback functionality. So we just make them do nothing which may or may not work depending on the code usage.

```

430 <latexrelease>\IncludeInRelease{0000/00/00}%
431 <latexrelease>           {\shipout}{Hook management (shipout)}%
432 <latexrelease>

```

If we roll forward then `\tex_shipout:D` may not be defined in which case `\shipout` does have it original definition and so we must not `\let` it to something else which is `\relax`!

```

433 <latexrelease>\ifcsname tex_shipout:D\endcsname
434 <latexrelease>\expandafter\let\expandafter\shipout
435 <latexrelease>           \csname tex_shipout:D\endcsname
436 <latexrelease>\fi
437 <latexrelease>
438 <latexrelease>\let \RawShipout\@undefined
439 <latexrelease>\let \ShipoutBox\@undefined
440 <latexrelease>\let \ReadOnlyShipoutCounter \@undefined
441 <latexrelease>\let \c@totalpages \@undefined
442 <latexrelease>\let \thetotalpages \@undefined
443 <latexrelease>
444 <latexrelease>\let \DiscardShipoutBox \@undefined
445 <latexrelease>\let \DebugShipoutsOn \@undefined
446 <latexrelease>\let \DebugShipoutsOff \@undefined
447 <latexrelease>
448 <latexrelease>\DeclareRobustCommand \AtBeginDvi [1]{%
449 <latexrelease> \global \setbox \@begindvibox
450 <latexrelease> \vbox{\unvbox \@begindvibox #1}%
451 <latexrelease>}

```

```

452 <latexrelease>
453 <latexrelease>\let \AtBeginShipout \@undefined
454 <latexrelease>\let \AtBeginShipoutNext \@undefined
455 <latexrelease>
456 <latexrelease>\let \AtBeginShipoutFirst \@undefined
457 <latexrelease>
458 <latexrelease>\let \ShipoutBoxHeight \@undefined
459 <latexrelease>\let \ShipoutBoxDepth \@undefined
460 <latexrelease>\let \ShipoutBoxWidth \@undefined
461 <latexrelease>

```

We do not undo a substitution when rolling back. As the file support gets undone the underlying data is no longer used (and sufficiently obscure that it should not interfere with existing commands) and properly removing it would mean we need to make the `\undeclare@...` and its support macros available in all earlier kernel releases which is pointless (and actually worse).

```

462 <latexrelease>
463 <latexrelease>\let \AtEndDvi \@undefined

```

We do not reenale a disabled package load when rolling back. As the file support gets undone the underlying data is no longer checked (and sufficiently obscure that it should not interfere with existing commands) and properly removing it would mean we need to make the `\reenable@package@load` command available in all earlier kernel releases which is pointless (and actually worse).

```

464 %\reenable@package@load{atenddvi}
465 <latexrelease>
466 <latexrelease>\EndIncludeInRelease
467 <*2kernel>

```

6 Package emulation for compatibility

6.1 Package `atenddvi` emulation

`\AtEndDvi` This package has only one public command, so simulating it is easy and actually sensible to provide as part of the kernel.

```

468 </2kernel>
469 <*2kernel | latexrelease>
470 <latexrelease>\IncludeInRelease{2020/10/01}%
471 <latexrelease>          {\AtEndDvi}{atenddvi emulation}%
472 \ExplSyntaxOn
473 \cs_new_protected:Npn \AtEndDvi #1 {\AddToHook{shipout/lastpage}{#1}}
474 \ExplSyntaxOff

```

As the package is integrate we prevent loading (no need to roll that back):

```

475 \disable@package@load{atenddvi}
476   {\PackageWarning{atenddvi}
477     {Functionality of this package is already\MessageBreak
478     provided by LaTeX.\MessageBreak\MessageBreak
479     It is there no longer necessary to load it\MessageBreak
480     and you can safely remove it.\MessageBreak
481     Found on}}
482 </2kernel | latexrelease>

```

```

483 <latexrelease>\EndIncludeInRelease
484 <latexrelease>\IncludeInRelease{0000/00/00}%
485 <latexrelease>          {\AtEndDvi}{atenddvi emulation}%
486 <latexrelease>\let \AtEndDvi \@undefined
487 <latexrelease>\EndIncludeInRelease
488 <*2ekernel>

```

(End of definition for \AtEndDvi. This function is documented on page 1127.)

```

489 </2ekernel>

```

6.2 Package atbegshi emulation

```

490 <*atbegshi-ltx>
491 \ProvidesPackage{atbegshi-ltx}
492   [2021/01/10 v1.0c
493   Emulation of the original atbegshi^^Jpackage with kernel methods]

```

\AtBeginShipoutBox

```

494 \let \AtBeginShipoutBox \ShipoutBox

```

(End of definition for \AtBeginShipoutBox. This function is documented on page 1130.)

\AtBeginShipoutInit

Compatibility only, we aren't delaying ...

```

495 \let \AtBeginShipoutInit \@empty

```

(End of definition for \AtBeginShipoutInit. This function is documented on page 1131.)

\AtBeginShipout \AtBeginShipoutNext

Filling hooks

```

496 \protected\long\def\AtBeginShipout   #1{\AddToHook{shipout/before}{#1}}
497 \protected\long\def\AtBeginShipoutNext #1{\AddToHookNext{shipout/before}{#1}}

```

(End of definition for \AtBeginShipout and \AtBeginShipoutNext. These functions are documented on page 1131.)

\AtBeginShipoutFirst

Slightly more complex as we need to know the name of the command under which the shipout/firstpage hook is filled.

```

498 \protected \def \AtBeginShipoutFirst
499   {\@expl@@@shipout@add@firstpage@material@@Nn \AtBeginShipoutFirst}

```

(End of definition for \AtBeginShipoutFirst. This function is documented on page 1131.)

\AtBeginShipoutDiscard

Just a different name.

```

500 \let \AtBeginShipoutDiscard \DiscardShipoutBox

```

(End of definition for \AtBeginShipoutDiscard. This function is documented on page 1131.)

\AtBeginShipoutAddToBox \AtBeginShipoutAddToBoxForeground \AtBeginShipoutUpperLeft \AtBeginShipoutUpperLeftForeground

We don't expose them.

```

501 \let \AtBeginShipoutAddToBox
502       \@expl@@@shipout@add@background@box@@n
503 \let \AtBeginShipoutAddToBoxForeground
504       \@expl@@@shipout@add@foreground@box@@n
505 \let \AtBeginShipoutUpperLeft
506       \@expl@@@shipout@add@background@picture@@n
507 \let \AtBeginShipoutUpperLeftForeground
508       \@expl@@@shipout@add@foreground@picture@@n

```


(End of definition for `\AtBeginShipoutAddToBox` and others. These functions are documented on page 1130.)

`\AtBeginShipoutOriginalShipout` This offers the raw `\shipout` primitive of the engine. A page shipped out with this is not counted by `\ReadonlyShipoutCounter` counter and thus the mechanism to place `\specials` at the very end of the output might fail, etc. It should therefore not be used in new applications but is only provided to allow running legacy code. For new code use the commands provided by the kernel instead.

```
509 \ExplSyntaxOn
510 \cs_new_eq:NN \AtBeginShipoutOriginalShipout \tex_shipout:D
```

(End of definition for `\AtBeginShipoutOriginalShipout`. This function is documented on page 1130.)

`\ShipoutBoxHeight`
`\ShipoutBoxWidth`
`\ShipoutBoxDepth` This is somewhat different from the original in `atbegshi` where `\ShipoutBoxHeight` etc. only holds the `\the\ht<box>` value. This may have some implications in some use cases and if that is a problem then it might need changing.

```
511 \cs_new:Npn \ShipoutBoxHeight { \dim_use:N \l_shipout_box_ht_dim }
512 \cs_new:Npn \ShipoutBoxDepth { \dim_use:N \l_shipout_box_dp_dim }
513 \cs_new:Npn \ShipoutBoxWidth { \dim_use:N \l_shipout_box_wd_dim }
514 \ExplSyntaxOff
```

(End of definition for `\ShipoutBoxHeight`, `\ShipoutBoxWidth`, and `\ShipoutBoxDepth`.)

```
515 </atbegshi-ltx>
```

If the package is requested we substitute the one above:

```
516 <*2kernel>
517 \declare@file@substitution{atbegshi.sty}{atbegshi-ltx.sty}
518 </2kernel>
```

6.3 Package `everyshi` emulation

This is now directly handled in that package so emulation is not necessary any more.

Rather important :-)

```
519 <@@=>
```

File 54

ltoutput.dtx

1 Output Routine and float handling

```
1 <*2kernel>
2 \message{output,}
```

1.1 Historical notes on the algorithm and commands

Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

```
*****
*                OUTPUT                *
*****
```

PAGE LAYOUT PARAMETERS

`\topmargin` : Extra space added to top of page.
`@twoside` : boolean. T if two-sided printing
`\oddsidemargin` : IF `@twoside = T`
THEN extra space added to left of odd-numbered
pages.
ELSE extra space added to left of all pages.
`\evensidemargin` : IF `@twoside = T`
THEN extra space added to left of even-numbered
pages.
`\headheight` : height of head
`\headsep` : separation between head and text
`\footskip` : distance separation between baseline of last
line of text and baseline of foot.
Note difference between `\footSKIP` and `\headSEP`.
`\textheight` : height of text on page, excluding head and foot
`\textwidth` : width of printing on page
`\columnsep` : IF `@twocolumn = T`
THEN width of space between columns
`\columnseprule` : IF `@twocolumn = T`
THEN width of rule between columns (0 if none).
`\columnwidth` : IF `@twocolumn = T`
THEN $(\text{\textwidth} - \text{\columnsep})/2$
ELSE `\textwidth`
It is set by the `\twocolumn` and
`\onecolumn` commands.
`@textbottom` : Command executed at bottom of vbox holding text of
page (including figures). The `\raggedbottom`
command almost `\let`'s this to `\vfil` (actually sets
it to `\vskip \z@ plus.0001fil`).
Should have depth 0pt.

`\@texttop` : Command executed at top of vbox holding text of page (including figures). Used by letter style; can also be used to produce centered pages. Let to `\relax` by `\raggedbottom` and `\flushbottom`.

Page layout must initialize `\@colht` and `\@colroom` to `\textheight`.

PAGE STYLE PARAMETERS:

`\floatsep` : Space left between floats.
`\textfloatsep` : Space between last top float or first bottom float and the text.
`\topfigrule` : Command to place rule (or whatever) between floats at top of page and text. Executed in inner vertical mode right before the `\textfloatsep` skip separating the floats from the text. Must occupy zero vertical space. (See `\footnoterule`.)
`\botfigrule` : Same as `\topfigrule`, but put after the `\textfloatsep` skip separating text from the floats at bottom of page.
`\intextsep` : Space left on top and bottom of an in-text float.
`\dblfloatsep` : Space between double-column floats.
`\dbltextfloatsep` : Space between top double-column floats and text.
`\dblfigrule` : Similar to `\topfigrule`, but for double-column floats.
`\@fptop` : Glue to go at top of float column – must be 0pt + stretch
`\@fpsep` : Glue to go between floats in a float column.
`\@fpbot` : Glue to go at bottom of float column
– must be 0pt + stretch
`\@dblfpsep`, `\@dblfpbot`
: Analogous for double-column float page in two-column format.

FOOTNOTES: As in PLAIN, footnotes use `\insert\footins`.

PAGE LAYOUT SWITCHES AND MACROS

`@twocolumn` : Boolean. T if two columns per page globally.

PAGE STYLE MACROS AND SWITCHES

`\@oddhead` : IF `@twoside = T`
THEN macro to generate head of odd-numbered pages.
ELSE macro to generate head of all pages.
`\@evenhead` : IF `@twoside = T`
THEN macro to generate head of even-numbered

pages.

`\@oddfoot` : IF @twoside = T
THEN macro to generate foot of odd-numbered
pages.
ELSE macro to generate foot of all pages.

`\@evenfoot` : IF @twoside = T
THEN macro to generate foot of even-numbered
pages.

`@specialpage` : boolean. T if current page is to have a special
format.

`\@specialstyle` : If its value is foo then
IF @specialpage = T
THEN the command `\ps@foo` is executed to
temporarily reset the page style parameters
before composing the current page.
This command should execute only `\def`'s and
`\edef`'s, making only local definitions.

FLOAT PLACEMENT PARAMETERS

The following parameters are set by the macro `\@floatplacement`.
When `\@floatplacement` is called,

`\@colht` is the height of the page or column being built. I.e.:

* For single-column page it equals `\textheight`.

* For double-column page it equals `\textheight - height`
of double-column floats on page.

Note that some are set globally and some locally:

`\@topnum` :=G Maximum number of floats allowed on the top of a
column.

`\@toproom` :=G Maximum amount of top of column devoted to floats—
excluding `\textfloatsep` separation below the floats
and `\floatsep` separation between them. For
two-column output, should be computed as a function
of `\@colht`.

`\@botnum`, `\@botroom`
: Analogous to above.

`\@colnum` :=G Maximum number of floats allowed in a column,
including in-text floats.

`\@textmin` :=L Minimum amount of text (excluding footnotes) that
must appear on a text page.

%% 27 Sep 85 : made local to

%% `\@addtocurcol` and `\@addtonextcol`

It is now also used locally in processing double
floats.

`\@fpmin` :=L Minimum height of floats in a float column.

The macro `\@dblfloatplacement` sets the following parameters.

`\@dbltopnum` :=G Maximum number of double-column floats allowed at
the top of a two-column page.

`\@dbltoproom` :=G Maximum height of double-column floats allowed at

top of two-column page.

`\@fpm` :=L Minimum height of floats in a float column.

It should also perform the following local assignments where necessary – i.e., where the new value differs from the old one:

`\@fptop` :=L `\@dblftop`

`\@fpsep` :=L `\@dblfpsep`

`\@fpbot` :=L `\@dblfpbot`

OUTPUT ROUTINE VARIABLES

`\@colht` : The total height of the current column. In single column style, it equals `\textheight`. In two-column style, it is `\textheight` minus the height of the double-column floats on the current page. MUST BE INITIALIZED TO `\textheight`.

`\@colroom` : The height available in the current column for text and footnotes. It equals `\@colht` minus the height of all floats committed to the top and bottom of the current column.

`\@textfloatsheight` : The total height of in-text floats on the current page.

`\@footins` : Footnote insertion number.

`\@maxdepth` : Saved value of TeX's `\maxdepth`. Must be set when any routine sets `\maxdepth`.

CALLING THE OUTPUT ROUTINE

The output routine is called either by TeX's normal page-breaking mechanism, or by a macro putting a penalty $<$ or $= -10000$ in the output list. In the latter case, the penalty indicates why the output routine was called, using the following code.

penalty	reason
-10000	<code>\pagebreak</code> <code>\newpage</code>
-10001	<code>\clearpage (\penalty -10000 \vbox{} \penalty -10001)</code>
-10002	float insertion, called from horizontal mode
-10003	float insertion, called from vertical mode.
-10004	float insertion.

Note: A float or marginpar puts the following sequence in the output list:

- (i) a penalty of -10004,
- (ii) a null `\vbox`
- (iii) a penalty of -10002 or -10003.

This solves two special problems:

1. If the float comes right after a `\newpage` or `\clearpage`, then the first penalty is ignored, but the second one invokes the output routine.
2. If there is a split footnote on the page, the second 'page' puts out the rest of the footnote.

THE OUTPUT ROUTINE

FUNCTIONS USED IN THE OUTPUT ROUTINE:

`\@outputpage` : Produces an output page with the contents of box `\@outputbox` as the text part.

Also sets `\@colht :=G \textheight`.

The page style is determined as follows.

```
IF  \@thispagestyle = true
  THEN use \@thispagestyle style
  ELSE use ordinary page style.
```

`\@tryfcolumn\FLIST` : Tries to form a float column composed of floats from `\FLIST` (if nonempty) with the following parameters:

```
\@colht : height of box
\@fpmin : minimum height of floats in the box
\@fpsep : interfloat space
\@fptop : glue at top of box
\@fpbot : glue at bottom of box.
```

If it succeeds, then it does the following:

```
* \@outputbox :=L the composed float box.
* \@fcolmade :=G true
* \FLIST      :=G \FLIST - floats put in box
* \@freelist  :=G \@freelist + floats put in box
```

If it fails, then:

```
* \@fcolmade :=G false
```

NOTE: BIT MUST BE A SINGLE TOKEN!

`\@makefcolumn \FLIST` : Same as `\@tryfcolumn` except that it fails to make a float column only if `\FLIST` is empty. Otherwise, it makes a float column containing at least the first box in `\FLIST`, disregarding `\@fpmin`.

`\@startcolumn` :

Calls `\@tryfcolumn\@deferlist`. If `\@tryfcolumn` returns with (globally set) `@fcolmade = false`, then:

```
* Globally sets \@toplist and \@botlist to floats
  from \@deferlist to go at top and bottom of column,
  deleting them from \@deferlist. It does
  this using \@colht as the total height, the page
  style parameters \@floatsep and \@textfloatsep, and
  the float placement parameters \@topnum, \@toproom,
  \@botnum, \@botroom, \@colnum and \textfraction.
```

* Globally sets `\@colroom` to `\@colht` minus the height of the added floats.

`\@startdblcolumn` :

Calls `\@tryfcolumn\@dbldeferlist{8}`. If `\@tryfcolumn` returns with (globally set) `@fcolmade = false`, then:

- * Globally sets `\@dbltoplist` to floats from `\@dbldeferlist` to go at top and bottom of column, deleting them from `\@dbldeferlist`. It does this using `\textheight` as the total height, and the parameters `\@dblfloatsep`, etc.
- * Globally sets `\@colht` to `\textheight` minus the height of the added floats.

`\@combinefloats` : Combines the text from box

`\@outputbox` with the floats from `\@toplist` and `\@botlist`, putting the new box in `\@outputbox`. It uses `\floatsep` and `\textfloatsep` for the appropriate separations. It puts the elements of `\TOPLIST` and `\BOTLIST` onto `\freelist`, and makes those lists null.

`\@makecol` : Makes the contents of `\box255` plus the accumulated footnotes, plus the floats in `\@toplist` and `\@botlist`, into a single column of height `\@colht` (unless the page height has been locally changed), which it puts into box `\@outputbox`. It puts boxes in `\@midlist` back onto `\freelist` and restores `\maxdepth`.

`\@opcol` : Outputs a column whose text is in box `\@outputbox`. If `@twocolumn = false`, then it calls `\@outputpage`, sets `\@colht :=G \textheight`, and calls `\@floatplacement`.

If `@twocolumn = true`, then:

If `@firstcolumn = true`, then it puts box `\@outputbox` into `\@leftcolumn` and sets `@firstcolumn :=G false`.

If `@firstcolumn = false`, then it puts out the current two-column page, any possible two-column float pages, and determines `\@dbltoplist` for the next page.

USER COMMANDS THAT CALL OR AFFECT THE OUTPUT
ROUTINE

`\newpage == BEGIN \par\vfil\penalty -10000 END`

`\clearpage == BEGIN \newpage`
`\write -1{}` % Part of hack to make sure no

```

\ vbox{}          % \write's get lost.
\ penalty -10001
END

```

```

\ cleardoublepage == BEGIN \ clearpage
                    if @twoside = true and c@page is even
                    then \ hbox{} \ newpage fi
END

```

`\twocolumn[BOX]` : starts a new page, changing to twocolumn setting and puts BOX in a parbox of width `\textwidth` across the top. Useful for full-width titles for double-column pages.
SURPRISE: The stretch from `\dbltextfloatsep` will be inserted between the BOX and the top of the two columns.

FLOAT-HANDLING MECHANISMS

The float environment obtains an insertion number B from the `\@freelist` (see below for a description of list manipulation), puts the float into box B and sets `\count B` to a FLOAT SPECIFIER. For a normal (not double-column) float, it then causes a page break in one of the following two ways:

- In outer hmode: `\vadjust{\penalty -10002}`
- In vmode : `\penalty -10003`.

For a double-column float, it puts B onto the `\@dbldeferlist`. The float specifier has two components:

- * A PLACEMENT SPECIFICATION, describing where the float may be placed.
- * A TYPE, which is a power of two—e.g., figures might be type 1 floats, tables type 2 floats, programs type 4 floats, etc.

The float specifier is encoded as follows, where bit 0 is the least significant bit.

Bit	Meaning
0	1 iff the float may go where it appears in the text.
1	1 iff the float may go on the top of a page.
2	1 iff the float may go on the bottom of a page.
3	1 iff the float may go on a float page.
4	1 unless the PLACEMENT includes a !
5	1 iff a type 1 float
6	1 iff a type 2 float
	etc.

A negative float specifier is used to indicate a marginal note.

MACROS AND DATA STRUCTURES FOR PROCESSING FLOATS

A FLOAT LIST consisting of the floats in boxes `\boxa ... \boxN` has the form:

```
\@elt \boxa ... \@elt \boxN
```

where `\boxI` is defined by

```
\newinsert\boxI
```

Normally, `\@elt` is `\let` to `\relax`. A test can be performed on the entire float list by locally `\def`'ing `\@elt` appropriately and executing the list.

This is a lot more efficient than looping through the list.

The following macros are used for manipulating float lists.

```
\@next \CS \LIST {NONEMPTY}{EMPTY} ==  %% NOTE: ASSUME \@elt
= \relax
  BEGIN  assume that \LIST == \@elt \B1 ... \@elt \Bn
        if n = 0
          then  EMPTY
        else  \CS      :=L \B1
              \LIST :=G \@elt \B2 ... \@elt \Bn
              NONEMPTY
        fi
  END
```

`\@bitor\NUM\LIST` : Globally sets switch `@test` to the disjunction for all `I` of bit `log2 \NUM` of the float specifiers of all the floats in `\LIST`.
 I.e., `@test` is set to true iff there is at least one float in `\LIST` having bit `log2 \NUM` of its float specifier equal to 1.

Note: `log2 [(\count I)/32]` is the bit number corresponding to the type of float `I`. To see if there is any float in `\LIST` having the same type as float `I`, you run `\@bitor` with
`\NUM = [(\count I)/32] * 32`.

```
\@bitor\NUM\LIST ==
  BEGIN
    @test :=G false
    { \@elt \CTR ==  if \NUM <> 0 then
                      if \count\CTR / \NUM is odd
                        then @test := true      fi fi
    }
  END
```

`\@cons\LIST\NUM` : Globally sets `\LIST := \LIST * \@elt \NUM`

```

\@cons\LIST\NUM ==
  BEGIN { \@elt == \relax
          \LIST :=G \LIST \@elt \NUM
        }

```

BOX LISTS FOR FLOAT-PLACEMENT ALGORITHMS

```

\@freelist      : List of empty boxes for placing new floats.
\@toplist       : List of floats to go at top of current column.
\@midlist       : List of floats in middle of current column.
\@botlist       : List of floats to go at bottom of current column.
\@deferlist     : List of floats to go after current column.
\@dbltoplist    : List of double-col. floats to go at top of current
                  page.
\@dbldeferlist  : List of double-column floats to go on subsequent
                  pages.

```

FLOAT-PLACEMENT ALGORITHMS

```

\@addtobot : Tries to put insert \@currbox on \@botlist.
             Called only when:
               * \ht BOX < \@colroom
               * type of \@currbox not on \@deferlist
               * \@colnum > 0
               * @insert = false
             If it succeeds, then:
               * sets @insert true
               * decrements \@botroom by \ht BOX
               * decrements \@botnum and \@colnum by 1
               * decrements \@colroom by \ht BOX + either \floatsep
                 or \textfloatsep, as appropriate.
               * sets \maxdepth to 0pt

\@addtotoporbot : Tries to put insert \@currbox on \@toplist or
                  \@botlist.
                  Called only under same conditions as \@addtobot.
                  If it succeeds, then:
                    * sets @insert true
                    * decrements \@toproom or \@botroom by \ht BOX
                    * decrements \@colnum and either \@topnum or
                      \@botnum by 1
                    * decrements \@colroom by \ht BOX + \floatsep
                      or \textfloatsep, as appropriate.

\@addtocurcol : Tries to add \@currbox to current column, setting
                 @insert true if it succeeds, false otherwise.
                 It will add \@currbox to top only if bit 0 of
                 \count \@currbox is 0, and to the bottom only if

```

bit 0 = 0 or an earlier float of the same type is put on the bottom.
 If the float is put in the text, then $\backslash\penalty\interlinepenalty$ is put right after the float, before the following $\backslash\vskip$, and $\backslash\outputpenalty :=L 0$.

$\backslash\@addtonextcol$: Tries to add $\backslash\@currbox$ to the next column, setting $\@insert$ true if it succeeds, false otherwise.

$\backslash\@addtodblcol$: Tries to add $\backslash\@currbox$ to the next double-column page, adding it to $\backslash\@dbltoplist$ if it succeeds and $\backslash\@dbldeferlist$ if it fails.

```

\@addmarginpar ==
BEGIN
  if \@currlist nonempty
    then remove \@marbox from \@currlist
      add \@marbox and \@currbox to \@freelist
      %% NOTE: \@currbox = left box
    else LaTeX error: ? %% shouldn't happen
  fi
  \@tempcnta := 1      %% 1 = right, -1 = left
  if @twocolumn = true
    then if @firstcolumn = true
      then \@tempcnta := -1
        fi
      else if @mparswitch = true
        then if count0 odd
          else \@tempcnta := -1
            fi
          fi
        if @reversemargin = true
          then \@tempcnta := -\@tempcnta
            fi
        fi
    if \@tempcnta < 0 then \box\@marbox :=G \box\@currbox
    fi
  \@tempdima :=L maximum(\@mparbottom - \@pageht
                        + ht of \@marbox, 0)
  if \@tempdima > 0 then LaTeX warning: 'marginpar moved' fi
  \@mparbottom :=G \@pageht + \@tempdima + depth of \@marbox
                + \marginparpush
  \@tempdima :=L \@tempdima - ht of \@marbox
  \box\@marbox :=G \box\@currbox
                    \vbox { \vskip \@tempdima
                          \box\@marbox
                          }
  height of \@marbox :=G depth of \@marbox :=G 0

```

```

\kern -\@pagedp
\nointerlineskip
\hbox{ if @tempcnta > 0 then \hskip \columnwidth
      \hskip \marginparsep
      else \hskip -\marginparsep
          \hskip -\marginparwidth
      fi
      \box\@marbox \hss
    }
\nobreak
\nointerlineskip
\hbox{\vrule height 0 width 0 depth \@pagedp}
END

```

End of historical L^AT_EX 2.09 comments.

1.2 Core definitions

Floats and marginpars add a lot of dead cycles.

```

3 \maxdeadcycles = 100
4 \let\@elt\relax
5 \def\@next#1#2#3#4{\ifx#2\empty #4\else
6   \expandafter\@xnext #2\@#1#2#3\fi}
7 \def\@xnext \@elt #1#2\@#3#4{\def#3{#1}\gdef#4{#2}}
8 \def\@testfalse{\global\let\if@test\iffalse}
9 \def\@testtrue {\global\let\if@test\iftrue}
10 \@testfalse
11 \def\@bitor#1#2{\@testfalse {\let\@elt\@xbitor
12   \@tempcnta #1\relax #2}}

```

RmS 91/11/22: Added test for \count#1 = 0. Suggested by Chris Rowley.

```

13 \def\@xbitor #1{\@tempcntb \count#1
14   \ifnum \@tempcnta =\z@
15   \else
16     \divide\@tempcntb\@tempcnta
17     \ifodd\@tempcntb \@testtrue\fi
18   \fi}

```

1.2.1 Definition of float boxes

```

19 </2kernel>
20 <latexrelease>\IncludeInRelease{2015/10/01}%
21 <latexrelease>           {\bx@ZZ}{Extended float list}%
22 <*2kernel | latexrelease>
23 \let\@elt\newinsert
24 <*2kernel>
25 \def\@freelist{%
26   \@elt\bx@A\@elt\bx@B\@elt\bx@C\@elt\bx@D\@elt\bx@E
27   \@elt\bx@F\@elt\bx@G\@elt\bx@H\@elt\bx@I\@elt\bx@J
28   \@elt\bx@K\@elt\bx@L\@elt\bx@M\@elt\bx@N

```

```

29 \elt\bx@O\elt\bx@P\elt\bx@Q\elt\bx@R}
30 \@freelist
31 \</2ekernel)
32 \ifx\numexpr\@undefined\else
33 \def\reserved@a{%
34 \elt\bx@S\elt\bx@T\elt\bx@U\elt\bx@V
35 \elt\bx@W\elt\bx@X\elt\bx@Y\elt\bx@Z
36 \elt\bx@AA\elt\bx@BB\elt\bx@CC\elt\bx@DD\elt\bx@EE
37 \elt\bx@FF\elt\bx@GG\elt\bx@HH\elt\bx@II\elt\bx@JJ
38 \elt\bx@KK\elt\bx@LL\elt\bx@MM\elt\bx@NN
39 \elt\bx@OO\elt\bx@PP\elt\bx@QQ\elt\bx@RR
40 \elt\bx@SS\elt\bx@TT\elt\bx@UU\elt\bx@VV
41 \elt\bx@WW\elt\bx@XX\elt\bx@YY\elt\bx@ZZ}
42 \reserved@a
43 \def\elt{\noexpand\elt\noexpand}
44 \edef\@freelist{\@freelist\reserved@a}
45 \fi
46 \let\reserved@a\relax
47 \let\elt\relax
48 \</2ekernel | latexrelease)
49 \<latexrelease)\EndIncludeInRelease
50 \<latexrelease)\IncludeInRelease{0000/00/00}%
51 \<latexrelease) \{ \bx@ZZ } {Extended float list}%
52 \<latexrelease)\def\@freelist{%
53 \<latexrelease) \elt\bx@A\elt\bx@B\elt\bx@C\elt\bx@D\elt\bx@E
54 \<latexrelease) \elt\bx@F\elt\bx@G\elt\bx@H\elt\bx@I\elt\bx@J
55 \<latexrelease) \elt\bx@K\elt\bx@L\elt\bx@M\elt\bx@N
56 \<latexrelease) \elt\bx@O\elt\bx@P\elt\bx@Q\elt\bx@R}
57 \<latexrelease) \insecunt=234
58 \<latexrelease)\EndIncludeInRelease
59 \*2ekernel)

60 \gdef\@toplist{}
61 \gdef\@botlist{}
62 \gdef\@midlist{}
63 \gdef\@currlist{}
64 \gdef\@deferlist{}
65 \gdef\@dbltoplist{}

```

The new algorithm stores page wide floats together with column floats in a single \@deferlist list. We keep \@dbldeferlist initialised as empty so that packages that are testing for deferred floats can use the same code for old or new float handling.

```
66 \gdef\@dbldeferlist{}

```

1.2.2 Page layout parameters

```

67 \newdimen\topmargin
68 \newdimen\oddsidemargin
69 \newdimen\evensidemargin
70 \let\themargin=\oddsidemargin
71 \newdimen\headheight
72 \newdimen\headsep
73 \newdimen\footskip
74 \newdimen\textheight
75 \newdimen\textwidth
76 \newdimen\columnwidth

```

```

77 \newdimen\columnsep
78 \newdimen\columnseprule
79 \newdimen\marginparwidth
80 \newdimen\marginparsep
81 \newdimen\marginparpush

```

`\AtBeginDvi` We use a box register in which to put stuff that must appear before anything else in the `.dvi` file.

The stuff in the box should not add any typeset material to the page when it is unboxed.

This interface is no longer used. Instead a new one is inside `ltshipout.dtx`. We only keep the box in case some old code refers to it directly (or we do some rollback).

```

82 \newbox\@begindvibox
83 %\DeclareRobustCommand \AtBeginDvi [1]{%
84 %   \global \setbox \@begindvibox
85 %     \vbox{\unvbox \@begindvibox #1}%
86 %}

```

(End of definition for `\AtBeginDvi` and `\@begindvibox`. These functions are documented on page 1127.)

`\@maxdepth` This is not the right place to set this; it needs to be set in a class/style file when `\maxdepth` is set.

Also, many settings to `\maxdepth` should be to `\@maxdepth`, probably?

```

87 \newdimen\@maxdepth
88 \@maxdepth = \maxdepth

```

(End of definition for `\@maxdepth`.)

`\paperheight` New `\paper...` registers.

```

\paperwidth 89 \newdimen\paperheight
90 \newdimen\paperwidth

```

(End of definition for `\paperheight` and `\paperwidth`.)

`\stockheight` New `\stock...` registers.

```

\stockwidth 91 \newdimen\stockheight
92 \newdimen\stockwidth

```

(End of definition for `\stockheight` and `\stockwidth`.)

`\if@insert` Local switches first:

```

\if@fcolmade 93 \newif \if@insert

```

`\if@specialpage` These should definitely be global:

```

\if@firstcolumn 94 \newif \if@fcolmade
\if@twocolumn 95 \newif \if@specialpage \@specialpagefalse
\if@twoside

```

`\if@reversemargin` These should be global but are not always set globally in other files.

```

\if@mparswitch 96 \newif \if@firstcolumn \@firstcolumntrue
\col@number 97 \newif \if@twocolumn \@twocolumnfalse

```

Not sure about these: two questions. Should things which must apply to a whole document be local or global (they probably should be ‘preamble only’ commands)? Are these three such things?

```

98 \newif \if@twoside      \@twosidefalse
99 \newif \if@reversemargin \@reversemarginfalse
100 \newif \if@mparswitch \@mparswitchfalse

```

This counter has been imported from ‘multicol’.

```

101 \newcount \col@number
102 \col@number \@ne

```

(End of definition for \if@insert and others.)

1.2.3 Internal registers

```

103 \newcount\@topnum
104 \newdimen\@toproom
105 \newcount\@dbltopnum
106 \newdimen\@dbltoproom
107 \newcount\@botnum
108 \newdimen\@botroom
109 \newcount\@colnum
110 \newdimen\@textmin
111 \newdimen\@fpmin
112 \newdimen\@colht
113 \newdimen\@colroom
114 \newdimen\@pageht
115 \newdimen\@pagedp
116 \newdimen\@mparbottom \@mparbottom\z@
117 \newcount\@currtype
118 \newbox\@outputbox
119 \newbox\@leftcolumn
120 \newbox\@holdpg
121 \def\@thehead{\@oddhead} % initialization
122 \def\@thefoot{\@oddfoot}

```

1.2.4 Page break commands

`\clearpage` The tests at the beginning are an experimental attempt to avoid a completely empty page after a `\twocolumn[...]`. This prevents the text from the argument vanishing into a float box, never to be seen again. We hope that it does not produce wrong formatting in other cases.

```

123 \def\clearpage{%
124   \ifvmode
125     \ifnum \@dbltopnum =\m@ne
126       \ifdim \pagetotal <\topskip
127         \hbox{}%
128       \fi
129     \fi
130   \fi
131   \newpage
132   \write\m@ne{}%
133   \vbox{}%
134   \penalty -\@Mi
135 }

```

(End of definition for `\clearpage`.)

`\cleardoublepage`

```
136 \def\cleardoublepage{\clearpage\if@twoside \ifodd\c@page\else
137   \hbox{}\newpage\if@twocolumn\hbox{}\newpage\fi\fi\fi}
138 \endkernel
```

(End of definition for `\cleardoublepage`.)

`\onecolumn`

```
139 \if@twocolumn\if@float\if@floatplacement
140 \def\onecolumn{%
141   \clearpage
142   \global\columnwidth\textwidth
143   \global\hsize\columnwidth
144   \global\linewidth\columnwidth
145   \global\@twocolumnfalse
146   \col@number \@one
147   \@floatplacement}
```

(End of definition for `\onecolumn`.)

`\newpage` The two checks at the beginning ensure that an item label or run-in section title immediately before a `\newpage` get printed on the correct page, the one before the page break.

All three tests are largely to make error processing more robust; that is why they all reset the flags explicitly, even when it would appear that this would be done by a `\leavevmode`.

```
148 \endkernel\if@float\if@floatplacement
149 \if@float\if@floatplacement\if@floatplacement
150 \if@float\if@floatplacement\if@floatplacement
151 \if@float\if@floatplacement\if@floatplacement
152 \def \newpage {%
153   \if@noskipsec
154     \ifx \@nodocument\relax
155       \leavevmode
156       \global \@noskipsecfalse
157     \fi
158   \fi
159   \if@inlabel
160     \leavevmode
161     \global \@inlabelfalse
162   \fi
163   \if@nobreak \@nobreakfalse \everypar{}\fi
164   \par
```

The `\vfil` at the end of the macro before the break penalty will normally result in the page being run short, even with `\flushbottom` in effect (in contrast to the behavior of `\pagebreak`). However, if there is some explicit stretch on the page, say, a `\vfill`, it has the undesired side-effect, that the last line will not align at its baseline if it contains characters going below the baseline, as the value of `\prevdepth` is no longer taken into account by T_EX. So we back up by that amount (or by `\maxdepth` if it is really huge), to mimic the normal behavior without the `\newpage`.

```
165 \ifdim\prevdepth>\z@
```



```

166     \vskip -%
167         \ifdim\prevdepth>\maxdepth
168             \maxdepth
169         \else
170             \prevdepth
171         \fi
172     \fi
173     \vfil
174     \penalty -\@M}
175 </2ekernel | latexrelease | fltrace>
176 <latexrelease>\EndIncludeInRelease
177 <latexrelease>\IncludeInRelease{0000/00/00}%
178 <latexrelease>        {\newpage}{Check depth of page}%
179 <latexrelease>\def \newpage {%
180 <latexrelease>     \if@noskipsec
181 <latexrelease>       \ifx \@nodocument\relax
182 <latexrelease>         \leavevmode
183 <latexrelease>         \global \@noskipsecfalse
184 <latexrelease>       \fi
185 <latexrelease>     \fi
186 <latexrelease>     \if@inlabel
187 <latexrelease>       \leavevmode
188 <latexrelease>       \global \@inlabelfalse
189 <latexrelease>     \fi
190 <latexrelease>     \if@nobreak \@nobreakfalse \everypar{}\fi
191 <latexrelease>     \par
192 <latexrelease>     \vfil
193 <latexrelease>     \penalty -\@M}
194 <latexrelease>\EndIncludeInRelease
195 <*2ekernel | fltrace>

```

(End of definition for \newpage.)

\@emptycol It may be better to use an invisible rule rather than an empty box here.

```

196 \def \@emptycol {\vbox{} \penalty -\@M}

```

(End of definition for \@emptycol.)

\twocolumn There are several bug fixes to the two-column stuff here.
\@topnewpage

```

197 \def \twocolumn {%
198     \clearpage
199     \global\columnwidth\textwidth
200     \global\advance\columnwidth-\columnsep
201     \global\divide\columnwidth\tw@
202     \global\hsize\columnwidth
203     \global\linewidth\columnwidth
204     \global\@twocolumntrue
205     \global\@firstcolumntrue
206     \col@number \tw@

```

There is no reason to put a `\@dblfloatplacement` here since `\@topnewpage` ignores these settings. The `\@floatplacement` is needed in case this comes after some changes.

```

207     \ifnextchar [\@topnewpage\@floatplacement
208     }

```

Note that here, getting a box from the freelist can assume success since this comes just after a `\clearpage`.

```

209 \long\def \@topnewpage [#1]{%
210   \nodocument
211   \@next\@currbox\@freelist{}-{}%
212   \global \setbox\@currbox
213     \color@vbox
214     \normalcolor
215     \vbox {%
216       \hsize\textwidth
217       \@parboxrestore
218       \col@number \@ne
219       #1%
220       \vskip -\dbltextfloatsep
221       }%
222   \color@endbox

```

Added size test and warning message; perhaps we should use an error message.

```

223   \ifdim \ht\@currbox>\textheight
224     \ht\@currbox \textheight
225   \fi

```

This next line is not essential but it is more robust to make this value non-zero, in case of weird errors.

This next bit is what is needed from `\@addtodblcol`, plus some extra checks for error trapping.

```

226   \global \count\@currbox \tw@
227   \@tempdima -\ht\@currbox
228   \advance \@tempdima -\dbltextfloatsep
229   \global \advance \@colht \@tempdima
230   \ifx \@dbltoplist \@empty
231   \else
232     \@latex@error{Float(s) lost}\@ehb
233     \let \@dbltoplist \@empty
234   \fi
235   \@cons \@dbltoplist \@currbox

```

This setting of `\@dbltopnum` is used only to change the typesetting in `\@combinedblfloats`.

```

236   \global \@dbltopnum \m@ne
237   <*trace>
238     \fl@trace{dbltopnum set to -1 (= \the \@dbltopnum) (topnewpage)}%
239   </trace>

```

At points such as this we need to check that there is still a minimal amount of room left on the page; this uses an arbitrary small value at present; but note that this value is larger than that used when checking that page is too full of normal floats.

If there is little room left we just force a page-break, OK? This involves producing two empty columns. The second empty column may be produced by `\output`, in which case an extra, misleading, warning will be generated, OK? (This happens only when there is too little room left on the page for any float.) Otherwise (i.e. if the size is such that it is allowed as a normal float) the extra `\@emptycol` will be invoked in the second column by the conditional code guarded by the `\if@firstcolumn` test.

I now think that the cut-off point here should be 3\baselineskip , but we make it a bit less so that 3 lines of text will be allowed, OK?

Since this happens only when there is nothing on the page but the ‘top-box’, the empty box should not cause any problem other than some overfull box messages, which is not entirely misleading.

Here we need two page-ends since both columns need to be empty.

```
240 \ifdim \@colht<2.5\baselineskip
241   \latex@warning@no@line {Optional argument of \noexpand\twocolumn
242     too tall on page \thepage}%
243   \@emptycol
244   \if@firstcolumn
245   \else
246     \@emptycol
247   \fi
248 \else
249   \global \vsize \@colht
250   \global \@colroom \@colht
251   \@floatplacement
252 \fi
253 }
```

(End of definition for `\twocolumn` and `\topnewpage`.)

2 The L^AT_EX output routine

2.1 Hooks and replaceable code blocks

To support packages that want to augment aspects of the output routine we offer a number of hooks (where several packages can add code) as well as some sockets (that can only be changed by one package or through options like those of the `footmisc` package).

2.1.1 Output routine hooks

build/page/before, build/page/after These two hooks enable packages to prepend or append code to the page processing in the output routine. They are implemented as mirrored hooks.

Technically, they are executed at the start and the end of the internal L^AT_EX 2_ε `\@outputpage` command, respectively.

build/page/reset Packages that set up special conventions for text in the main galley (such as catcode changes, etc.) can use this hook to undo these changes within the output routine, so that they aren’t applied to unrelated material, e.g., the text for running header or footers.

build/column/before, build/column/after These two hooks enable packages to prepend or append code to the column processing in the output routine. They are implemented as mirrored hooks.

Technically, they are executed at the start and the end of the internal L^AT_EX 2_ε `\@makecol` command, respectively.

2.1.2 Replaceable code blocks (sockets)

To cater for different layouts with respect to text, footnotes, and bottom-floats placements there are sockets for now. They are sockets not hooks, because the overall layout can only be controlled by one package, i.e., the last setting wins.

build/column/outputbox (0 arguments) In code for this socket the `\@outputbox` (holding the galley text for the current column or page) is augmented by attaching floats and footnote areas together with appropriate spacing.

Prior to calling the socket the output routine has already decided which floats go into which area and which get deferred. Therefore, the assumption is that the code in the socket attaches all areas that contain floats. If this is not done, then the order of floats is likely to be screwed up unless unused floats are moved to the defer list in an appropriate way (for now we don't offer any interface for that scenario).

Before the code in the socket is run, any existing glue at the bottom of the `\@outputbox` is removed and stored in a safe place. If needed, it can be reinserted with one of the helper commands.

To support setting this up the following helper commands are available:

`\@outputbox@append` (1 argument)

This general purpose command alters the `\@outputbox` box by appending material to it.

`\@outputbox@appendfootnotes` (0 arguments)

This command appends the footnotes to the `\@outputbox` (if there are any). If not, then it does nothing.

`\@outputbox@attachfloats` (0 arguments)

`\@outputbox@attachtopfloats` (0 arguments)

`\@outputbox@attachbottomfloats` (0 arguments)

Attaching top and bottom floats can usually be done in one go, but for special layouts we might want more control so we provide also separate commands.

`\@outputbox@reinsertbskip` (0 arguments)

Reinsert the bottom skip of the `\@outputbox` that was saved before in `\@makecol`.

Testing for existence of material

There are a number of helpers to run conditional code depending on whether or not there are footnotes or bottom floats. They are `\@if@footnotes@TF` and `\@if@bottomfloats@TF` (names are likely to change).

This socket cannot be empty but needs appropriate code; a set of suitable plugs for it are already given in the kernel. These are

space-footnotes-floats After the galley text there is a vertical `\vfill` followed by any footnotes followed by the bottom floats, if any.

footnotes-space-floats As before but the `\vfill` is between footnotes and floats.

floats-space-footnotes Floats are directly after the text, then a space and then footnotes at the bottom.

space-floats-footnotes Both floats and footnotes are pushed to the bottom with footnotes last.⁶²

floats-footnotes All excess space is distributed across the existing glue on the page, e.g., within the text galley, the separation between blocks, etc. The order is text, floats, footnotes.

footnotes-floats As the previous one but floats and footnotes are swapped. This is the L^AT_EX default for newer document, i.e., this plug is assigned to the socket when `\DocumentMetadata` is used.

footnotes-floats-legacy As the previous one but L^AT_EX's bottom skip bug is not corrected, i.e., in ragged bottom designs where footnotes are supposed to be directly attached to the text, they suddenly appear at the bottom of the page when the page is ended with `\newpage` or `\clearpage`. While this is clearly a bug, it was the case since the days of L^AT_EX 2.09; thus for compatibility we continue to support this behavior.

build/column/footnotes (0 arguments) This socket is used to manipulate the footnote material inside `\box\footins`. If it contains code, it is supposed to do some processing of that box and then write the result back into it (and nothing else!). By default it does nothing, i.e., has the `noop` assigned.

If (short) footnotes are run as a paragraph this socket gets the plug `para` assigned which is defined elsewhere.

2.1.3 Tagging sockets

The following sockets are used to implement tagging. They are used via `\UseTaggingSocket` which turns them off when tagging is disabled. For each of them a `default` plug is implemented that holds the tagging code.

build/column/outputbox (0 arguments) This socket is used to add any missing tagging structures to the `\@outputbox` box, if necessary.

build/column/footins (0 arguments) This socket is used to add any missing tagging structures to `\footins` box, if necessary.

build/page/header (2 arguments) This socket receives the content of the formatted page header as its second argument and adds the necessary tagging around it. The first argument is empty as no special setup is necessary.

build/page/footer (2 arguments) This socket receives the content of the formatted page footer as its second argument and adds the necessary tagging around it. The first argument is empty as no special setup is necessary.

2.1.4 Output routine commands

`\output` This needs some small adjustments. We cannot guarantee that the float mechanism will interact correctly with this stuff, but that mechanism does not always work properly with footnotes already.

RmS 91/09/29:

⁶²There are two more permutations, but neither of them has ever been requested so they aren't set up by default — doing that in a class would be trivial though.

added reset of `\par` to the output routine. This avoids problems when the output routine is called within a list where `\par` may be a no-op.

```

254 \output {%
255   \let \par \@par
256   \ifnum \outputpenalty<-\@M
257     \specialoutput
258   \else
259     \@makecol
260     \opcol

```

Moved to `\opcol`: `\floatplacement`.

```

261   \@startcolumn

```

This loop could be replaced by an `\expandafter` tail recursion in `\startcolumn`.

```

262   \@whilesw \if@fcolmade \fi
263   {%
264   (*trace)
265     \fl@trace{PAGE: float \if@twocolumn column \else page \fi
266               completed}%
267   (/trace)
268     \@opcol\@startcolumn}%
269   \fi
270   \ifnum \outputpenalty>-\@Miv

```

At points such as this we need to check that there is still a minimal amount of room left on the page; this uses an arbitrary small value at present. If there is little room left we just force a page-break, OK?

This bit is essential only if a float has just been processed so maybe it should be moved; but this is the natural place at which to set the `vsize` and a test would need to be done anyway. A check has been added to ensure that there really has been a change in the value of `\colroom`.

Since this happens only when there is nothing on the page but floats, the empty box should not cause any problem other than some overfull box messages, which is not entirely misleading.

The `twocolumn` case does not need any extra code here since this is the `\output` itself; in the second column there will still not be enough room left so `\emptycol` will be executed again when the OR is called by the page builder when it gets to the penalty inserted by the first execution. (The page-builder is never invoked whilst the OR is being executed since it builds an inner `vlist`; thus any conditional code for the two-column case within `\output` may not get executed with the correct value of `\if@firstcolumn`.

```

271   \ifdim \@colroom<1.5\baselineskip
272     \ifdim \@colroom<\textheight
273       \@latex@warning@no@line {Text page \thepage\space
274                               contains only floats}%
275     \emptycol
276   %   \if@twocolumn
277   %     \if@firstcolumn
278   %       \else
279   %         \emptycol
280   %       \fi
281   %     \fi
282   \else
283     \global \vsize \@colroom

```

```

284     \fi
285     \else
286       \global \vsize \@colroom
287     \fi
288   \else
289     \global \vsize \maxdimen
290   \fi
291 }

```

Historical L^AT_EX 2.09 comments (not necessarily accurate any more):

CHANGES TO \@specialoutput:

* \penalty\z@ changed to \penalty\interlinepenalty so \samepage works properly with figure and table environments.

(Changed 23 Oct 86)

* Definition of \@specialoutput changed 26 Feb 88 so \@pageht and \@pagedp aren't changed for a marginal note.

(Change suggested by Chris Rowley.)

End of historical L^AT_EX 2.09 comments.

```

292 \gdef\@specialoutput{%
293   \ifnum \outputpenalty>-\@Mii
294     \doclearpage
295   \else
296     \ifnum \outputpenalty<-\@Miii
297       \ifnum \outputpenalty<-\@MM \deadcycles \z@ \fi
298       \global \setbox\@holdpg \vbox {\unvbox\@cc1v}%
299     \else

```

Note that \boxmaxdepth should not be set here since we wish to record the natural depth of the holdpg box.

This is changed so as to not lose anything, such as writes and marks, which may get into box 255 and should be returned to the list. This should only happen when the first penalty in the mechanism is discarded and therefore \@holdpg should always be void in this case. This can happen because a penalty is discarded whenever there is no box on the list.

It was just: \setbox\@tempboxa \box \@cc1v.

The last box which is removed is the box put there by the double-penalty mechanism. The \unskip then removes the \topskip which is put there since the box is the first on the page.

```

300     \global \setbox\@holdpg \vbox{%
301       \unvbox\@holdpg
302     \unvbox\@cc1v

```

We must now remove the box added by the float mechanism and the \topskip glue therefore added above it by T_EX.

```

303       \setbox\@tempboxa \lastbox
304       \unskip
305     }%

```

These two are needed as separate dimensions only by \@addmarginpar; for other purposes we put the whole size into \@pageht (see below).

```

306     \@pagedp \dp\@holdpg
307     \@pageht \ht\@holdpg

```

```

308     \unvbox \@holdpg
309     \@next\@currbox\@currlist{%
310     \ifnum \count\@currbox>\z@

```

Putting the whole size into \@pageht (see above).

```

311     \advance \@pageht \@pagedp
312     \ifvoid\footins \else
313     \advance \@pageht \ht\footins
314     \advance \@pageht \skip\footins
315     \advance \@pageht \dp\footins
316     \fi
317     \ifvbox \@kludgeins

```

We want to make the adjustment due to this insert only if the non-star form is used. The *-form will probably not work with floats, but maybe it still could make some adjustment here even so?

```

318     \ifdim \wd\@kludgeins=\z@
319     \advance \@pageht \ht\@kludgeins
320     \*trace
321     \fl@trace {Extra size added: \the \ht\@kludgeins}%
322     \*trace
323     \fi
324     \fi

```

This version puts the inserts back just before the additional material; it could be moved earlier, before unboxing the page-so-far. Neither is guaranteed not to put things on the wrong page. This version is similar to the original version.

```

325     \@reinserts
326     \@addtocurcol
327     \else
328     \@reinserts
329     \@addmarginpar
330     \fi
331     }\@latexbug

```

A 2e change: use \addpenalty instead of \penalty here. Some penalty is needed to create a potential break-point immediately after the reinserts (or the marginal). Otherwise there can be no possibility to break here and this can cause the reinserts or the marginal to appear on the next page (which is often incorrect). However, if the nobreak flag is true, a \nobreak must be correct.

```

332     \ifnum \outputpenalty<\z@
333     \if@nobreak
334     \nobreak
335     \else
336     \addpenalty \interlinepenalty
337     \fi
338     \fi
339     \fi
340     \fi
341     }
342     \*trace

```

(End of definition for \output and \specialoutput.)

`\@testwrongwidth` `\f@depth` Test if the float box has the wrong width when trying to place it into some area. (Actually the test is for a conventional depth setting rather than for the width of the float. For that reason the box depth was explicitly tailored when the float was created).

```

343 <latexrelease>\IncludeInRelease{2015/01/01}%
344 <latexrelease>                {\@testwrongwidth}{float order in 2-column}%
345 <*2ekernel | latexrelease | fltrace>

346 \def\@testwrongwidth #1{%
347   \ifdim\dp#1=\f@depth
348   <*trace>
349     \fl@trace{\string#1
350               \ifdim\f@depth=\z@ single \else double \fi
351               column float -- ok}%
352 </trace>
353   \else
354     \global\@testtrue
355   <*trace>
356     \fl@trace{\string#1
357               \ifdim\f@depth=\z@ double \else single \fi
358               column float -- wrong}%
359 </trace>
360   \fi}%

361 \let\f@depth\z@

362 </2ekernel | latexrelease | fltrace>
363 <latexrelease>\EndIncludeInRelease
364 <latexrelease>\IncludeInRelease{0000/00/00}%
365 <latexrelease>                {\@testwrongwidth}{float order in 2-column}%
366 <latexrelease>\let\@testwrongwidth\@undefined
367 <latexrelease>\let\f@depth\@undefined
368 <latexrelease>\EndIncludeInRelease

```

(End of definition for \@testwrongwidth and \f@depth.)

`\@docclearpage` This is a very much an emergency action, just dumping everything: footnotes first then floats. A more sophisticated version is needed; but even more urgent is a bug-free version (see, for example, pr/3528).

Also, it puts any left-over non-boxes (writes, specials, etc.) back after any float pages created: this is a very bad bug since, for example, a kludge insert will be in quite the wrong place and, worse, be irremovable and uncancelable.

All the remaining changes are replacing the double column defer list or inserting the extra test `\@testwrongwidth{<box>}` at suitable places. That is at places where a box is taken off the deferlist.

```

369 <latexrelease>\IncludeInRelease{2015/01/01}{\@docclearpage}%
370 <latexrelease>                {float order in 2-column}%
371 <*2ekernel | latexrelease>

372 \def \@docclearpage {%
373   \ifvoid\footins

374     \ifvbox\@kludgeins
375       {\setbox \@tempboxa \box \@kludgeins}%
376   <*trace>
377     \fl@trace {kludgeins box made void}%

```

```

378 </trace>
379 \fi
380 \setbox\@tempboxa\vsplit\@cclv to\z@ \unvbox\@tempboxa
381 \setbox\@tempboxa\box\@cclv
382 \xdef\@deferlist{\@toplist\@botlist\@deferlist}%
383 \global \let \@toplist \@empty
384 \global \let \@botlist \@empty
385 \global \@colroom \@colht
386 \ifx \@currlist\@empty
387 \else
388 \latexerror{Float(s) lost}\@ehb
389 \global \let \@currlist \@empty
390 \fi
391 \@makefcolumn\@deferlist
392 \@whilesw\if@colmade \fi{\@opcol\@makefcolumn\@deferlist}%
393 \if@twocolumn
394 \if@firstcolumn
395 \xdef\@deferlist{\@dbltoplist\@deferlist}%
396 \global \let \@dbltoplist \@empty
397 \global \@colht \textheight
398 \begingroup
399 \dblfloatplacement
400 \@makefcolumn\@deferlist
401 \whilesw\if@colmade \fi{\@outputpage
402 \@makefcolumn\@deferlist}%
403 \endgroup
404 \else
405 \vbox{}\clearpage
406 \fi
407 \fi

```

the next line is needed to avoid losing floats in certain circumstances a single call to the original `\docclearpage` will now no longer output all floats.

```

408 \ifx\@deferlist\@empty \else\clearpage \fi
409 \else
410 \setbox\@cclv\vbox{\box\@cclv\vfil}%
411 \@makecol\@opcol
412 \clearpage
413 \fi
414 }%
415 </2kernel | latexrelease>
416 <latexrelease>\EndIncludeInRelease
417 <latexrelease>\IncludeInRelease{0000/00/00}{\@docclearpage}%
418 <latexrelease> {float order in 2-column}%
419 <latexrelease>\def \@docclearpage {%
420 <latexrelease> \ifvoid\footins
421 <latexrelease> \ifvbox\@kludgeins
422 <latexrelease> {\setbox \@tempboxa \box \@kludgeins}%
423 <*trace>

```

```

424 <latexrelease>         \fl@trace {kludgeins box made void}%
425 </trace>
426 <latexrelease>         \fi
427 <latexrelease>         \setbox\@tempboxa\vsplit\@cclv to\z@ \unvbox\@tempboxa
428 <latexrelease>         \setbox\@tempboxa\box\@cclv
429 <latexrelease>         \xdef\@deferlist{\@toplist\@botlist\@deferlist}%

430 <latexrelease>         \global \let \@toplist \@empty
431 <latexrelease>         \global \let \@botlist \@empty
432 <latexrelease>         \global \@colroom \@colht
433 <latexrelease>         \ifx \@currlist\@empty
434 <latexrelease>         \else
435 <latexrelease>         \latexerr{Float(s) lost}\@ehb

436 <latexrelease>         \global \let \@currlist \@empty
437 <latexrelease>         \fi
438 <latexrelease>         \@makefcolumn\@deferlist
439 <latexrelease>         \@whilesw\if@fcolmade \fi
440 <latexrelease>         {\@opcol\@makefcolumn\@deferlist}%
441 <latexrelease>         \if@twocolumn
442 <latexrelease>         \if@firstcolumn
443 <latexrelease>         \xdef\@dbldeferlist{\@dbltoplist\@dbldeferlist}%

444 <latexrelease>         \global \let \@dbltoplist \@empty
445 <latexrelease>         \global \@colht \textheight
446 <latexrelease>         \begingroup
447 <latexrelease>         \dblfloatplacement
448 <latexrelease>         \@makefcolumn\@dbldeferlist
449 <latexrelease>         \@whilesw\if@fcolmade \fi
450 <latexrelease>         {\@outputpage\@makefcolumn\@dbldeferlist}%
451 <latexrelease>         \endgroup
452 <latexrelease>         \else
453 <latexrelease>         \vbox{}\clearpage
454 <latexrelease>         \fi
455 <latexrelease>         \fi
456 <latexrelease>         \else
457 <latexrelease>         \setbox\@cclv\vbox{\box\@cclv\vfil}%
458 <latexrelease>         \@makecol\@opcol
459 <latexrelease>         \clearpage
460 <latexrelease>         \fi
461 <latexrelease>         }%
462 <latexrelease> \EndIncludeInRelease

```

(End of definition for \@doclearpage.)

`\@opcol` Several changes in detail here.

```

463 <*2ekernel | fltrace>
464 \def \@opcol {%
465   \if@twocolumn

```

The funny-looking internal commands are interfacing with the new marks mechanism. We make sure (elsewhere) that those are always defined, even when we roll back, so here we add them unconditionally. This still need turning into a hook or config point eventually:

```

466   \@expl@@@mark@update@dblcol@structures@@

```

```

467   \@outputdblcol
468   \else
469     \expl@@@mark@update@singlecol@structures@@
470     \@outputpage
471   <*trace>
472     \fl@trace{PAGE: one column (float? see above) page completed}%
473 </trace>

```

Not needed since it comes after \@outputpage:

```

474 %   \global\@colht\textheight
475   \fi

```

These do not need to be done every time \@opcol is used: they should be grouped together since they all need to be done at the end of the non-special output routine, or at the end of a clearpage one.

```

476   \global \@mparbottom \z@ \global \@textfloatsheight \z@
477   \@floatplacement
478 }
479 </2kernel | fltrace>

```

(End of definition for \@opcol.)

```

480 <*2kernel | latexrelease>
481 <latexrelease>\IncludeInRelease{2025/06/01}%
482 <latexrelease>   {\@makecol}{\@makecol adding hooks/sockets}%

```

\@makecol \@makecol is shortened a lot, basically all the hardwired code in the middle has moved into a socket.

```

483 \def \@makecol {%

```

A number of packages want to prepend code to \@makecol; this is the hook for that:

```

484   \UseHook {build/column/before}%

```

Save away box 255 as \@outputbox to make it available for further adjustments.

```

485   \setbox\@outputbox \box\@cclv

```

The only real addition is the next command which either does nothing or removes an infinite glue from the bottom of the \@outputbox.

```

486   \@outputbox@removebskip

```

Now a kernel hook for tagging that adjusts the content of \@outputbox, if necessary. At this point it just contains the material from the galley.

```

487   \UseTaggingSocket{build/column/outputbox}%

```

When this code is run any “here” floats in the \@outputbox are already handled, so we recycle their registers and put them back to the \@freelist.

```

488   \let\@elt\relax
489   \xdef\@freelist{\@freelist\@midlist}%
490   \global \let \@midlist \@empty

```

Here we have the configurable part. This socket is supposed to add floats, footnotes and stretchable vertical space as appropriate to the \@outputbox. It is used by packages such as footmisc to implement different layout, e.g., footnotes above or below bottom floats, etc.

```

491   \UseSocket {build/column/outputbox}%

```

Then we deal with any `\enlargethispage` or run the normal code to build a column.

```

492 \ifvbox\@kludgeins
493 \@make@specialcolbox
494 \else
495 \@make@normalcolbox
496 \fi
497 \global \maxdepth \@maxdepth

```

Finally, another hook for external packages or classes that want to augment or alter the output routine by appending to `\@makecol`.

```

498 \UseHook {build/column/after}%
499 }

```

(End of definition for \@makecol.)

`\@outputbox@depth` We need to know the depth of `\@outputbox` once in a while. Rather than using a temp dimen (as it was done in the past), we give it a proper register.

```

500 \newdimen\@outputbox@depth

```

(End of definition for \@outputbox@depth.)

`\@make@normalcolbox` Taken out of `\@makecol` for readability.

```

501 \def \@make@normalcolbox {%
502 \setbox\@outputbox \vbox to\@colht {%
503 \@texttop
504 \@outputbox@depth \dp\@outputbox
505 \unvbox \@outputbox

```

The `\vskip -\@outputbox@depth` ensures that the visible depth of the box does not affect the placement of anything on the page. Thus very deep pages will overprint the footer; but these should have been prevented by suitable settings of the `maxdepths` at appropriate times.

If `\@textbottom` ends with a box or rule of non-zero depth then this skip adjustment should be done again after it.

```

506 \vskip -\@outputbox@depth
507 \@textbottom
508 }%
509 }

```

(End of definition for \@make@normalcolbox.)

`\@make@specialcolbox` Make the colbox when `\enlargethispage` was used.

```

510 </2ekernel | latexrelease>
511 < *2ekernel | latexrelease | fltrace>
512 \def \@make@specialcolbox {%
513 < *trace>
514 \fl@trace{Kludgeins ht \the\ht\@kludgeins\space
515 dp \the\dp\@kludgeins\space
516 wd \the\wd\@kludgeins}%
517 </trace>
518 \@outputbox@append {\vskip-\@outputbox@depth}%
519 \@tempdima \@colht
520 \ifdim \wd\@kludgeins>\z@

```

Note that in this case (the *-version), the height of the `\@kludgeins` box is not used since its value is somewhat arbitrary: it need only be big enough to ensure that the page-break is not taken prematurely.

Here we calculate how much vertical space needs to be added in order to enable the column to fit into a box of size `\@colht` using the best information we have about the amount of shrink available (another thing which is known internally about a box, but cannot be accessed at the T_EX level!).

This needs T_EX3 otherwise `\pageshrink` is zero anyway; it may not be exactly the figure we wish as it is the total available from the all the material collected before the page-break decision is made. It will, we think, always be an overestimate of the actual shrink in the box; therefore this should always force the shortest possible column with the possibility of an overfull box.

This should work for both flush- and ragged-bottom setting since it makes the contents no smaller than the size (`\@colht`) of the box into which they are put.

Their should perhaps be an upper limit, of 0pt?, on the extra space added to force shrinking.

```

521     \advance \@tempdima -\ht\@outputbox
522     \advance \@tempdima \pageshrink
523 (*trace)
524     \fl@trace {Natural ht of col: \the \ht\@outputbox}%
525     \fl@trace {\string \@colht: \the \@colht}%
526     \fl@trace {Pageshrink added: \the \pageshrink}%
527     \fl@trace {Hence, space added: \the \@tempdima}%
528 (/trace)
529     \setbox\@outputbox \vbox to \@colht {%
530         \unvbox\@outputbox
531         \vskip \@tempdima
532         \@textbottom
533     }%
```

For the unstarred version, the final size of the page is precisely specified. Therefore, at least for the flush-bottom case, we need to ensure that, visually, it has this size exactly.

Thus we calculate this size and set the material in a box of this size, which is then put into a box of size `\@colht` with `\vss` at the bottom.

```

534     \else
535     \advance \@tempdima -\ht\@kludgeins
536 (*trace)
537     \fl@trace {Natural ht of col: \the \ht\@outputbox}%
538     \fl@trace {\string \@colht: \the \@colht}%
539     \fl@trace {Extra size added: -\the \ht \@kludgeins}%
540     \fl@trace {Hence, height of inner box: \the \@tempdima}%
541     \fl@trace {Max? pageshrink available: \the \pageshrink}%
542 (/trace)
```

This type of final packaging could be done always; this may simplify all of this page-makeup.

It is not necessary to set `\boxmaxdepth` here since the `\@outputbox` ends with glue.

```

543     \setbox \@outputbox \vbox to \@colht {%
544         \vbox to \@tempdima {%
545             \unvbox\@outputbox
546             \@textbottom}%
```

```

547     \vss}%
548   \fi
549   {\setbox \@tempboxa \box \@kludgeins}%
550  (*trace)
551   \fl@trace {kludgeins box made void}%
552  (/trace)
553  }
554  (/2ekernel | latexrelease | fltrace)
555  (*2ekernel | latexrelease)

```

(End of definition for \@make@specialcolbox.)

\@outputbox@removebskip

This is really a bug fix for the kernel (from the 2.09 days) but one we only make by default in new documents that are using \DocumentMetadata. If \raggedbottom is in force, footnotes get attached to the main galley at a distance of \footskip on all pages except on those that are ended by \newpage or \clearpage where the \vfil from \newpage pushes the footnotes to the very bottom.

This is kind of a weird difference to a page ending with \pagebreak—in that case the page is also run short, but the footnotes are not pushed to the bottom.

In footmisc \@outputbox@removebskip is only applied when footmisc is called with an option specifying the footnote placement, i.e., not in the default case. In new documents we apply it always.

```
556 \def\@outputbox@removebskip{%
```

We first test if we are in a \raggedbottom layout. If not we do nothing, but we don't disable the code because \raggedbottom may get used only for some parts of the document.

```
557   \ifx\@textbottom\relax \else
```

In some special circumstances the \@outputbox might be void. As \@outputbox@append below would change this, we better handle this case and leave it unchanged in this situation, because otherwise we can't detect that situation any longer (needed, for example, in ftnright).

```
558     \ifvoid \@outputbox
559       \global\let\@outputbox@reinsertbskip\relax
560     \else
```

If not, we drop the final skip at the end of \@outputbox provided there is one and it has a glue stretch order of 1 or more (i.e., contains a fil or fill part).

```
561       \@outputbox@append{%
562         \@tempskipa\lastskip
563         \ifnum \gluestretchorder\@tempskipa>\z@
564           \unskip
```

We also record the value so that it can be reinserted elsewhere. As we have to do this globally, we also need to explicitly reset it if we don't find any such glue.

```

565         \xdef\@outputbox@reinsertbskip
566           {\noexpand\@outputbox@append{\vskip\the\@tempskipa}}%
567       \else
568         \global\let\@outputbox@reinsertbskip\relax
569       \fi
570     }%
571   \fi
572 \fi
573 }

```

We need a trivial top-level definition for `\@outputbox@reinsertbskip` in case the first page has no bottom glue and the command gets called.

```
574 \let \@outputbox@reinsertbskip \relax
```

(End of definition for `\@outputbox@removebskip` and `\@outputbox@reinsertbskip`.)

2.2 The output routine configuration components

Here we provide the commands that are used to define code for the socket `build/column/outputbox`. The all manipulate the `\@outputbox` box in one way or another.

`\@outputbox@append` This general purpose command alters the `\@outputbox` box by appending material to it. As this is a box typesetting operation we make sure that the last line of the box reflects the true depth of the last line (in case that is needed later). We also expose the current depth of `\@outputbox` as `\@outputbox@depth` before unboxing so that its value can be used by `#1` if wanted.

```
575 \def \@outputbox@append #1{%
576   \setbox \@outputbox \vbox {%
```

This `\boxmaxdepth` setting is to ensure that deep footnotes do not overwrite the footer (on account of the negative skip added later): it should use `\@maxdepth` otherwise the change is pointless when there are footnotes.

But see also its use when combining floats.

```
577   \boxmaxdepth \@maxdepth
578   \@outputbox@depth\dp \@outputbox      % if needed in #1
579   \unvbox \@outputbox
580   #1%
581 }%
582 }
```

(End of definition for `\@outputbox@append`.)

`\@outputbox@appendfootnotes` This command appends the footnotes to the `\@outputbox` (if there are any). If not, then it does nothing.

```
583 \def \@outputbox@appendfootnotes {%
584   \ifvoid \footins \else
```

Handling split footnotes need further work so this is for the moment just a dummy that does nothing. It might vanish and get replaced by a socket eventually.

```
585   \@makecol@handlesplitfootnotes
```

The following socket can be used to manipulate the material in `\footins` box, for example, if the footnotes are to be presented all together in a single paragraph. By default it does nothing.

```
586   \UseSocket{build/column/footnotes}%
```

Then the footnotes are appended:

```
587   \@outputbox@append{%
588     \vskip \skip \footins
```

This is socket adjusts the tagging of the `\footins` box, if necessary.

```
589     \UseTaggingSocket{build/column/footins}%
590     \color@begingroup
591     \normalcolor
592     \footnoterule
```


Support for pdfcolfoot, eventually this can go once color is properly supported. The csname is constructed in case the command is not defined, i.e., the package not loaded.

```

593         \csname pdfcolfoot@current\endcsname
594         \unvbox \footins
595         \color@endgroup
596     }%
597 \fi
598 }

```

(End of definition for \@outputbox@appendfootnotes.)

\makecol@handlesplitfootnotes For future extensions ...

```

599 \let \makecol@handlesplitfootnotes \@empty

```

(End of definition for \@makecol@handlesplitfootnotes.)

\@outputbox@attachfloats Attaching top and bottom floats can usually be done in one go, but for special layouts
\@outputbox@attachtopfloats we might want more control so we provide also separate commands.

\@outputbox@attachbottomfloats The next command was called \@combinefloats in the past.

```

600 \def \@outputbox@attachfloats {%
601     \ifx \@toplist\@empty \else \@cflt \fi
602     \ifx \@botlist\@empty \else \@cflb \fi
603 }
604 \def \@outputbox@attachtopfloats {%
605     \ifx \@toplist\@empty \else \@cflt \fi
606 }
607 \def \@outputbox@attachbottomfloats {%
608     \ifx \@botlist\@empty \else \@cflb \fi
609 }

```

(End of definition for \@outputbox@attachfloats, \@outputbox@attachtopfloats, and \@outputbox@attachbottomfloats.)

The next three conditionals might be useful when setting up running headers and footers so perhaps they should be change to CamelCase names. For now they are internal.

\@if@flushbottom@TF Test for \flushbottom (currently not used).

```

610 \def \@if@flushbottom@TF{%
611     \ifx \@textbottom\relax
612         \expandafter \@firstoftwo
613     \else
614         \expandafter \@secondoftwo
615     \fi
616 }

```

(End of definition for \@if@flushbottom@TF.)

\@if@footnotes@TF Test if footnotes are present on the current page.

```

617 \def \@if@footnotes@TF{%
618     \ifvoid \footins
619         \expandafter \@secondoftwo
620     \else
621         \expandafter \@firstoftwo
622     \fi
623 }

```

(End of definition for \@if@footnotes@TF.)

```
\@if@bottomfloats@TF Test if bottom floats are around.  
624 \def\@if@bottomfloats@TF{%  
625   \ifx \@botlist\@empty  
626     \expandafter\@secondoftwo  
627   \else  
628     \expandafter\@firstoftwo  
629   \fi  
630 }
```

(End of definition for \@if@bottomfloats@TF.)

2.2.1 Configuration sockets

build/column/outputbox (*socket*) We have one socket that is supposed to augment the \@outputbox by attaching floats and footnotes with appropriate spacing.

```
631 \NewSocket{build/column/outputbox}{0}
```

The following plugs are available for this socket:

space-footnotes-floats (*plug*) After the galley text there is a vertical \vfill followed by any footnotes followed by the bottom floats, if any.

```
632 \NewSocketPlug {build/column/outputbox}{space-footnotes-floats} {%  
633   \@if@footnotes@TF  
634     {\@outputbox@append{\vfill}}%  
635     {\@if@bottomfloats@TF  
636       {\@outputbox@append{\vfill}}%  
637       {\@outputbox@reinsertbskip}}%  
638   }%  
639   \@outputbox@appendfootnotes  
640   \@outputbox@attachfloats  
641 }
```

footnotes-space-floats (*plug*) As before but the \vfill is between footnotes and floats.

```
642 \NewSocketPlug {build/column/outputbox}{footnotes-space-floats} {%  
643   \@outputbox@appendfootnotes  
644   \@if@bottomfloats@TF  
645     {\@outputbox@append{\vfill}}%  
646     {\@outputbox@reinsertbskip}}%  
647   \@outputbox@attachfloats  
648 }
```

floats-space-footnotes (*plug*) Floats immediately after the galley text and footnotes at the bottom.

```
649 \NewSocketPlug {build/column/outputbox}{floats-space-footnotes} {%  
650   \@outputbox@attachfloats  
651   \@if@footnotes@TF  
652     {\@outputbox@append{\vfill}}%  
653     {\@outputbox@reinsertbskip}}%  
654   \@outputbox@appendfootnotes  
655 }
```

`space-floats-footnotes` (*plug*) Both floats and footnotes are pushed to the bottom with footnotes last.⁶³

```
656 \NewSocketPlug {build/column/outputbox}{space-floats-footnotes} {%
657   \if@footnotes@TF
658     {\@outputbox@append{\vfill}}%
659     {\@if@bottomfloats@TF
660       {\@outputbox@append{\vfill}}%
661       {\@outputbox@reinsertbskip}}%
662   \@outputbox@attachfloats
663   \@outputbox@appendfootnotes
664 }
```

`floats-footnotes` (*plug*) All excess space has to be distributed across the existing glue on the page, e.g., within the text galley, the separation between blocks, etc. The order is text, floats, footnotes.

```
665 \NewSocketPlug {build/column/outputbox}{floats-footnotes} {%
666   \@outputbox@attachfloats
667   \@outputbox@appendfootnotes
```

We do reinsert the bottom skip from `\newpage` if it was taken out earlier. This is, strictly speaking, not necessary in most cases, but it is a `\vfil` while `\raggedbottom` is only generating `\vspace{0pt plus .0001fil}`, so if you have several `\vfil` on the page before the `\newpage` you would alter the space distribution if one is taken out.

```
668   \@outputbox@reinsertbskip
669 }
```

`footnotes-floats` (*plug*)

```
670 \NewSocketPlug {build/column/outputbox}{footnotes-floats} {%
671   \@outputbox@appendfootnotes
672   \@outputbox@attachfloats
673   \@outputbox@reinsertbskip
674 }
```

The `footnote-floats` plug implements the layout used by L^AT_EX but with the bottom skip bug corrected. This will be the default when `\DocumentMetadata` is used; it can be overwritten either through `footmisc` or by assigning any of the other plugs (or by coding yet another plug for the socket).

`footnotes-floats-legacy` (*plug*) This implements the 2.09 layout (including its bottom skip bug).

```
675 \NewSocketPlug {build/column/outputbox}{footnotes-floats-legacy} {%
```

In the legacy case we don't really want to take out the bottom skip, but rather than altering `\@outputbox@removebskip` in `\makecol` to do nothing, which we would then have to undo in each other layout, we simply reinsert the dropped skip immediately again.

```
676   \@outputbox@reinsertbskip
677   \@outputbox@appendfootnotes
678   \@outputbox@attachfloats
679 }
```

The `footnote-floats-legacy` plug is the default used by L^AT_EX when `\DocumentMetadata` is not used; it can be overwritten either through `footmisc` or by assigning any of the other plugs (or by coding yet another plug for the socket).

```
680 \AssignSocketPlug {build/column/outputbox}{footnotes-floats-legacy}
```

⁶³There are two more permutations, but neither of them has ever been requested so they aren't set up by default — doing that in a class would be trivial though.

`build/column/footnotes` (*socket*) The socket allowing the manipulation of `\footins` box (result needs to be moved back in there). Used when footnotes are reformatted into a single paragraph by the `para` option of `footmisc`. By default it does nothing.

```
681 \NewSocket{build/column/footnotes}{0}
```

`build/page/before` (*hook*) Hooks at the start and end of `\@outputpage` for use by packages.

```
build/page/after
```

```
682 \NewMirroredHookPair{build/page/before}{build/page/after}
```

`build/page/reset` (*hook*) Hook in `\@outputpage` to reset special galley conventions within the output routine.

```
683 \NewHook {build/page/reset}
```

`build/column/before` (*hook*) Hooks at the start and end of `\@makecol` for use by packages.

```
build/column/after
```

```
684 \NewMirroredHookPair{build/column/before}{build/column/after}
```

```
685 </2kernel | latexrelease>
686 <latexrelease>\EndIncludeInRelease
687 <latexrelease>\IncludeInRelease{0000/00/00}%
688 <latexrelease>  {\@makecol}{\@makecol adding hooks/sockets}%
689 <latexrelease>
690 <latexrelease>\gdef \@makecol {%
691 <latexrelease>  \ifvoid\footins
692 <latexrelease>    \setbox\@outputbox \box\@cclv
693 <latexrelease>  \else
694 <latexrelease>    \setbox\@outputbox \vbox {%
695 <latexrelease>      \boxmaxdepth \@maxdepth
696 <latexrelease>      \unvbox \@cclv
697 <latexrelease>      \vskip \skip\footins
698 <latexrelease>      \color@begingroup
699 <latexrelease>      \normalcolor
700 <latexrelease>      \footnoterule
701 <latexrelease>      \unvbox \footins
702 <latexrelease>      \color@endgroup
703 <latexrelease>    }%
704 <latexrelease>  \fi
705 <latexrelease>  \let\@elt\relax
706 <latexrelease>  \xdef\@freelist{\@freelist\@midlist}%
707 <latexrelease>  \global \let \@midlist \@empty
708 <latexrelease>  \@combinefloats
709 <latexrelease>  \ifvbox\@kludgeins
710 <latexrelease>    \@makespecialcolbox
711 <latexrelease>  \else
712 <latexrelease>    \setbox\@outputbox \vbox to\@colht {%
713 <latexrelease>      \texttop
714 <latexrelease>      \dimen@ \dp\@outputbox
715 <latexrelease>      \unvbox \@outputbox
716 <latexrelease>      \vskip -\dimen@
717 <latexrelease>      \textbottom
718 <latexrelease>    }%
719 <latexrelease>  \fi
720 <latexrelease>  \global \maxdepth \@maxdepth
721 <latexrelease>}
722 <latexrelease>
723 <latexrelease>\gdef \@makespecialcolbox {%
724 <latexrelease>  \setbox\@outputbox \vbox {%
```

```

725 <latexrelease> \@texttop
726 <latexrelease> \dimen@ \dp\@outputbox
727 <latexrelease> \unvbox\@outputbox
728 <latexrelease> \vskip-\dimen@
729 <latexrelease> }%
730 <latexrelease> \@tempdima \@colht
731 <latexrelease> \ifdim \wd\@kludgeins>\z@
732 <latexrelease> \advance \@tempdima -\ht\@outputbox
733 <latexrelease> \advance \@tempdima \pageshrink
734 <latexrelease> \setbox\@outputbox \vbox to \@colht {%
735 <latexrelease> \unvbox\@outputbox
736 <latexrelease> \vskip \@tempdima
737 <latexrelease> \@textbottom
738 <latexrelease> }%
739 <latexrelease> \else
740 <latexrelease> \advance \@tempdima -\ht\@kludgeins
741 <latexrelease> \setbox \@outputbox \vbox to \@colht {%
742 <latexrelease> \vbox to \@tempdima {%
743 <latexrelease> \unvbox\@outputbox
744 <latexrelease> \@textbottom}%
745 <latexrelease> \vss}%
746 <latexrelease> \fi
747 <latexrelease> {\setbox \@tempboxa \box \@kludgeins}%
748 <latexrelease>}
749 <latexrelease>
750 <latexrelease> \let \@make@normalcolbox \@undefined
751 <latexrelease> \let \@make@specialcolbox \@undefined
752 <latexrelease> \let \@outputbox@removebskip \@undefined
753 <latexrelease> \let \@outputbox@reinsertbskip \@undefined
754 <latexrelease>
755 <latexrelease> \let \@outputbox@append \@undefined
756 <latexrelease> \let \@outputbox@appendfootnotes \@undefined
757 <latexrelease> \let \@outputbox@attachfloats \@undefined
758 <latexrelease> \let \@outputbox@attachtopfloats \@undefined
759 <latexrelease> \let \@outputbox@attachbottomfloats \@undefined
760 <latexrelease>
761 <latexrelease> \let \@if@flushbottom@TF \@undefined
762 <latexrelease> \let \@if@footnotes@TF \@undefined
763 <latexrelease> \let \@if@bottomfloats@TF \@undefined
764 <latexrelease>
765 <latexrelease>\EndIncludeInRelease
766 <*2kernel>

```

`\@reinserts` This is the code which reinserts the inserts. It puts them all in one place; this can make some of them come out on the wrong page. It has been put into a separate macro to expedite experimentation.

```

767 \gdef \@reinserts{%
768 \ifvoid\footins\else\insert\footins{\unvbox\footins}\fi
769 \ifvbox\@kludgeins\insert\@kludgeins
770 \{\unvbox\@kludgeins}\fi
771 }

```

(End of definition for \@reinserts.)

`\@texttop` These do nothing as a default.
`\@textbottom` 772 `\let \@texttop \relax`
773 `\let \@textbottom \relax`

(End of definition for \@texttop and \@textbottom.)

`\@resetactivechars` RmS 93/09/06: added hook to protect against certain active characters in the output
`\@activechar@info` routine. Default checks are for active space and end-of-line.

```
774 \def\@activechar@info #1{%
775     \@latex@info@no@line {Active #1 character found while
776                           output routine is active
777                           \MessageBreak
778                           This may be a bug in a package file
779                           you are using}%
780 }
```

Do not put any spaces in this next bit!

```
781 \begingroup
782 \obeylines\obeyspaces%
783 \catcode'\@active%
784 \gdef\@resetactivechars{%
785 \def^~M{\@activechar@info{EOL}\space}%
786 \def {\@activechar@info{space}\space}%
787 \let'\active@math@prime}%
788 \endgroup
```

(End of definition for \@resetactivechars and \@activechar@info.)

`\@outputpage` The `\color@hbox` hooks here are used to avoid putting just a colour special into an
`\@shipoutsetup` otherwise empty box (in a header or footer). These boxes are often set to be completely
`\@writesetup` empty and so adding a special produces a very underfull box message.

There has been extensive tidying up of the old code here; including the removal of a level of grouping.

The setting of `\protect` immediately before the `\shipout` is needed so that protected commands within `\writes` are handled correctly.

Within `shipout`'s `vbox` it is reset to its default value, `\relax`.

Resetting it to its default value after the `shipout` has been completed (and the contents of the `writes` have been expanded) must be done by use of `\aftergroup`. This is because it must have the value `\relax` before macros coming from other uses of `\aftergroup` within this box are expanded.

Putting this into the `\aftergroup` token list does not affect the definition used in expanding the `\writes` because the `aftergroup` token list is only constructed when popping the `save-stack`, it is not expanded until after the `shipout` is completed.

Question: should things from an `\aftergroup` within the shipped out box be executed in the environment set up for the `writes`, or after it finishes?

A lot of this code has been in-lined to prevent mis-use of internal commands as hooks.

```
789 </2kernel>
790 <latexrelease>\IncludeInRelease{2025/06/01}%
791 <latexrelease> {\@outputpage}{Use new mark mechanism}%
792 <*2kernel | latexrelease>
```

Temp definition to vanish again when something is offered by the backend:

```
793 \protected \def \pdfannot@link@on@@ { \csname pdfannot_link_on:\endcsname }
794 \protected \def \pdfannot@link@off@@ { \csname pdfannot_link_off:\endcsname }
795 \def \@outputpage {%
```

We start with a hook available to packages that want to prepend code to \@outputpage.

```
796 \UseHook {build/page/before}%
```

The \endgroup is put in by \aftergroup.

```
797 \begingroup
```

Now all the set-up stuff has been in-lined for Frank.

First the stuff for the writes.

From here on was orginially in the command \@writsetup.

The following defenition of \protect is the one active during the \write statement that migrate out of the \shipout box, so even though it is immediately changed below (inside \shipout it still has to be here!

```
798 \let \protect \noexpand
```

RmS 93/08/19: Redefined accents to allow changes in font encoding; but exactly why was this needed?

Reset \language to the value current at \begin{document}. In particular this ensures that a pagebreak in verbatim does not prevent hyphenation in the page head.

```
799 \language\document@default@language
```

The \catcode`\ = 10 was removed as it was considered useless (presumably because nothing gets tokenized during shipout).

This was put in as some error produced active spaces in a mark, I think.

Why was the hyphen reset?

```
800 \@resetactivechars
```

If a page break happens between the start of a list and its first item the @newlist will be true and this will mess up any list that is used in the header or footer of the page. So we have to reset that flag.

```
801 \global\let\@if@newlist\if@newlist
```

```
802 \global\@newlistfalse
```

This next hook replaces the following:

```
\let\-\@dischyph
\let\'@\acci\let\'@\accii\let\=\@acciii
\let\\\@normalcr
\let\par\@par %% 15 Sep 87 (this was once inside the box)
```

and it does more than they did; in particular it sets:

```
\parindent\z@
\parskip\z@skip
\everypar{}%
\leftskip\z@skip
\rightskip\z@skip
\parfillskip\@flushglue
\lineskip\normallineskip
\baselineskip\normalbaselineskip
\sloppy
```

```
803 \@parboxrestore
```

... to here was in the command `\@writsetup`.

Hook to allow adding resets local to the output routine processing, e.g., in `\write` or running headers/footers, for packages that set up special conventions in the main galley that should not leak into the page production just because a page break happens in the middle of such an environment.

```
804 \UseHook {build/page/reset}%
```

```
805 \shipout \vbox{%
```

Inside the `\shipout` box we have a different setting for `\protect`.

```
806 \set@typeset@protect
```

```
807 \aftergroup \endgroup
```

Once the `\shipout` box ends `\protect` is again `\noexpand` which is correct for any `\write` statements, but once they are processed we want to get back to the following setting:

```
808 \aftergroup \set@typeset@protect
```

Now the setup inside the shipped out box; this should contain all the stuff that could only affect typesetting; other stuff may need to be reset for the writes also.

From here ... was in the command `\@shipoutsetup`.

```
809 \if@specialpage
```

```
810 \global \@specialpagefalse
```

```
811 \@nameuse {ps@\@specialstyle}%
```

```
812 \fi
```

```
813 \if@twoside
```

```
814 \ifodd\count\z@
```

```
815 \let \@thehead \@oddhead
```

```
816 \let \@thefoot \@oddfoot
```

```
817 \let \@themargin \oddsidemargin
```

```
818 \else
```

```
819 \let \@thehead \@evenhead
```

```
820 \let \@thefoot \@evenfoot
```

```
821 \let \@themargin \evensidemargin
```

```
822 \fi
```

```
823 \fi
```

The rest was always inside the box.

RmS 91/08/15: added this line:

```
824 \reset@font
```

RmS 93/08/06 Added `\lineskiplimit=0pt` to guard against it being nonzero: e.g. by `\offinterlineskip` being in effect.

There are probably lots of other things that may need resetting.

```
825 \normalsize
```

Reset the space factors.

```
826 \normalsfcodes
```

Reset these here (previously reset separately for head and foot)

```
827 \let \label \@gobble@with@sphack@om
```

```
828 \let \index \@gobble@with@sphack@som
```

```
829 \let \glossary \@gobble@with@sphack@om
```



```

830 \baselineskip \z@skip
831 \lineskip \z@skip
832 \lineskiplimit \z@
... to here was in the command \@shipoutsetup.
833 \@beginndvi
834 \vskip \topmargin
835 \moveright\@themargin \vbox {%
836 \setbox\@tempboxa \vbox to\headheight {%
837 \vfil

```

Tagging socket that receives the header in its second argument to surround the header with appropriate tagging structures (first argument is unused). If tagging is disabled it returns the content of the second argument.

```

838 \pdfannot@link@off@@
839 \UseTaggingSocket{build/page/header}{}%
840 {
841 \color@hbox
842 \normalcolor
843 \hb@xt@ \textwidth {\@thehead }%
844 \color@endbox
845 }
846 \pdfannot@link@on@@
847 }
848 \dp \@tempboxa \z@
849 \box \@tempboxa
850 \vskip \headsep
851 \box \@outputbox
852 \baselineskip \footskip

```

Tagging socket that receives the footer in its second argument to surround the footer with appropriate tagging structures (first argument is unused). If tagging is disabled it returns the content of the second argument.

```

853 \pdfannot@link@off@@
854 \UseTaggingSocket{build/page/footer}{}%
855 {
856 \color@hbox
857 \normalcolor
858 \hb@xt@ \textwidth {\@thefoot }%
859 \color@endbox
860 }
861 \pdfannot@link@on@@
862 }
863 }

```

\endgroup now inserted by \aftergroup

```

Restore \if@newlist
864 \global \let \if@newlist \@@if@newlist
865 \global \@colht \textheight
866 \stepcounter{page}%

```

Another hook for packages that want to append material to \@outputpage.

```

867 \UseHook {build/page/after}%
868 }
869 (latexrelease)\EndIncludeInRelease

```

```

870 <latexrelease>\IncludeInRelease{2017/04/15}%
871 <latexrelease> {\@outputpage}{Reset language for hyphenation}%
872 <latexrelease>\def\@outputpage{%
873 <latexrelease>\begingroup
874 <latexrelease> \let \protect \noexpand
875 <latexrelease> \language\document@default@language
876 <latexrelease> \@resetactivechars
877 <latexrelease> \global\let\@if@newlist@if@newlist
878 <latexrelease> \global\@newlistfalse
879 <latexrelease> \@parboxrestore
880 <latexrelease> \shipout \vbox{%
881 <latexrelease> \set@typeset@protect
882 <latexrelease> \aftergroup \endgroup
883 <latexrelease> \aftergroup \set@typeset@protect
884 <latexrelease> \if@specialpage
885 <latexrelease> \global\@specialpagefalse\@nameuse{ps@\@specialstyle}%
886 <latexrelease> \fi
887 <latexrelease> \if@twoside
888 <latexrelease> \ifodd\count\z@ \let\@thehead\@oddhead \let\@thefoot\@oddfoot
889 <latexrelease> \let\@themargin\oddsidemargin
890 <latexrelease> \else \let\@thehead\@evenhead
891 <latexrelease> \let\@thefoot\@evenfoot \let\@themargin\evensidemargin
892 <latexrelease> \fi
893 <latexrelease> \fi
894 <latexrelease> \reset@font
895 <latexrelease> \normalsize
896 <latexrelease> \normalsfcodes
897 <latexrelease> \let\label\@gobble
898 <latexrelease> \let\index\@gobble
899 <latexrelease> \let\glossary\@gobble
900 <latexrelease> \baselineskip\z@skip \lineskip\z@skip \lineskiplimit\z@
901 <latexrelease> \@begindvi
902 <latexrelease> \vskip \topmargin
903 <latexrelease> \moveright\@themargin \vbox {%
904 <latexrelease> \setbox\@tempboxa \vbox to\headheight{%
905 <latexrelease> \vfil
906 <latexrelease> \color@hbox
907 <latexrelease> \normalcolor
908 <latexrelease> \hb@xt@\textwidth{\@thehead}%
909 <latexrelease> \color@endbox
910 <latexrelease> }%
911 <latexrelease> \dp\@tempboxa \z@
912 <latexrelease> \box\@tempboxa
913 <latexrelease> \vskip \headsep
914 <latexrelease> \box\@outputbox
915 <latexrelease> \baselineskip \footskip
916 <latexrelease> \color@hbox
917 <latexrelease> \normalcolor
918 <latexrelease> \hb@xt@\textwidth{\@thefoot}%
919 <latexrelease> \color@endbox
920 <latexrelease> }%
921 <latexrelease> }%
922 <latexrelease> \global\let@if@newlist\@if@newlist
923 <latexrelease> \global \@colht \textheight

```

```
924 <latexrelease> \stepcounter{page}%
```

It is now clear that this does something useful (in the old mark mechanism), thanks to Pieter van Oostrum pointing this out. It is needed because a float page is made without using TeX's page-builder; thus the output routine is never called so the marks are not updated.

```
925 <latexrelease> \let\firstmark\botmark
926 <latexrelease>}
927 </2ekernel | latexrelease>
928 <latexrelease>\EndIncludeInRelease

929 <latexrelease>\IncludeInRelease{0000/00/00}%
930 <latexrelease> {\@outputpage}{Reset language for hyphenation}%
931 <latexrelease>\def\@outputpage{%
932 <latexrelease>\begingroup
933 <latexrelease> \let \protect \noexpand
934 <latexrelease> \resetactivechars
935 <latexrelease> \global\let\@if@newlist@if@newlist
936 <latexrelease> \global\@newlistfalse
937 <latexrelease> \@parboxrestore
938 <latexrelease> \shipout \vbox{%
939 <latexrelease> \set@typeset@protect
940 <latexrelease> \aftergroup \endgroup
941 <latexrelease> \aftergroup \set@typeset@protect
942 <latexrelease> \if@specialpage
943 <latexrelease> \global\@specialpagefalse\@nameuse{ps@\@specialstyle}%
944 <latexrelease> \fi
945 <latexrelease> \if@twoside
946 <latexrelease> \ifodd\count\z@
947 <latexrelease> \let\@thehead\@oddhead \let\@thefoot\@oddfoot
948 <latexrelease> \let\@themargin\oddsidemargin
949 <latexrelease> \else \let\@thehead\@evenhead
950 <latexrelease> \let\@thefoot\@evenfoot \let\@themargin\evensidemargin
951 <latexrelease> \fi
952 <latexrelease> \fi
953 <latexrelease> \reset@font
954 <latexrelease> \normalsize
955 <latexrelease> \normalsfcodes
956 <latexrelease> \let\label\@gobble
957 <latexrelease> \let\index\@gobble
958 <latexrelease> \let\glossary\@gobble
959 <latexrelease> \baselineskip\z@skip \lineskip\z@skip \lineskiplimit\z@
960 <latexrelease> \@begindvi
961 <latexrelease> \vskip \topmargin
962 <latexrelease> \moveright\@themargin \vbox {%
963 <latexrelease> \setbox\@tempboxa \vbox to\headheight{%
964 <latexrelease> \vfil
965 <latexrelease> \color@hbox
966 <latexrelease> \normalcolor
967 <latexrelease> \hb@xt@\textwidth{\@thehead}%
968 <latexrelease> \color@endbox
969 <latexrelease> }%
970 <latexrelease> \dp\@tempboxa \z@
971 <latexrelease> \box\@tempboxa
972 <latexrelease> \vskip \headsep
```

```

973 <latexrelease>      \box\@outputbox
974 <latexrelease>      \baselineskip \footskip
975 <latexrelease>      \color@hbox
976 <latexrelease>      \normalcolor
977 <latexrelease>      \hb@xt@\textwidth{\@thefoot}%
978 <latexrelease>      \color@endbox
979 <latexrelease>      }%
980 <latexrelease>      }%
981 <latexrelease>      \global\let\if@newlist\@if@newlist
982 <latexrelease>      \global \@colht \textheight
983 <latexrelease>      \stepcounter{page}%
984 <latexrelease>      \let\firstmark\botmark
985 <latexrelease>}
986 <latexrelease>\EndIncludeInRelease
987 (*2ekernel)

```

(End of definition for \@outputpage, \@shipoutsetup, and \@writesetup.)

\@begindivi This unboxes stuff that must appear before anything else in the .dvi file, then returns that box register to the free list and cancels itself.

The stuff in the box should not add any typeset material to the page.

```

988 \def \@begindivi{%
989   \unvbox \@begindivibox
990   \global\let \@begindivi \@empty
991 }

```

(End of definition for \@begindivi.)

2.2.2 Dealing with floats

\@combinefloats Old name for what is now called \@outputbox@attachfloats; kept to support legacy packages.

```

992 \let \@combinefloats \@outputbox@attachfloats

```

(End of definition for \@combinefloats.)

\@cflt The \@boxmaxdepth setting here was not made local to a box so was dangerous. It is needed only within the box made by \@cflt (and not normally even there), so it has been moved there; this also agrees with the original pseudocode.

\@cflb

```

993 \def \@cflt{%
994   \let \@elt \@comflelt
995   \setbox\@tempboxa \vbox{}%
996   \@toplist
997   \setbox\@outputbox \vbox{%
998     \boxmaxdepth \maxdepth
999     \unvbox\@tempboxa
1000     \vskip -\floatsep
1001     \topfigrule
1002     \vskip \textfloatsep
1003     \unvbox\@outputbox
1004   }%
1005   \let\@elt\relax
1006   \xdef\@freelist{\@freelist\@toplist}%
1007   \global\let\@toplist\@empty
1008 }

```

```

1009 \def \@cflb {%
1010   \let\@elt\@comflelt
1011   \setbox\@tempboxa \vbox{}%
1012   \@botlist
1013   \setbox\@outputbox \vbox{%
1014     \unvbox\@outputbox
1015     \vskip \textfloatsep
1016     \botfigrule
1017     \unvbox\@tempboxa
1018     \vskip -\floatsep
1019   }%
1020   \let\@elt\relax
1021   \xdef\@freelist{\@freelist\@botlist}%
1022   \global \let \@botlist\@empty
1023 }

```

(End of definition for \@cflt and \@cflb.)

```

\@comflelt
\@comdblfilelt
\@combinedblfloats
1024 \def\@comflelt#1{\setbox\@tempboxa
1025   \vbox{\unvbox\@tempboxa\box #1\vskip\floatsep}}
1026 \def\@comdblfilelt#1{\setbox\@tempboxa
1027   \vbox{\unvbox\@tempboxa\box #1\vskip\dblfloatsep}}
1028 \def \@combinedblfloats{%
1029   \ifx \@dbltoplist \@empty
1030   \else
1031     \setbox\@tempboxa \vbox{}%
1032     \let \@elt \@comdblfilelt
1033     \@dbltoplist
1034     \let \@elt \relax
1035     \xdef \@freelist {\@freelist\@dbltoplist}%
1036     \global\let \@dbltoplist \@empty
1037     \setbox\@outputbox \vbox to\textheight

```

The setting of \boxmaxdepth here has no effect since the \@outputbox should already have depth zero. Even so, it would have no effect on the layout of the page.

```

1038   {\boxmaxdepth\maxdepth %% probably not needed, CAR
1039   \unvbox\@tempboxa\vskip-\dblfloatsep

```

Here we need different typesetting if the top float comes from \@topnewpage.

```

1040   \ifnum \@dbltopnum>\m@ne
1041     \dblfigrule
1042     \fi
1043     \vskip \dbltextfloatsep

```

If pdf links are present in the galley and those links get broken across pages they have to end up being on the same level of boxing (even if not actually in the same structure) due to some engine restrictions in pdfTEX and LuaTEX. We therefore unbox \@outputbox here (which only contains a single \hbox) so that this case has the same boxing level as a normal twocolumn page without top floats.

```

1044     \unvbox\@outputbox
1045   }%
1046   \fi
1047 }
1048 (/2ekernel)

```

(End of definition for \@comflelt, \@comdblfllelt, and \@combinedblfloats.)

\@startcolumn We could combine (most of) these two into \@startcol <list>. Note that \@xstartcol
\@startdblcolumn was only used once (i.e. in \@startcolumn); it has therefore been removed. This is not
quite as efficient but it now has the same structure as \@startdblcolumn.

The empty-list test has been moved to \@tryfcolumn.

```
1049 <*2ekernel | fltrace>
1050 \def \@startcolumn {%
1051   \global \@colroom \@colht
1052   \@tryfcolumn \@deferlist
1053   \if@fcolmade
1054 <*trace>
1055   \fl@trace{PAGE: float \if@twocolumn column \else page \fi
1056             completed}%
1057 </trace>
1058   \else
1059   \begingroup
1060     \let \reserved@b \@deferlist
1061     \global \let \@deferlist \@empty
1062     \let \@elt \@scolelt
1063     \reserved@b
1064   \endgroup
1065   \fi
1066 }
```

This one does not need to set \@colht.

```
1067 </2ekernel | fltrace>
1068 <latexrelease | fltrace>\IncludeInRelease{2015/01/01}%
1069 <latexrelease | fltrace> {\@startdblcolumn}{float order in 2-column}%
1070 <*2ekernel | latexrelease | fltrace>
1071 \def \@startdblcolumn {%
1072   \@tryfcolumn \@deferlist
1073   \if@fcolmade
1074 <fltrace> \fl@trace{PAGE: double float page completed}%
1075   \else
1076   \begingroup
1077     \let \reserved@b \@deferlist
1078     \global \let \@deferlist \@empty
1079     \let \@elt \@sdblcolelt
1080     \reserved@b
1081   \endgroup
1082   \fi
1083 }%
1084 </2ekernel | latexrelease | fltrace>
1085 <latexrelease | fltrace>\EndIncludeInRelease
1086 <latexrelease | fltrace>\IncludeInRelease{0000/00/00}%
1087 <latexrelease | fltrace> {\@startdblcolumn}{float order in 2-column}%
1088 <latexrelease | fltrace>\def \@startdblcolumn {%
1089 <latexrelease | fltrace>% \global \@colht \textheight
1090 <latexrelease | fltrace> \@tryfcolumn \@dbldeferlist
1091 <latexrelease | fltrace> \if@fcolmade
```

```

1092 <*trace>
1093 <latexrelease | fltrace> \fl@trace{PAGE: double float page completed}%
1094 </trace>
1095 <latexrelease | fltrace> \else

1096 <latexrelease | fltrace> \begingroup
1097 <latexrelease | fltrace> \let \reserved@b \@dbldeferlist
1098 <latexrelease | fltrace> \global \let \@dbldeferlist \@empty
1099 <latexrelease | fltrace> \let \@elt \@sdblcolelt
1100 <latexrelease | fltrace> \reserved@b
1101 <latexrelease | fltrace> \endgroup
1102 <latexrelease | fltrace> \fi
1103 <latexrelease | fltrace>}%
1104 <latexrelease | fltrace>\EndIncludeInRelease
1105 <*2ekernel | fltrace>

```

(End of definition for \@startcolumn and \@startdblcolumn.)

`\@tryfcolumn` Now tests if its list is empty before any further exertion.

```

1106 \def \@tryfcolumn #1{%
1107   \global \@fcolmadefalse
1108   \ifx #1\@empty
1109     \else
1110 <*trace>
1111   \fl@trace{PAGE: try float \if@twocolumn column/page\else page\fi
1112             ---\string #1}%
1113   \fl@trace{----- \string #1: #1}%
1114 </trace>

1115   \xdef \@trylist{#1}%
1116   \global \let \@failedlist \@empty
1117   \begingroup
1118     \let \@elt \@xtryfc \@trylist
1119   \endgroup
1120   \if@fcolmade
1121     \@vtryfc #1%
1122   \fi
1123 \fi
1124 }
1125 </2ekernel | fltrace>

```

(End of definition for \@tryfcolumn.)

1126 <*2ekernel>

`\@scolelt`

```

1127 \def \@scolelt#1{\def \@currbox{#1}\@addtonextcol}

```

(End of definition for \@scolelt.)

`\@sdblcolelt`

```

1128 \def \@sdblcolelt#1{\def \@currbox{#1}\@addtodblcol}

```

(End of definition for \@sdblcolelt.)

`\@vtryfc`

```
1129 \def\@vtryfc #1{%
1130   \global\setbox\@outputbox\vbox{}%
1131   \let\@elt\@vtryfc
1132   \@flsucceed
1133   \global\setbox\@outputbox \vbox to\@colht{%
1134     \vskip \@fptop
1135     \vskip -\@fpsep
1136     \unvbox \@outputbox
1137     \vskip \@fpbot}%
1138   \let\@elt\relax
1139   \xdef #1{\@failedlist\@flfail}%
1140   \xdef\@freelist{\@freelist\@flsucceed}}
```

(End of definition for \@vtryfc.)

`\@wtryfc`

```
1141 \def\@wtryfc #1{%
1142   \global\setbox\@outputbox\vbox{%
1143     \unvbox\@outputbox
1144     \vskip\@fpsep
1145     \box #1}}
```

(End of definition for \@wtryfc.)

`\@xtryfc`

```
1146 </2ekernel>
1147 <latexrelease>\IncludeInRelease{2015/01/01}{\@xtryfc}%
1148 <latexrelease>                                {float order in 2-column}%
1149 <*2ekernel | latexrelease>
1150 \def\@xtryfc #1{%
1151   \@next\reserved@a\@trylist{}{}%
1152   \@currtype \count #1%
1153   \divide\@currtype\@xxxii
1154   \multiply\@currtype\@xxxii
1155   \@bitor \@currtype \@failedlist
1156   \@testfp #1%
1157   \@testwrongwidth #1%
1158   \ifdim \ht #1>\@colht
1159     \@testtrue
1160   \fi
1161   \if@test
1162     \@cons\@failedlist #1%
1163   \else
1164     \@ytryfc #1%
1165   \fi}%
1166 </2ekernel | latexrelease>
1167 <latexrelease>\EndIncludeInRelease
1168 <latexrelease>\IncludeInRelease{0000/00/00}{\@xtryfc}%
1169 <latexrelease>                                {float order in 2-column}%
1170 <latexrelease>\def\@xtryfc #1{%
1171 <latexrelease>   \@next\reserved@a\@trylist{}{}%
1172 <latexrelease>   \@currtype \count #1%
```



```

1173 <latexrelease> \divide\@currtype\@xxxii
1174 <latexrelease> \multiply\@currtype\@xxxii
1175 <latexrelease> \@bitor \@currtype \@failedlist
1176 <latexrelease> \@testfp #1%
1177 <latexrelease> \ifdim \ht #1>\@colht
1178 <latexrelease> \@testtrue
1179 <latexrelease> \fi
1180 <latexrelease> \if@test
1181 <latexrelease> \@cons\@failedlist #1%
1182 <latexrelease> \else
1183 <latexrelease> \@ytryfc #1%
1184 <latexrelease> \fi}%
1185 <latexrelease>\EndIncludeInRelease
1186 <*2ekernel)

```

(End of definition for \@xtryfc.)

\@ytryfc

```

1187 \def\@ytryfc #1{%
1188 \begingroup
1189 \gdef\@flsucceed{\@elt #1}%
1190 \global\let\@flfail\@empty
1191 \@tempdima\ht #1%
1192 \let\@elt\@ztryfc
1193 \@trylist
1194 \ifdim \@tempdima >\@fpmi
1195 \global\@fcolmadetrue
1196 \else
1197 \@cons\@failedlist #1%
1198 \fi
1199 \endgroup
1200 \if@fcolmade
1201 \let\@elt\@gobble
1202 \fi}

```

(End of definition for \@ytryfc.)

\@ztryfc

```

1203 </2ekernel)
1204 <latexrelease>\IncludeInRelease{2015/01/01}{\@ztryfc}%
1205 <latexrelease> \float order in 2-column}%
1206 <*2ekernel | latexrelease)
1207 \def\@ztryfc #1{%
1208 \@tempcnta\count #1%
1209 \divide\@tempcnta\@xxxii
1210 \multiply\@tempcnta\@xxxii
1211 \@bitor \@tempcnta {\@failedlist \@flfail}%
1212 \@testfp #1%
1213 not in fixfloats?
1214 \@testwrongwidth #1%
1215 \@tempdimb\@tempdima
1216 \advance\@tempdimb\ht #1%
1217 \advance\@tempdimb\@fpsep

```

```

1217 \ifdim \@tempdimb >\@colht
1218 \testtrue
1219 \fi
1220 \if@test
1221 \cons\@flfail #1%
1222 \else
1223 \cons\@flsucceed #1%
1224 \tempdima\@tempdimb
1225 \fi}%
1226 </2kernel | latexrelease>
1227 <latexrelease>\EndIncludeInRelease
1228 <latexrelease>\IncludeInRelease{0000/00/00}{\ztryfc}%
1229 <latexrelease> \float order in 2-column}%
1230 <latexrelease>\def\@ztryfc #1{%
1231 <latexrelease> \tempcnta \count#1%
1232 <latexrelease> \divide\@tempcnta\@xxxii
1233 <latexrelease> \multiply\@tempcnta\@xxxii
1234 <latexrelease> \bitor \@tempcnta {\@failedlist \@flfail}%
1235 <latexrelease> \testfp #1%
1236 <latexrelease> \tempdimb\@tempdima
1237 <latexrelease> \advance\@tempdimb \ht#1%
1238 <latexrelease> \advance\@tempdimb\@fpsep
1239 <latexrelease> \ifdim \@tempdimb >\@colht
1240 <latexrelease> \testtrue
1241 <latexrelease> \fi
1242 <latexrelease> \if@test
1243 <latexrelease> \cons\@flfail #1%
1244 <latexrelease> \else
1245 <latexrelease> \cons\@flsucceed #1%
1246 <latexrelease> \tempdima\@tempdimb
1247 <latexrelease> \fi}%
1248 <latexrelease>\EndIncludeInRelease

```

(End of definition for \@ztryfc.)

The major changes for float suppression and the changes to the float mechanism to make it conform to the documentation are in these next macros.

\@addtobot Lots of changes.

```

1249 <*2kernel | fltrace>
1250 \def \@addtobot {%
1251 <*trace>
1252 \fl@trace{***Start addtobot}%
1253 </trace>
1254 \getfpsbit 4\relax
1255 <*trace>
1256 \fl@trace{fpstype \ifodd \@tempcnta OK \else not \fi bot:
1257 \the \@fpstype}%
1258 </trace>
1259 \ifodd \@tempcnta
1260 \flsetnum \@botnum
1261 \ifnum \@botnum>\z@
1262 \tempswafalse
1263 \flcheckspace \@botroom \@botlist
1264 \if@tempswa

```

This next line means that this page is produced with box 255 having depth zero, rather than the normal maxdepth: is this needed, useful?

```

1265     \global \maxdepth \z@
1266     \flupdates \@botnum \@botroom \@botlist
1267 (*trace)
1268     \fl@trace{colroom (after-bot) = \the \@colroom}%
1269     \fl@trace{colnum (after-bot) = \the \@colnum}%
1270     \fl@trace{botnum (after-bot) = \the \@botnum}%
1271     \fl@trace{***Success: bot}%
1272 (/trace)
1273     \inserttrue
1274     \fi
1275 (*trace)
1276     \else
1277     \fl@trace{Fail: botnum = \the \@botnum:
1278             fpstype \the \@fpstype=ORD?}%
1279     \ifnum \@fpstype<\sixt@n
1280     \fl@trace{ERROR: !b float not successful (addtobot)}%
1281     \fi
1282 (/trace)
1283     \fi
1284     \fi
1285 }

```

(End of definition for \@addtobot.)

\@addtotoporbot Lots of changes.

```

1286 \def \@addtotoporbot {%
1287 (*trace)
1288     \fl@trace{***Start addtotoporbot}%
1289 (/trace)
1290     \@getfpsbit \tw@
1291 (*trace)
1292     \fl@trace{fpstype \ifodd \@tempcnta OK \else not \fi top:
1293             \the \@fpstype}%
1294 (/trace)
1295     \ifodd \@tempcnta
1296     \@flsetnum \@topnum
1297     \ifnum \@topnum>\z@
1298     \@tempswafalse
1299     \@flcheckspace \@toproom \@toplist
1300     \if@tempswa
1301     \@bitor\@currtype{\@midlist\@botlist}%
1302 (*trace)
1303     \fl@trace{(mid+bot)list: \@midlist, \@botlist:
1304             (addtotoporbot-before)}%
1305 (/trace)
1306     \if@test
1307 (*trace)
1308     \fl@trace{type already on list: mid or bot---sent to addtobot}%
1309 (/trace)
1310     \else
1311     \flupdates \@topnum \@toproom \@toplist
1312 (*trace)

```

```

1313     \fl@trace{colroom (after-top) = \the \@colroom}%
1314     \fl@trace{colnum (after-top) = \the \@colnum}%
1315     \fl@trace{topnum (after-top) = \the \@topnum}%
1316     \fl@trace{***Success: top}%
1317 
```

```

1317 
```

```

1318     \inserttrue
1319     \fi
1320     \fi
1321 
```

```

1321 
```

```

1322     \else
1323     \fl@trace{Fail: topnum = \the \@topnum: fpstype
1324                                     \the \@fpstype=ORD?}%
1325     \ifnum \@fpstype<\sift@n
1326     \fl@trace{ERROR: !t float not successful (addtotoporbot)}%
1327     \fi
1328 
```

```

1328 
```

```

1329     \fi
1330     \fi
1331     \if@insert
1332     \else
1333 
```

```

1333 
```

```

1334     \fl@trace{sent to addtobot (addtotoporbot)}%
1335 
```

```

1335 
```

```

1336     \addtobot
1337     \fi
1338 }
1339 
```

(End of definition for \@addtotoporbot.)

\@addtocurcol Lots of changes.

```

1340 <latexrelease | fltrace | flafter>\IncludeInRelease{2025/06/01}%
1341 <latexrelease | fltrace | flafter> {\@addtocurcol}{float order in 2-column}%
1342 <*2ekernel | latexrelease | fltrace | flafter>
1343 \def \@addtocurcol {%
1344 <*trace>
1345     \fl@trace{***Start addtocurcol}%
1346 </trace>
1347     \insertfalse
1348     \@setfloatypecounts
1349     \ifnum \@fpstype=8
1350 <*trace>
1351     \fl@trace{fpstype !p only (addtocurcol): \the \@fpstype = 8?}%
1352 </trace>
1353     \else
1354     \ifnum \@fpstype=24
1355 <*trace>
1356     \fl@trace{fpstype p only (addtocurcol): \the \@fpstype = 24?}%
1357 </trace>
1358     \else
1359     \flsettextmin

```

This is a new adjustment which is quite a major change in functionality; but it implements the documentation. Note that \@reqcolroom will include the whole of the

page-so-far, and hence includes `\@textfloatsheight` of floats, so before comparing it with `\@textmin`, we add this to `\@textmin` also.

```

1360 (*trace)
1361     \fl@trace{textfloatsheight (before) = \the \@textfloatsheight}%
1362 </trace>
1363     \advance \@textmin \@textfloatsheight
1364     \reqcolroom \@pageht

```

This line must be removed since `\@specialoutput` changed.

```

1365 %     \advance \@reqcolroom \@pagedp
1366 (*trace)
1367     \fl@trace{textmin + textfloatsheight: \the \@textmin}%
1368     \fl@trace{page-so-far: \the \@reqcolroom}%
1369 </trace>
1370     \ifdim \@textmin>\@reqcolroom
1371         \reqcolroom \@textmin
1372 (*trace)
1373     \fl@trace{ORD? textmin being used}%
1374 </trace>
1375     \fi
1376     \advance \@reqcolroom \ht\@currbox

```

We save the current value of `\@reqcolroom` so that we can return to this value later, in case a test fails that may have added something to it.

```

1377     \saved@reqcolroom \@reqcolroom
1378 (*trace)
1379     \fl@trace{float size = \the \ht \@currbox (addtocurcol)}%
1380     \fl@trace{colroom = \the \@colroom (addtocurcol)}%
1381     \fl@trace{reqcolroom = \the \@reqcolroom (addtocurcol)}%
1382 </trace>
1383     \ifdim \@colroom>\@reqcolroom
1384         \@flsetnum \@colnum
1385         \ifnum \@colnum>\z@
1386             \@bitor\@currtype\@deferlist

```

We need to defer the float also if its width doesn't fit.

```

1387     \@testwrongwidth\@currbox
1388 (*trace)
1389     \fl@trace{deferlist: \@deferlist: (addtocurcol-before)}%
1390 </trace>
1391     \if@test
1392 (*trace)
1393     \fl@trace{type already on list: defer (addtocurcol)}%
1394 </trace>
1395     \else
1396         \@bitor\@currtype\@botlist
1397 (*trace)
1398     \fl@trace{botlist: \@botlist: (addtocurcol-before)}%
1399 </trace>
1400     \if@test
1401 (*trace)
1402     \fl@trace{type already on list: bot---sent to addtobot}%
1403 </trace>
1404     \@addtobot

```

```

1405         \else
1406     (*trace)
1407         \fl@trace{fpstype \ifodd \@tempcnta OK \else not \fi
1408             here: \the \@fpstype}%
1409     (/trace)
1410         \ifodd \count\@currbox
1411         \advance \@reqcolroom \intextsep
1412         \ifdim \@colroom>\@reqcolroom
1413         \global \advance \@colnum \m@ne
1414         \global \advance \@textfloatsheight \ht\@currbox

```

This may sometimes give an overestimate.

```

1415         \global \advance \@textfloatsheight 2\intextsep
1416         \@cons \@midlist \@currbox
1417     (*trace)
1418         \fl@trace{***Success: here}%
1419         \fl@trace{textfloatsheight (after-here) =
1420             \the \@textfloatsheight}%
1421         \fl@trace{colnum (after-here) = \the \@colnum}%
1422     (/trace)

```

CHANGE TO \@addtocurcol:

\penalty\z@ changed to \penalty\interlinepenalty so \samepage works properly with figure and table environments. (Changed 23 Oct 86)

There is also an \addpenalty\interlinepenalty above.

Although it is best to use \addvspace in case two h floats come together, this makes other spacing more difficult to adjust; whereas if a user specifies two h floats together then they can more easily get the spacing correct by ad hoc commands.

It is necessary to adjust for the addition of \parskip here in case the float is added between paragraphs (i.e. when in vertical mode).

If the nobreak switch is true we need to reset it and clear \everypar since the float may not reset the flag and cannot reset the \everypar globally.

Typesetting starts here (we are in vertical mode).

```

1423         \if@nobreak
1424         \nobreak
1425         \@nobreakfalse
1426         \everypar{}%
1427     \else
1428         \addpenalty \interlinepenalty
1429         \fi
1430         \vskip \intextsep
1431         \box\@currbox
1432         \penalty\interlinepenalty
1433         \vskip\intextsep
1434         \ifnum\outputpenalty <-\@Mii \vskip -\parskip\fi

```

Typesetting ends here.

```

1435         \outputpenalty \z@
1436         \@inserttrue
1437     (*trace)
1438         \else
1439         \fl@trace{Fail---no room at 2nd test of colroom
1440             (addtocorcol \string\intextsep)}%
1441     (/trace)

```

```

1442         \fi
1443     \fi
1444     \if@insert
1445     \else
Next set of docstrip guards are a bit weird, essentially \@addtotoporbot ends up inside
the kernel and the fltrace package and \@addtobot shows up in the flafter package.
Guess that could have been done a bit more obvious :-)
1446 <*2kernel | fltrace | latexrelease>
1447 <*trace>
1448         \fl@trace{not here: sent to addtotoporbot}%
1449 </trace>
1450         \@addtotoporbot
1451 </2kernel | fltrace | latexrelease>
1452 <!*2kernel&!fltrace&!latexrelease>
1453 <*trace>
1454         \fl@trace{not here: sent to addtobot}%
1455 </trace>
1456         \@addtobot
1457 </!2kernel&!fltrace&!latexrelease>
1458         \fi
1459     \fi
1460     \fi
1461 <*trace>
1462     \else
1463         \fl@trace{Fail: colnum = \the \@colnum:
1464             fpstype \the \@fpstype=ORD?}%
1465         \ifnum \@fpstype<\sist@n
1466             \fl@trace{ERROR: BANG float not successful (addtocurcol)}%
1467         \fi
1468 </trace>
1469     \fi
1470 <*trace>
1471     \else
1472         \fl@trace{Fail---no room: fl box ht: \the \ht \@currbox
1473             (addtocurcol)}%
1474 </trace>
1475     \fi
1476     \fi
1477     \fi
1478     \if@insert
1479     \else
1480         \@resethfps
1481 <*trace>
1482         \fl@trace{put on deferlist (addtocurcol)}%
1483 </trace>
1484         \@cons\@deferlist\@currbox
1485 <*trace>
1486         \fl@trace{deferlist: \@deferlist: (addtocurcol-after)}%
1487 </trace>
1488     \fi
1489 }%
1490 </2kernel | latexrelease | fltrace | flafter>
1491 <latexrelease | fltrace | flafter>\EndIncludeInRelease

```

The rollback code is rather complicated as it has to account for the fact that we roll back not just the kernel but also the packages flafter and fltrace.

```

1492 <latexrelease | fltrace | flafter> \IncludeInRelease{2015/01/01}%
1493 <latexrelease | fltrace | flafter> {\@addtocurcol}{float order in 2-column}%
1494 <latexrelease | fltrace | flafter> \def \@addtocurcol {%
1495 *trace)
1496 <latexrelease | fltrace | flafter> \fl@trace{***Start addtocurcol}%
1497 /trace)
1498 <latexrelease | fltrace | flafter> \@insertfalse
1499 <latexrelease | fltrace | flafter> \@setfloattyperecounts
1500 <latexrelease | fltrace | flafter> \ifnum \@fpstype=8
1501 *trace)
1502 <latexrelease | fltrace | flafter> \fl@trace{fpstype !p only (addtocurcol): \the \@fpstype = 8?}%
1503 /trace)
1504 <latexrelease | fltrace | flafter> \else
1505 <latexrelease | fltrace | flafter> \ifnum \@fpstype=24
1506 *trace)
1507 <latexrelease | fltrace | flafter> \fl@trace{fpstype p only (addtocurcol): \the \@fpstype = 24?}%
1508 /trace)
1509 <latexrelease | fltrace | flafter> \else
1510 <latexrelease | fltrace | flafter> \@flsettextmin
1511 *trace)
1512 <latexrelease | fltrace | flafter> \fl@trace{textfloatsheight (before) = \the \@textfloatsheight}%
1513 /trace)
1514 <latexrelease | fltrace | flafter> \advance \@textmin \@textfloatsheight
1515 <latexrelease | fltrace | flafter> \@reqcolroom \@pageht
1516 *trace)
1517 <latexrelease | fltrace | flafter> \fl@trace{textmin + textfloatsheight: \the \@textmin}%
1518 <latexrelease | fltrace | flafter> \fl@trace{page-so-far: \the \@reqcolroom}%
1519 /trace)
1520 <latexrelease | fltrace | flafter> \ifdim \@textmin>\@reqcolroom
1521 <latexrelease | fltrace | flafter> \@reqcolroom \@textmin
1522 *trace)
1523 <latexrelease | fltrace | flafter> \fl@trace{ORD? textmin being used}%
1524 /trace)
1525 <latexrelease | fltrace | flafter> \fi
1526 <latexrelease | fltrace | flafter> \advance \@reqcolroom \ht\@currbox
1527 *trace)
1528 <latexrelease | fltrace | flafter> \fl@trace{float size = \the \ht \@currbox (addtocurcol)}%
1529 <latexrelease | fltrace | flafter> \fl@trace{colroom = \the \@colroom (addtocurcol)}%
1530 <latexrelease | fltrace | flafter> \fl@trace{reqcolroom = \the \@reqcolroom (addtocurcol)}%
1531 /trace)
1532 <latexrelease | fltrace | flafter> \ifdim \@colroom>\@reqcolroom
1533 <latexrelease | fltrace | flafter> \@flsetnum \@colnum
1534 <latexrelease | fltrace | flafter> \ifnum \@colnum>\z@
1535 <latexrelease | fltrace | flafter> \@bitor\@currtype\@deferlist
1536 <latexrelease | fltrace | flafter> \@testwrongwidth\@currbox
1537 *trace)
1538 <latexrelease | fltrace | flafter> \fl@trace{deferlist: \@deferlist: (addtocurcol-before)}%
1539 /trace)
1540 <latexrelease | fltrace | flafter> \if@test
1541 *trace)
1542 <latexrelease | fltrace | flafter> \fl@trace{type already on list: defer (addtocurcol)}%
1543 /trace)

```



```

1544 <latexrelease | fltrace | flafter> \else
1545 <latexrelease | fltrace | flafter> \@bitor\@currtype\@botlist
1546 <*trace>
1547 <latexrelease | fltrace | flafter> \fl@trace{botlist: \@botlist: (addtocurcol-before)}%
1548 </trace>
1549 <latexrelease | fltrace | flafter> \if@test
1550 <*trace>
1551 <latexrelease | fltrace | flafter> \fl@trace{type already on list: bot---sent to addtobot}
1552 </trace>
1553 <latexrelease | fltrace | flafter> \@addtobot
1554 <latexrelease | fltrace | flafter> \else
1555 <*trace>
1556 <latexrelease | fltrace | flafter> \fl@trace{fpstype \ifodd \@tempcnta OK \else not \fi
1557 <latexrelease | fltrace | flafter> here: \the \@fpstype}%
1558 </trace>
1559 <latexrelease | fltrace | flafter> \ifodd \count\@currbox
1560 <latexrelease | fltrace | flafter> \advance \@reqcolroom \intextsep
1561 <latexrelease | fltrace | flafter> \ifdim \@colroom>\@reqcolroom
1562 <latexrelease | fltrace | flafter> \global \advance \@colnum \m@ne
1563 <latexrelease | fltrace | flafter> \global \advance \@textfloatsheight \ht\@currbox
1564 <latexrelease | fltrace | flafter> \global \advance \@textfloatsheight 2\intextsep
1565 <latexrelease | fltrace | flafter> \@cons \@midlist \@currbox
1566 <*trace>
1567 <latexrelease | fltrace | flafter> \fl@trace{***Success: here}%
1568 <latexrelease | fltrace | flafter> \fl@trace{textfloatsheight (after-here) =
1569 <latexrelease | fltrace | flafter> \the \@textfloatsheight}%
1570 <latexrelease | fltrace | flafter> \fl@trace{colnum (after-here) = \the \@colnum}%
1571 </trace>
1572 <latexrelease | fltrace | flafter> \if@nobreak
1573 <latexrelease | fltrace | flafter> \nobreak
1574 <latexrelease | fltrace | flafter> \@nobreakfalse
1575 <latexrelease | fltrace | flafter> \everypar{}%
1576 <latexrelease | fltrace | flafter> \else
1577 <latexrelease | fltrace | flafter> \addpenalty \interlinepenalty
1578 <latexrelease | fltrace | flafter> \fi
1579 <latexrelease | fltrace | flafter> \vskip \intextsep
1580 <latexrelease | fltrace | flafter> \box\@currbox
1581 <latexrelease | fltrace | flafter> \penalty\interlinepenalty
1582 <latexrelease | fltrace | flafter> \vskip\intextsep
1583 <latexrelease | fltrace | flafter> \ifnum\outputpenalty <-\@Mii \vskip -\parskip\fi
1584 <latexrelease | fltrace | flafter> \outputpenalty \z@
1585 <latexrelease | fltrace | flafter> \@inserttrue
1586 <*trace>
1587 <latexrelease | fltrace | flafter> \else
1588 <latexrelease | fltrace | flafter> \fl@trace{Fail---no room at 2nd test of colroom
1589 <latexrelease | fltrace | flafter> (addtocorcol \string\intextsep)}%
1590 </trace>
1591 <latexrelease | fltrace | flafter> \fi
1592 <latexrelease | fltrace | flafter> \fi
1593 <latexrelease | fltrace | flafter> \if@insert
1594 <latexrelease | fltrace | flafter> \else
1595 <*trace>
1596 <fltrace | latexrelease> \fl@trace{not here: sent to addtotoporbot}%
1597 </trace>

```

```

1598 <fltrace | latexrelease>                \@addtotoporbot
1599 <*trace>
1600 <flafter>                                \fl@trace{not here: sent to addtobot}%
1601 </trace>
1602 <flafter>                                \@addtobot
1603 <latexrelease | fltrace | flafter>        \fi
1604 <latexrelease | fltrace | flafter>        \fi
1605 <latexrelease | fltrace | flafter>        \fi
1606 <*trace>
1607 <latexrelease | fltrace | flafter>        \else
1608 <latexrelease | fltrace | flafter>        \fl@trace{Fail: colnum = \the \@colnum:
1609 <latexrelease | fltrace | flafter>        fpstype \the \@fpstype=ORD?}%
1610 <latexrelease | fltrace | flafter>        \ifnum \@fpstype<\sist@n
1611 <latexrelease | fltrace | flafter>        \fl@trace{ERROR: BANG float not successful (addtocurcol)}
1612 <latexrelease | fltrace | flafter>        \fi
1613 </trace>
1614 <latexrelease | fltrace | flafter>        \fi
1615 <*trace>
1616 <latexrelease | fltrace | flafter>        \else
1617 <latexrelease | fltrace | flafter>        \fl@trace{Fail---no room: fl box ht: \the \ht \@currbox
1618 <latexrelease | fltrace | flafter>        (addtocurcol)}%
1619 </trace>
1620 <latexrelease | fltrace | flafter>        \fi
1621 <latexrelease | fltrace | flafter>        \fi
1622 <latexrelease | fltrace | flafter>        \fi
1623 <latexrelease | fltrace | flafter>        \if@insert
1624 <latexrelease | fltrace | flafter>        \else
1625 <latexrelease | fltrace | flafter>        \@resethfps
1626 <*trace>
1627 <latexrelease | fltrace | flafter>        \fl@trace{put on deferlist (addtocurcol)}%
1628 </trace>
1629 <latexrelease | fltrace | flafter>        \@cons\@deferlist\@currbox
1630 <*trace>
1631 <latexrelease | fltrace | flafter>        \fl@trace{deferlist: \@deferlist: (addtocurcol-after)}%
1632 </trace>
1633 <latexrelease | fltrace | flafter>        \fi
1634 <latexrelease | fltrace | flafter>}%
1635 <latexrelease | fltrace | flafter>\EndIncludeInRelease
1636 <latexrelease | fltrace | flafter>\IncludeInRelease{0000/00/00}%
1637 <latexrelease | fltrace | flafter> {\@addtocurcol}{float order in 2-column}%
1638 <latexrelease | fltrace | flafter>\def \@addtocurcol {%
1639 <*trace>
1640 <latexrelease | fltrace | flafter> \fl@trace{***Start addtocurcol}%
1641 </trace>
1642 <latexrelease | fltrace | flafter> \@insertfalse
1643 <latexrelease | fltrace | flafter> \@setfloattyperecounts
1644 <latexrelease | fltrace | flafter> \ifnum \@fpstype=8
1645 <*trace>
1646 <latexrelease | fltrace | flafter> \fl@trace{fpstype !p only (addtocurcol):
1647 <latexrelease | fltrace | flafter> \the \@fpstype = 8?}%
1648 </trace>
1649 <latexrelease | fltrace | flafter> \else
1650 <latexrelease | fltrace | flafter> \ifnum \@fpstype=24
1651 <*trace>

```

```

1652 <latexrelease | fltrace | flafter> \fl@trace{fpstype p only (addtocurcol):
1653 <latexrelease | fltrace | flafter> \the \@fpstype = 24?}%
1654 </trace>
1655 <latexrelease | fltrace | flafter> \else
1656 <latexrelease | fltrace | flafter> \@flsettextmin

```

This is a new adjustment which is quite a major change in functionality; but it implements the documentation. Note that `\@reqcolroom` will include the whole of the page-so-far, and hence includes `\@textfloatsheight` of floats, so before comparing it with `\@textmin`, we add this to `\@textmin` also.

```

1657 <*trace>
1658 <latexrelease | fltrace | flafter> \fl@trace{textfloatsheight (before) =
1659 <latexrelease | fltrace | flafter> \the \@textfloatsheight}%
1660 </trace>
1661 <latexrelease | fltrace | flafter> \advance \@textmin \@textfloatsheight
1662 <latexrelease | fltrace | flafter> \@reqcolroom \@pageht

```

This line must be removed since `\@specialoutput` changed.

```

1663 % \advance \@reqcolroom \@pagedp
1664 <*trace>
1665 <latexrelease | fltrace | flafter> \fl@trace{textmin + textfloatsheight:
1666 <latexrelease | fltrace | flafter> \the \@textmin}%
1667 <latexrelease | fltrace | flafter> \fl@trace{page-so-far: \the \@reqcolroom}%
1668 <latexrelease | fltrace | flafter>
1669 </trace>
1670 <latexrelease | fltrace | flafter> \ifdim \@textmin>\@reqcolroom
1671 <latexrelease | fltrace | flafter> \@reqcolroom \@textmin
1672 <*trace>
1673 <latexrelease | fltrace | flafter> \fl@trace{ORD? textmin being used}%
1674 </trace>
1675 <latexrelease | fltrace | flafter> \fi
1676 <latexrelease | fltrace | flafter> \advance \@reqcolroom \ht\@currbox
1677 <*trace>
1678 <latexrelease | fltrace | flafter> \fl@trace{float size =
1679 <latexrelease | fltrace | flafter> \the \ht \@currbox (addtocurcol)}%
1680 <latexrelease | fltrace | flafter> \fl@trace{colroom =
1681 <latexrelease | fltrace | flafter> \the \@colroom (addtocurcol)}%
1682 <latexrelease | fltrace | flafter> \fl@trace{reqcolroom =
1683 <latexrelease | fltrace | flafter> \the \@reqcolroom (addtocurcol)}%
1684 </trace>
1685 <latexrelease | fltrace | flafter> \ifdim \@colroom>\@reqcolroom
1686 <latexrelease | fltrace | flafter> \@flsetnum \@colnum
1687 <latexrelease | fltrace | flafter> \ifnum \@colnum>\z@
1688 <latexrelease | fltrace | flafter> \@bitor\@currtype\@deferlist
1689 <*trace>
1690 <latexrelease | fltrace | flafter> \fl@trace{deferlist:
1691 <latexrelease | fltrace | flafter> \@deferlist: (addtocurcol-before)}%
1692 </trace>
1693 <latexrelease | fltrace | flafter> \if@test
1694 <*trace>
1695 <latexrelease | fltrace | flafter> \fl@trace{type already on list:
1696 <latexrelease | fltrace | flafter> defer (addtocurcol)}%
1697 </trace>
1698 <latexrelease | fltrace | flafter> \else

```

```

1699 <latexrelease | fltrace | flafter>          \@bitor\@currtype\@botlist
1700 <*trace>
1701 <latexrelease | fltrace | flafter>          \fl@trace{botlist: \@botlist:
1702 <latexrelease | fltrace | flafter>          (addtocurcol-before)}%
1703 </trace>
1704 <latexrelease | fltrace | flafter>          \if@test
1705 <*trace>
1706 <latexrelease | fltrace | flafter>          \fl@trace{type already on list:
1707 <latexrelease | fltrace | flafter>          bot---sent to addtobot}%
1708 </trace>
1709 <latexrelease | fltrace | flafter>          \@addtobot
1710 <latexrelease | fltrace | flafter>          \else
1711 <*trace>
1712 <latexrelease | fltrace | flafter>          \fl@trace{fpstype
1713 <latexrelease | fltrace | flafter>          \ifodd \@tempcnta OK \else not \fi
1714 <latexrelease | fltrace | flafter>          here: \the \@fpstype}%
1715 </trace>
1716 <latexrelease | fltrace | flafter>          \ifodd \count\@currbox
1717 <latexrelease | fltrace | flafter>          \advance \@reqcolroom \intextsep
1718 <latexrelease | fltrace | flafter>          \ifdim \@colroom>\@reqcolroom
1719 <latexrelease | fltrace | flafter>          \global \advance \@colnum \m@ne
1720 <latexrelease | fltrace | flafter>          \global \advance
1721 <latexrelease | fltrace | flafter>          \@textfloatsheight\ht\@currbox

```

This may sometimes give an overestimate.

```

1722 <latexrelease | fltrace | flafter>          \global \advance
1723 <latexrelease | fltrace | flafter>          \@textfloatsheight 2\intextsep
1724 <latexrelease | fltrace | flafter>          \@cons \@midlist \@currbox
1725 <*trace>
1726 <latexrelease | fltrace | flafter>          \fl@trace{***Success: here}%
1727 <latexrelease | fltrace | flafter>          \fl@trace{textfloatsheight
1728 <latexrelease | fltrace | flafter>          (after-here) =
1729 <latexrelease | fltrace | flafter>          \the \@textfloatsheight}%
1730 <latexrelease | fltrace | flafter>          \fl@trace{colnum (after-here) =
1731 <latexrelease | fltrace | flafter>          \the \@colnum}%
1732 </trace>

```

CHANGE TO \@addtocurcol:

\penalty\z@ changed to \penalty\interlinepenalty so \samepage works properly with figure and table environments. (Changed 23 Oct 86)

There is also an \addpenalty\interlinepenalty above.

Although it is best to use \addvspace in case two h floats come together, this makes other spacing more difficult to adjust; whereas if a user specifies two h floats together then they can more easily get the spacing correct by ad hoc commands.

It is necessary to adjust for the addition of \parskip here in case the float is added between paragraphs (i.e. when in vertical mode).

If the nobreak switch is true we need to reset it and clear \everypar since the float may not reset the flag and cannot reset the \everypar globally.

Typesetting starts here (we are in vertical mode).

```

1733 <latexrelease | fltrace | flafter>          \if@nobreak
1734 <latexrelease | fltrace | flafter>          \nobreak
1735 <latexrelease | fltrace | flafter>          \@nobreakfalse
1736 <latexrelease | fltrace | flafter>          \everypar{}%
1737 <latexrelease | fltrace | flafter>          \else

```

```

1738 <latexrelease | fltrace | flafter>          \addpenalty\interlinepenalty
1739 <latexrelease | fltrace | flafter>          \fi
1740 <latexrelease | fltrace | flafter>          \vskip \intextsep
1741 <latexrelease | fltrace | flafter>          \box\@currbox
1742 <latexrelease | fltrace | flafter>          \penalty\interlinepenalty
1743 <latexrelease | fltrace | flafter>          \vskip\intextsep
1744 <latexrelease | fltrace | flafter>          \ifnum\outputpenalty
1745 <latexrelease | fltrace | flafter>          <-\@Mii \vskip
1746 <latexrelease | fltrace | flafter>          -\parskip\fi

```

Typesetting ends here.

```

1747 <latexrelease | fltrace | flafter>          \outputpenalty \z@
1748 <latexrelease | fltrace | flafter>          \@inserttrue
1749 <*trace>
1750 <latexrelease | fltrace | flafter>          \else
1751 <latexrelease | fltrace | flafter> \fl@trace{Fail---no room at 2nd test of colroom
1752 <latexrelease | fltrace | flafter>          (addtocorcol \string\intextsep)}%
1753 </trace>
1754 <latexrelease | fltrace | flafter>          \fi
1755 <latexrelease | fltrace | flafter>          \fi
1756 <latexrelease | fltrace | flafter>          \if@insert
1757 <latexrelease | fltrace | flafter>          \else

```

Next set of docstrip guards are a bit weird, essentially \@addtotoporbot ends up inside the kernel and the fltrace package and \@addtotoporbot shows up in the flafter package. Guess that could have been done a bit more obvious :-)

```

1758 <*2kernel | fltrace>
1759 <*trace>
1760 <latexrelease | fltrace | flafter>          \fl@trace{not here: sent to addtotoporbot}%
1761 </trace>
1762 <latexrelease | fltrace | flafter>          \@addtotoporbot
1763 </2kernel | fltrace>
1764 <!*2kernel&!autoload&!fltrace>
1765 <*trace>
1766 <latexrelease | fltrace | flafter>          \fl@trace{not here: sent to addtobot}%
1767 </trace>
1768 <latexrelease | fltrace | flafter>          \@addtobot
1769 </!2kernel&!autoload&!fltrace>
1770 <latexrelease | fltrace | flafter>          \fi
1771 <latexrelease | fltrace | flafter>          \fi
1772 <latexrelease | fltrace | flafter>          \fi
1773 <*trace>
1774 <latexrelease | fltrace | flafter>          \else
1775 <latexrelease | fltrace | flafter>          \fl@trace{Fail: colnum = \the \@colnum:
1776 <latexrelease | fltrace | flafter>          fpstype \the \@fpstype=ORD?}%
1777 <latexrelease | fltrace | flafter>          \ifnum \@fpstype<\sist@n
1778 <latexrelease | fltrace | flafter> \fl@trace{ERROR: BANG float not successful
1779 <latexrelease | fltrace | flafter>          (addtocurcol)}%
1780 <latexrelease | fltrace | flafter>          \fi
1781 </trace>
1782 <latexrelease | fltrace | flafter>          \fi
1783 <*trace>
1784 <latexrelease | fltrace | flafter>          \else
1785 <latexrelease | fltrace | flafter>          \fl@trace{Fail---no room: fl box ht:
1786 <latexrelease | fltrace | flafter>          \the \ht \@currbox (addtocurcol)}%

```

```

1787 </trace>
1788 <latexrelease | fltrace | flafter>      \fi
1789 <latexrelease | fltrace | flafter>      \fi
1790 <latexrelease | fltrace | flafter>      \fi
1791 <latexrelease | fltrace | flafter>      \if@insert
1792 <latexrelease | fltrace | flafter>      \else
1793 <latexrelease | fltrace | flafter>      \@resetfps
1794 <*trace>
1795 <latexrelease | fltrace | flafter>      \fl@trace{put on deferlist (addtocurcol)}%
1796 </trace>
1797 <latexrelease | fltrace | flafter>      \@cons\@deferlist\@currbox
1798 <*trace>
1799 <latexrelease | fltrace | flafter>      \fl@trace{deferlist: \@deferlist:
1800 <latexrelease | fltrace | flafter>      (addtocurcol-after)}%
1801 </trace>
1802 <latexrelease | fltrace | flafter>      \fi
1803 <latexrelease | fltrace | flafter>      }%
1804 <latexrelease | fltrace | flafter>\EndIncludeInRelease

```

(End of definition for \@addtocurcol.)

\@addtonextcol Lots of changes.

```

1805 <latexrelease | fltrace>\IncludeInRelease{2025/06/01}
1806 <latexrelease | fltrace> {\@addtonextcol}{float order in 2-column}%
1807 <*2ekernel | latexrelease | fltrace>
1808 \def\@addtonextcol{%
1809   \begingroup
1810 <*trace>
1811   \fl@trace{***Start addtonextcol}%
1812 </trace>
1813   \@insertfalse
1814   \@setfloattyperecounts
1815   \ifnum \@fpstype=8
1816 <*trace>
1817   \fl@trace{fpstype not curcol: \the \@fpstype = 8?}%
1818 </trace>
1819   \else
1820   \ifnum \@fpstype=24
1821 <*trace>
1822   \fl@trace{fpstype not curcol: \the \@fpstype = 24?}%
1823 </trace>
1824   \else
1825   \@flsettextmin
1826 <*trace>
1827   \fl@trace{text-so-far: Opt (top of col)}%
1828 </trace>
1829   \@reqcolroom \ht\@currbox
1830 <*trace>
1831   \fl@trace{float size: \the \@reqcolroom (addtonextcol)}%
1832 </trace>
1833   \advance \@reqcolroom \@textmin
1834   \saved@reqcolroom \@reqcolroom
1835 <*trace>

```

```

1836     \fl@trace{colroom = \the \@colroom (addtonextcol)}%
1837     \fl@trace{reqcolroom = \the \@reqcolroom (addtonextcol)}%
1838 </trace>
1839     \ifdim \@colroom>\@reqcolroom
1840         \@flsetnum \@colnum
1841         \ifnum \@colnum>\z@
1842             \@bitor\@currtype\@deferlist
1843 <*trace>
1844         \fl@trace{deferlist: \@deferlist: (addtonextcol-before)}%
1845 </trace>
1846
1847         \@testwrongwidth\@currbox
1848
1849         \if@test
1850 <*trace>
1851             \fl@trace{type already on list: defer (addtonextcol)}%
1852 </trace>
1853         \else
1854 <*trace>
1855             \fl@trace{sent to addtotoporbot (addtonextcol)}%
1856 </trace>
1857             \@addtotoporbot
1858         \fi
1859 <*trace>
1860         \else
1861             \fl@trace{Fail---no room: fl box ht: \the \ht \@currbox
1862                 (addtonextcol)}%
1863 </trace>
1864         \fi
1865         \fi
1866         \if@insert
1867         \else
1868 <*trace>
1869             \fl@trace{put back on deferlist (addtonextcol)}%
1870 </trace>
1871             \@cons\@deferlist\@currbox
1872 <*trace>
1873             \fl@trace{deferlist: \@deferlist: (addtonextcol-after)}%
1874 </trace>
1875         \fi
1876 <*trace>
1877         \fl@trace{End of addtonextcol -- locally counts:}%
1878         \fl@trace{col: \the \@colnum. top: \the \@topnum. bot: \the \@botnum.}%
1879 </trace>
1880         \endgroup
1881 <*trace>
1882         \fl@trace{End of addtonextcol -- globally counts:}%
1883         \fl@trace{col: \the \@colnum. top: \the \@topnum. bot: \the \@botnum.}%
1884 </trace>
1885     }%
1886 </2ekernel | latexrelease | fltrace>
1887 <latexrelease | fltrace>\EndIncludeInRelease
1888 <latexrelease | fltrace>\IncludeInRelease{2015/01/01}

```

```

1889 <latexrelease | fltrace> {\@addtonextcol}{float order in 2-column}%
1890 <latexrelease | fltrace> \def\@addtonextcol{%
1891 <latexrelease | fltrace> \begingroup
1892 <*trace>
1893 <latexrelease | fltrace> \fl@trace{***Start addtonextcol}%
1894 </trace>
1895 <latexrelease | fltrace> \@insertfalse
1896 <latexrelease | fltrace> \@setfloattyperecounts
1897 <latexrelease | fltrace> \ifnum \@fpstype=8
1898 <*trace>
1899 <latexrelease | fltrace> \fl@trace{fpstype not curcol: \the \@fpstype = 8?}%
1900 </trace>
1901 <latexrelease | fltrace> \else
1902 <latexrelease | fltrace> \ifnum \@fpstype=24
1903 <*trace>
1904 <latexrelease | fltrace> \fl@trace{fpstype not curcol: \the \@fpstype = 24?}%
1905 </trace>
1906 <latexrelease | fltrace> \else
1907 <latexrelease | fltrace> \@flsettextmin
1908 <*trace>
1909 <latexrelease | fltrace> \fl@trace{text-so-far: Opt (top of col)}%
1910 </trace>
1911 <latexrelease | fltrace> \@reqcolroom \ht\@currbox
1912 <*trace>
1913 <latexrelease | fltrace> \fl@trace{float size: \the \@reqcolroom (addtonextcol)}%
1914 </trace>
1915 <latexrelease | fltrace> \advance \@reqcolroom \@textmin
1916 <*trace>
1917 <latexrelease | fltrace> \fl@trace{colroom = \the \@colroom (addtonextcol)}%
1918 <latexrelease | fltrace> \fl@trace{reqcolroom = \the \@reqcolroom (addtonextcol)}%
1919 </trace>
1920 <latexrelease | fltrace> \ifdim \@colroom>\@reqcolroom
1921 <latexrelease | fltrace> \@flsetnum \@colnum
1922 <latexrelease | fltrace> \ifnum \@colnum>\z@
1923 <latexrelease | fltrace> \@bitor\@currtype\@deferlist
1924 <*trace>
1925 <latexrelease | fltrace> \fl@trace{deferlist: \@deferlist: (addtonextcol-before)}%
1926 </trace>
1927 <latexrelease | fltrace> \@testwrongwidth\@currbox
1928 <latexrelease | fltrace> \if@test
1929 <*trace>
1930 <latexrelease | fltrace> \fl@trace{type already on list: defer (addtonextcol)}%
1931 </trace>
1932 <latexrelease | fltrace> \else
1933 <*trace>
1934 <latexrelease | fltrace> \fl@trace{sent to addtotoporbot (addtonextcol)}%
1935 </trace>
1936 <latexrelease | fltrace> \@addtotoporbot
1937 <latexrelease | fltrace> \fi
1938 <latexrelease | fltrace> \fi
1939 <*trace>
1940 <latexrelease | fltrace> \else
1941 <latexrelease | fltrace> \fl@trace{Fail---no room: fl box ht: \the \ht \@currbox
1942 <latexrelease | fltrace> (addtonextcol)}%

```



```

1943 </trace>
1944 <latexrelease | fltrace> \fi
1945 <latexrelease | fltrace> \fi
1946 <latexrelease | fltrace> \fi
1947 <latexrelease | fltrace> \if@insert
1948 <latexrelease | fltrace> \else
1949 <*trace>
1950 <latexrelease | fltrace> \fl@trace{put back on deferlist (addtonextcol)}%
1951 </trace>
1952 <latexrelease | fltrace> \@cons\@deferlist\@currbox
1953 <*trace>
1954 <latexrelease | fltrace> \fl@trace{deferlist: \@deferlist: (addtonextcol-after)}%
1955 </trace>
1956 <latexrelease | fltrace> \fi
1957 <*trace>
1958 <latexrelease | fltrace> \fl@trace{End of addtonextcol -- locally counts:}%
1959 <latexrelease | fltrace> \fl@trace{col: \the\@colnum. top: \the \@topnum. bot: \the \@botnum.}%
1960 </trace>
1961 <latexrelease | fltrace> \endgroup
1962 <*trace>
1963 <latexrelease | fltrace> \fl@trace{End of addtonextcol -- globally counts:}%
1964 <latexrelease | fltrace> \fl@trace{col: \the\@colnum. top: \the \@topnum. bot: \the \@botnum.}%
1965 </trace>
1966 <latexrelease | fltrace>}%
1967 <latexrelease | fltrace>\EndIncludeInRelease

1968 <latexrelease | fltrace>\IncludeInRelease{0000/00/00}%
1969 <latexrelease | fltrace> {\@addtonextcol}{float order in 2-column}%
1970 <latexrelease | fltrace>\def\@addtonextcol{%
1971 <latexrelease | fltrace> \begingroup
1972 <*trace>
1973 <latexrelease | fltrace> \fl@trace{***Start addtonextcol}%
1974 </trace>
1975 <latexrelease | fltrace> \@insertfalse
1976 <latexrelease | fltrace> \@setfloattyperecounts
1977 <latexrelease | fltrace> \ifnum \@fpstype=8
1978 <*trace>
1979 <latexrelease | fltrace> \fl@trace{fpstype not curcol:
1980 <latexrelease | fltrace> \the \@fpstype = 8?}%
1981 </trace>
1982 <latexrelease | fltrace> \else
1983 <latexrelease | fltrace> \ifnum \@fpstype=24
1984 <*trace>
1985 <latexrelease | fltrace> \fl@trace{fpstype not curcol:
1986 <latexrelease | fltrace> \the \@fpstype = 24?}%
1987 </trace>
1988 <latexrelease | fltrace> \else
1989 <latexrelease | fltrace> \@flsettextmin
1990 <*trace>
1991 <latexrelease | fltrace> \fl@trace{text-so-far: Opt (top of col)}%
1992 </trace>
1993 <latexrelease | fltrace> \@reqcolroom \ht\@currbox
1994 <*trace>
1995 <latexrelease | fltrace> \fl@trace{float size:
1996 <latexrelease | fltrace> \the \@reqcolroom (addtonextcol)}%

```

```

1997 <latexrelease | fltrace>
1998 </trace>
1999 <latexrelease | fltrace> \advance \@reqcolroom \@textmin
2000 <*trace>
2001 <latexrelease | fltrace> \fl@trace{colroom =
2002 <latexrelease | fltrace> \the \@colroom (addtonextcol)}%
2003 <latexrelease | fltrace> \fl@trace{reqcolroom =
2004 <latexrelease | fltrace> \the \@reqcolroom (addtonextcol)}%
2005 </trace>
2006 <latexrelease | fltrace> \ifdim \@colroom>\@reqcolroom
2007 <latexrelease | fltrace> \flsetnum \@colnum
2008 <latexrelease | fltrace> \ifnum \@colnum>\z@
2009 <latexrelease | fltrace> \@bitor \@currtype \@deferlist
2010 <*trace>
2011 <latexrelease | fltrace> \fl@trace{deferlist: \@deferlist:
2012 <latexrelease | fltrace> (addtonextcol-before)}%
2013 </trace>
2014 <latexrelease | fltrace> \if@test
2015 <*trace>
2016 <latexrelease | fltrace> \fl@trace{type already on list:
2017 <latexrelease | fltrace> defer (addtonextcol)}%
2018 </trace>
2019 <latexrelease | fltrace> \else
2020 <*trace>
2021 <latexrelease | fltrace> \fl@trace{sent to addtotoporbot
2022 <latexrelease | fltrace> (addtonextcol)}%
2023 </trace>
2024 <latexrelease | fltrace> \@addtotoporbot
2025 <latexrelease | fltrace> \fi
2026 <latexrelease | fltrace> \fi
2027 <*trace>
2028 <latexrelease | fltrace> \else
2029 <latexrelease | fltrace> \fl@trace{Fail---no room: fl box ht:
2030 <latexrelease | fltrace> \the \ht \@currbox (addtonextcol)}%
2031 </trace>
2032 <latexrelease | fltrace> \fi
2033 <latexrelease | fltrace> \fi
2034 <latexrelease | fltrace> \fi
2035 <latexrelease | fltrace> \if@insert
2036 <latexrelease | fltrace> \else
2037 <*trace>
2038 <latexrelease | fltrace> \fl@trace{put back on deferlist
2039 <latexrelease | fltrace> (addtonextcol)}%
2040 </trace>
2041 <latexrelease | fltrace> \@cons \@deferlist \@currbox
2042 <*trace>
2043 <latexrelease | fltrace> \fl@trace{deferlist: \@deferlist:
2044 <latexrelease | fltrace> (addtonextcol-after)}%
2045 </trace>
2046 <latexrelease | fltrace> \fi
2047 <*trace>
2048 <latexrelease | fltrace> \fl@trace{End of addtonextcol --
2049 <latexrelease | fltrace> locally counts:}%
2050 <latexrelease | fltrace> \fl@trace{col: \the \@colnum.

```

```

2051 <latexrelease | fltrace>      top: \the \@topnum. bot: \the \@botnum.}%
2052 </trace>
2053 <latexrelease | fltrace> \endgroup
2054 <*trace>
2055 <latexrelease | fltrace> \fl@trace{End of addtonextcol --
2056 <latexrelease | fltrace>      globally counts:}%
2057 <latexrelease | fltrace> \fl@trace{col: \the \@colnum.
2058 <latexrelease | fltrace>      top: \the \@topnum. bot: \the \@botnum.}%
2059 </trace>
2060 <latexrelease | fltrace>}%
2061 <latexrelease | fltrace>\EndIncludeInRelease

```

(End of definition for \@addtonextcol.)

\@addtodblcol Lots of changes.

```

2062 <latexrelease | fltrace>\IncludeInRelease{2015/01/01}%
2063 <latexrelease | fltrace> {\@addtodblcol}{float order in 2-column}%
2064 <*2ekernel | latexrelease | fltrace>
2065 \def\@addtodblcol{%
2066   \begingroup
2067   <*trace>
2068   \fl@trace{***Start addtodblcol}%
2069 </trace>
2070   \@insertfalse
2071   \@setfloattypescounts
2072   \@getfpsbit \tw@
2073 <*trace>
2074   \fl@trace{fpstype \ifodd \@tempcnta OK \else not \fi dbltop:
2075   \the \@fpstype}%
2076 </trace>
2077   \ifodd\@tempcnta
2078     \@flsetnum \@dbltopnum
2079     \ifnum \@dbltopnum>\z@
2080       \@tempswafalse
2081       \ifdim \@dbltoproom>\ht\@currbox
2082         \@tempwatrue
2083 <*trace>
2084         \fl@trace{Space OK: \@dbltoproom =
2085         \the \@dbltoproom > \the \ht \@currbox
2086         (dbltoproom)}%
2087 </trace>
2088   \else
2089 <*trace>
2090     \fl@trace{fpstype: \the \@fpstype (addtodblcol)}%
2091 </trace>
2092     \ifnum \@fpstype<\sift@@n
2093 <*trace>
2094       \fl@trace{BANG float ignoring \@dbltoproom}%
2095       \fl@trace{\@spaces \@dbltoproom = \the \@dbltoproom.
2096       Ht float: \the \ht \@currbox-BANG}%
2097 </trace>

```

Need to check that there is room on the page, using the local value of \@textmin to make the necessary adjustment to \@dbltoproom.

```

2098         \advance \@dbltoproom \@textmin
2099 (*trace)
2100         \fl@trace{Local value of texmin: \the\@textmin}%
2101         \fl@trace{\@spaces space on page = \the \@dbltoproom.
2102                 Ht float: \the \ht \@currbox-BANG}%
2103 (/trace)
2104         \ifdim \@dbltoproom>\ht\@currbox
2105             \@tempwatrue
2106 (*trace)
2107             \fl@trace{Space OK BANG: space on page =
2108                     \the \@dbltoproom > \the \ht \@currbox}%
2109         \else
2110             \fl@trace{fpstype: \the \@fpstype}%
2111             \fl@trace{Fail---no room dbltoproom-BANG?:}%
2112             \fl@trace{\@spaces space on page = \the \@dbltoproom.
2113                     Ht float: \the \ht \@currbox}%
2114 (/trace)
2115         \fi
2116         \advance \@dbltoproom -\@textmin
2117 (*trace)
2118         \else
2119             \fl@trace{fpstype: \the \@fpstype}%
2120             \fl@trace{Fail---no room dbltoproom-ORD?:}%
2121             \fl@trace{\@spaces \@dbltoproom = \the \@dbltoproom.
2122                     Ht float: \the \ht \@currbox}%
2123 (/trace)
2124         \fi
2125         \fi
2126         \if@tempswa
2127             \@bitor \@currtype \@deferlist
2128 (*trace)
2129             \fl@trace{(dbl)deferlist: \@deferlist: (before)}%
2130 (/trace)
2131         not in fixfloats?
2132         \@testwrongwidth\@currbox
2133 (*trace)
2134         \fl@trace{type already on list: (dbl)defer}%
2135 (/trace)
2136         \else
2137             \@tempdima -\ht\@currbox
2138             \advance\@tempdima
2139             -\ifx \@dbltoplist\@empty \dbltextfloatsep \else
2140                 \dblfloatsep \fi
2141             \global \advance \@dbltoproom \@tempdima
2142             \global \advance \@colht \@tempdima
2143             \global \advance \@dbltopnum \m@ne
2144             \@cons \@dbltoplist \@currbox
2145 (*trace)
2146             \fl@trace{dbltopnum (after) = \the \@dbltopnum}%
2147             \fl@trace{***Success: dbltop}%
2148 (/trace)
2149             \@inserttrue

```

```

2150         \fi
2151     \fi
2152 <*trace>
2153     \else
2154         \fl@trace{Fail: dbltopnum = \the \@dbltopnum: fpstype
2155                                     \the \@fpstype=ORD?}%
2156         \ifnum \@fpstype<\sixt@n
2157             \fl@trace{ERROR: !t float not successful (addtodblcol)}%
2158         \fi
2159 </trace>
2160     \fi
2161     \fi
2162     \if@insert
2163     \else
2164 <*trace>
2165         \fl@trace{put on deferlist}%
2166 </trace>
2167         \@cons\@deferlist\@currbox
2168 <*trace>
2169         \fl@trace{(dbl)deferlist: \@deferlist: (after)}%
2170 </trace>
2171     \fi
2172 <*trace>
2173         \fl@trace{End of addtodblcol -- locally count:}%
2174         \fl@trace{dbltop: \the \@dbltopnum.}%
2175 </trace>
2176     \endgroup
2177 <*trace>
2178         \fl@trace{End of addtodblcol -- globally count:}%
2179         \fl@trace{dbltop: \the \@dbltopnum.}%
2180 </trace>
2181 }%
2182 </2ekernel | latexrelease | fltrace>
2183 <latexrelease | fltrace> \EndIncludeInRelease
2184 <latexrelease | fltrace> \IncludeInRelease{0000/00/00}%
2185 <latexrelease | fltrace> {\@addtodblcol}{float order in 2-column}%
2186 <latexrelease | fltrace> \def\@addtodblcol{%
2187 <latexrelease | fltrace> \beginngroup
2188 <*trace>
2189 <latexrelease | fltrace> \fl@trace{***Start addtodblcol}%
2190 </trace>
2191 <latexrelease | fltrace> \@insertfalse
2192 <latexrelease | fltrace> \@setfloattypecounts
2193 <latexrelease | fltrace> \@getfpsbit \tw@
2194 <*trace>
2195 <latexrelease | fltrace> \fl@trace{fpstype \ifodd \@tempcnta OK
2196 <latexrelease | fltrace> \else not \fi dbltop: \the \@fpstype}%
2197 </trace>
2198 <latexrelease | fltrace> \ifodd\@tempcnta
2199 <latexrelease | fltrace> \@flsetnum \@dbltopnum
2200 <latexrelease | fltrace> \ifnum \@dbltopnum>\z@
2201 <latexrelease | fltrace> \@tempswafalse
2202 <latexrelease | fltrace> \ifdim \@dbltoproom>\ht\@currbox
2203 <latexrelease | fltrace> \@tempswatrue

```

```

2204 (*trace)
2205 <latexrelease | fltrace> \fl@trace{Space OK: \@dbltoproom =
2206 <latexrelease | fltrace> \the \@dbltoproom > \the \ht \@currbox
2207 <latexrelease | fltrace> (dbltoproom)}%
2208 </trace>
2209 <latexrelease | fltrace> \else
2210 (*trace)
2211 <latexrelease | fltrace> \fl@trace{fpstype: \the \@fpstype (addtodblcol)}%
2212 </trace>
2213 <latexrelease | fltrace> \ifnum \@fpstype<\sift@n
2214 (*trace)
2215 <latexrelease | fltrace> \fl@trace{BANG float ignoring \@dbltoproom}%
2216 <latexrelease | fltrace> \fl@trace{\@spaces \@dbltoproom =
2217 <latexrelease | fltrace> \the \@dbltoproom.
2218 <latexrelease | fltrace> Ht float: \the \ht \@currbox-BANG}%
2219 </trace>

```

Need to check that there is room on the page, using the local value of \@textmin to make the necessary adjustment to \@dbltoproom.

```

2220 <latexrelease | fltrace> \advance \@dbltoproom \@textmin
2221 (*trace)
2222 <latexrelease | fltrace> \fl@trace{Local value of texmin: \the \@textmin}%
2223 <latexrelease | fltrace> \fl@trace{\@spaces space on page =
2224 <latexrelease | fltrace> \the \@dbltoproom.
2225 <latexrelease | fltrace> Ht float: \the \ht \@currbox-BANG}%
2226 </trace>
2227 <latexrelease | fltrace> \ifdim \@dbltoproom>\ht \@currbox
2228 <latexrelease | fltrace> \@tempwatrue
2229 (*trace)
2230 <latexrelease | fltrace> \fl@trace{Space OK BANG: space on page =
2231 <latexrelease | fltrace> \the \@dbltoproom > \the \ht \@currbox}%
2232 <latexrelease | fltrace> \else
2233 <latexrelease | fltrace> \fl@trace{fpstype: \the \@fpstype}%
2234 <latexrelease | fltrace> \fl@trace{Fail---no room dbltoproom-BANG?:}%
2235 <latexrelease | fltrace> \fl@trace{\@spaces space on page =
2236 <latexrelease | fltrace> \the \@dbltoproom.
2237 <latexrelease | fltrace> Ht float: \the \ht \@currbox}%
2238 </trace>
2239 <latexrelease | fltrace> \fi
2240 <latexrelease | fltrace> \advance \@dbltoproom -\@textmin
2241 (*trace)
2242 <latexrelease | fltrace> \else
2243 <latexrelease | fltrace> \fl@trace{fpstype: \the \@fpstype}%
2244 <latexrelease | fltrace> \fl@trace{Fail---no room dbltoproom-ORD?:}%
2245 <latexrelease | fltrace> \fl@trace{\@spaces \@dbltoproom =
2246 <latexrelease | fltrace> \the \@dbltoproom.
2247 <latexrelease | fltrace> Ht float: \the \ht \@currbox}%
2248 </trace>
2249 <latexrelease | fltrace> \fi
2250 <latexrelease | fltrace> \fi
2251 <latexrelease | fltrace> \if@tempswa
2252 <latexrelease | fltrace> \@bitor \@currtype \@dbldeferlist
2253 (*trace)
2254 <latexrelease | fltrace> \fl@trace{dbldeferlist:

```

```

2255 <latexrelease | fltrace> \dbldeferlist: (before)}%
2256 </trace>
2257 <latexrelease | fltrace> \if@test
2258 <*trace>
2259 <latexrelease | fltrace> \fl@trace{type already on list: dbldefer}%
2260 </trace>
2261 <latexrelease | fltrace> \else
2262 <latexrelease | fltrace> \@tempdima -\ht\@currbox
2263 <latexrelease | fltrace> \advance\@tempdima
2264 <latexrelease | fltrace> -\ifx \dbltoplist\@empty
2265 <latexrelease | fltrace> \dbltextfloatsep
2266 <latexrelease | fltrace> \else \dblfloatsep \fi
2267 <latexrelease | fltrace> \global \advance \dbltoproom \@tempdima
2268 <latexrelease | fltrace> \global \advance \colht \@tempdima
2269 <latexrelease | fltrace> \global \advance \dbltopnum \m@ne
2270 <latexrelease | fltrace> \@cons \dbltoplist \@currbox
2271 <*trace>
2272 <latexrelease | fltrace> \fl@trace{dbltopnum (after) =
2273 <latexrelease | fltrace> \the \dbltopnum}%
2274 <latexrelease | fltrace> \fl@trace{***Success: dbltop}%
2275 </trace>
2276 <latexrelease | fltrace> \@inserttrue
2277 <latexrelease | fltrace> \fi
2278 <latexrelease | fltrace> \fi
2279 <*trace>
2280 <latexrelease | fltrace> \else
2281 <latexrelease | fltrace> \fl@trace{Fail: dbltopnum = \the \dbltopnum:
2282 <latexrelease | fltrace> fpstype \the \@fpstype=ORD?}%
2283 <latexrelease | fltrace> \ifnum \@fpstype<\sist@n
2284 <latexrelease | fltrace> \fl@trace{ERROR: !t float not successful
2285 <latexrelease | fltrace> (addtodblcol)}%
2286 <latexrelease | fltrace> \fi
2287 </trace>
2288 <latexrelease | fltrace> \fi
2289 <latexrelease | fltrace> \fi
2290 <latexrelease | fltrace> \if@insert
2291 <latexrelease | fltrace> \else
2292 <*trace>
2293 <latexrelease | fltrace> \fl@trace{put on dbldeferlist}%
2294 </trace>
2295 <latexrelease | fltrace> \@cons\@dbldeferlist\@currbox
2296 <*trace>
2297 <latexrelease | fltrace> \fl@trace{dbldeferlist: \@dbldeferlist: (after)}%
2298 </trace>
2299 <latexrelease | fltrace> \fi
2300 <*trace>
2301 <latexrelease | fltrace> \fl@trace{End of addtodblcol -- locally count:}%
2302 <latexrelease | fltrace> \fl@trace{ dbltop: \the \dbltopnum.}%
2303 </trace>
2304 <latexrelease | fltrace> \endgroup
2305 <*trace>
2306 <latexrelease | fltrace> \fl@trace{End of addtodblcol -- globally count:}%
2307 <latexrelease | fltrace> \fl@trace{dbltop: \the \dbltopnum.}%
2308 </trace>

```

```

2309 <latexrelease | fltrace>}%
2310 <latexrelease | fltrace>\EndIncludeInRelease

```

(End of definition for \@addtodblcol.)

\@addmarginpar

```

2311 <*2kernel>
2312 \def\@addmarginpar{\@next\@marbox\@currlist{\@cons\@freelist\@marbox
2313   \@cons\@freelist\@currbox}\@latexbug\@tempcnta\@ne
2314   \if@twocolumn
2315     \if@firstcolumn \@tempcnta\m@ne \fi
2316   \else
2317     \if@mparswitch
2318       \ifodd\c@page \else\@tempcnta\m@ne \fi
2319     \fi
2320     \if@reversemargin \@tempcnta -\@tempcnta \fi
2321   \fi
2322   \ifnum\@tempcnta <\z@ \global\setbox\@marbox\box\@currbox \fi
2323   \@tempdima\@mparbottom
2324   \advance\@tempdima -\@pageht
2325   \advance\@tempdima\ht\@marbox
2326   \ifdim\@tempdima >\z@
2327     \@latex@warning@no@line {Marginpar on page \thepage\space moved}%
2328   \else
2329     \@tempdima\z@
2330   \fi
2331   \global\@mparbottom\@pageht
2332   \global\advance\@mparbottom\@tempdima
2333   \global\advance\@mparbottom\dp\@marbox
2334   \global\advance\@mparbottom\marginparpush
2335   \advance\@tempdima -\ht\@marbox

```

Putting box movement inside the ‘marbox’:

```

2336   \global\setbox \@marbox
2337     \vbox {\vskip \@tempdima
2338           \box \@marbox}%
2339   \global \ht\@marbox \z@
2340   \global \dp\@marbox \z@

```

Sticking (rather than gluing:-) the ‘marbox’ to the line above, changed vskip to kern:

```

2341   \kern -\@pagedp
2342   \nointerlineskip
2343   \hb@xt@\columnwidth
2344     {\ifnum \@tempcnta >\z@
2345       \hskip\columnwidth \hskip\marginparsep
2346     \else
2347       \hskip -\marginparsep \hskip -\marginparwidth
2348     \fi
2349     \box\@marbox \hss}%

```

For this reason the following code can vanish:

```

\ nobreak           %% No longer needed. CAR92/12
\ vskip -\@tempdima %% No longer needed. CAR92/12

2350   \nointerlineskip
2351   \hbox{\vrule \@height\z@ \@width\z@ \@depth\@pagedp}}

```


(End of definition for \@addmarginpar.)

2.2.3 Kludgeins

This part of the file is part of the implementation of the following two new commands for L^AT_EX₂ε.

`\enlargethispage{<dim>}`

Adds <dim> to the height of the current column only. On the printed page the bottom of this column is extended downwards by exactly <dim> without having any effect on the placement of the footer; this may result in an overprinting.

`\enlargethispage*{<dim>}`

Similar to `\enlargethispage` but it tries to squeeze the column to be printed in as small a space as possible, ie it uses any shrinkability in the column. If the column was not explicitly broken (e.g. with `\pagebreak`) this may result in an overfull box message but except for this it will come out as expected (if you know what to expect).

The star form of this command is dedicated to Leslie Lamport, the other we need for ourselves (FMi, CAR).

These commands may well have unwanted effects if used soon before a `\clearpage`: please give keep them clear of such places.

`\@kludgeins` The insert which makes T_EX do a lot of the necessary work. All we need to put into it is the amount by which the pagegoal should be changed.

```
2352 \newinsert \@kludgeins
2353 \global\dimen\@kludgeins \maxdimen
2354 \global\count\@kludgeins 1000
```

(End of definition for \@kludgeins.)

`\enlargethispage` The user command.
`\enlargethispage*`

```
2355 \gdef \enlargethispage {%
2356   \@ifstar
2357   {%
2358     <*trace>
2359     \fl@trace{Enlarging page height * }%
2360   </trace>
2361     \@enlargepage{\hbox{\kern\p@}}%
2362     {%
2363     <*trace>
2364     \fl@trace{Enlarging page height exactly---}%
2365   </trace>
2366     \@enlargepage\@empty}%
2367   }
```

(End of definition for \enlargethispage and \enlargethispage*.)

`\enlargepage` This actually inserts the insert, after checking for extreme values of the change.

```
2368 \gdef\enlargepage#1#2{%
2369 <*trace>
2370   \fl@trace{\@spaces\@spaces by #2}%
2371 </trace>
2372   \@tempskipa#2\relax
2373   \ifdim \@tempskipa>.5\maxdimen
2374     \@latex@error{Suggested\space extra\space height\space
2375                 (\the\@tempskipa)\space dangerously\space
2376                 large}\@eha
2377   \else
2378     \ifdim \vsize<.5\maxdimen
2379 <*trace>
2380     \fl@trace {Kludgeins added--pagegoal before: \the\pagegoal}%
2381 </trace>
2382     \bsphack
2383     \insert\@kludgeins{#1\vskip-\@tempskipa}%
2384     \esphack
```

This next bit is for tracing only:

```
2385 <*trace>
2386   \ifvmode \par
2387   \fl@trace {Kludgeins added--pagegoal after: \the \pagegoal}%
2388   \fi
2389 </trace>
2390   \else
2391     \@latex@error{Page\space height\space already\space
2392                 too\space large}\@eha
2393   \fi
2394   \fi
2395 }
```

(End of definition for \enlargepage.)

`\ShowFloat` This command provides some information about the contents of a float register. Float registers have internal names of the form `\bx@<Uppercase-letter(s)-or numbers>` and you specify just this letter or letters as the argument, e.g., `\ShowFloat{A}`. (There is not much error recovery if you specify something that isn't a float.)

```
2396 </2kernel>
2397 <*2kernel | latexrelease>
2398 <latexrelease>\IncludeInRelease{2021/11/15}%
2399 <latexrelease>           {\ShowFloat}{Show float register contents}%
2400 \def\ShowFloat#1{\begingroup
2401   \let \fl@trace \fl@tracemessage
2402   \fl@trace{***Float #1 details:}%
2403   \ifcsname bx@#1\endcsname
2404     \expandafter\fl@ShowFloat\csname bx@#1\endcsname
2405   \else
2406     \fl@trace{Not a float!}%
2407   \fi
2408   \endgroup
2409 }
2410 \def\fl@ShowFloat#1{%
2411   \fl@traceval{\count#1}% % this here should be interpreted on day
```

```

2412 \fl@traceval{\ht#1}%
2413 \fl@traceval{\dp#1}%
2414 \fl@traceval{\wd#1}%
2415 {\tracingonline1\showboxbreadth10\showboxdepth3\showbox#1}%
2416 }

```

Here are two definitions from `fltrace` that make the above code work:

```

2417 \def \fl@traceval #1{\fl@trace{\string #1 = \the #1}}
2418 \def \fl@tracemessage #1{\let\@elt\@empty\typeout{LaTeX2e: #1}}
2419 </2ekernel | latexrelease>
2420 <latexrelease>\EndIncludeInRelease
2421 <latexrelease>\IncludeInRelease{0000/00/00}%
2422 <latexrelease> \ShowFloat{Show float register contents}%
2423 <latexrelease>
2424 <latexrelease>\let\ShowFloat\@undefined
2425 <latexrelease>\let\fl@ShowFloat\@undefined
2426 <latexrelease>\let\fl@traceval\@undefined
2427 <latexrelease>\let\fl@tracemessage\@undefined
2428 <latexrelease>\EndIncludeInRelease

```

(End of definition for \ShowFloat.)

2.2.4 Float control

This part implements controllable floats and other changes to the float mechanism.

It provides, at the document level, the following command for inclusion in `LATEX2e`.

```
\suppressfloats
```

This suppresses all further floats on the current page.

With an optional argument it suppresses only floats only in certain positions on the current page.

[t] suppresses only floats at the top of the page [b] suppresses only floats at the bottom of the page

It also enables the use of an extra specifier, `!`, in the location optional argument of a float. If this is present then, just for this particular float, whenever it is processed by the float mechanism the following are ignored:

- all restrictions on the number of floats which can appear;
- all explicit restrictions on the amount of space which should (not) be occupied by floats and/or text.

The mechanism will still attempt to ensure that pages are not overfull.

These specifiers override, for the single float, the suppression commands described above.

In its current form, it also supplies a reasonably exhaustive, and somewhat baroque, means of tracing some aspects of the float mechanism.

More tracing.

```

\fl@trace Set-up tracing for floats independent of other tracing as it produces mega-output. Default
\tracefloatsoff is no tracing.
\tracefloats
\fl@traceval 2429 <*\fltrace>
\tracefloatvals 2430 \def \fl@tracemessage #1{\let\@elt\@empty\typeout{LaTeX2e: #1}}
\fl@tracemessage 2431 \def \tracefloats{\let \fl@trace \fl@tracemessage}
2432 \def \tracefloatsoff {\let \fl@trace \@gobble}
2433 \tracefloatsoff
2434 \def \fl@traceval #1{\fl@trace{\string #1 = \the #1}}
2435 \IncludeInRelease{2015/01/01}{\tracefloatvals}%
2436 {trace float vals}%
2437 \def \tracefloatvals{%

```

As `\@dblfloatplacement` sets `\f@depth` it needs to be run inside a group, otherwise the float placement will test for the wrong value.⁶⁴

```
2438 \begingroup
```

When the user requests `\tracefloatvals` then they should show regardless of the tracing state, so locally we make sure that it is activated.

```

2439 \tracefloats
2440 \@dblfloatplacement
2441 \@floatplacement
2442 \fl@trace{***Float placement parameters:}%
2443 \fl@traceval\@colnum
2444 \fl@traceval\@colroom
2445 \fl@traceval\@topnum
2446 \fl@traceval\@toproom
2447 \fl@traceval\@botnum
2448 \fl@traceval\@botroom
2449 \fl@traceval\@fpmin
2450 \fl@trace{\string\textfraction = \textfraction}%
2451 \fl@traceval\@dbltopnum
2452 \fl@traceval\@dbltoproom
2453 \fl@trace{\string\textfraction = \textfraction}%
2454 \fl@trace{toplist: \@toplist}%
2455 \fl@trace{botlist: \@botlist}%
2456 \fl@trace{midlist: \@midlist}%
2457 \fl@trace{deferlist: \@deferlist}%
2458 \fl@trace{dbltoplist: \@dbltoplist}%
2459 %Fmi \fl@trace{dbldeferlist: \@dbldeferlist}%
2460 \endgroup
2461 }
2462 \EndIncludeInRelease
2463 \IncludeInRelease{0000/00/00}{\tracefloatvals}%
2464 {trace float vals}%
2465 \def \tracefloatvals{%
2466 \begingroup
2467 \tracefloats
2468 \@dblfloatplacement
2469 \@floatplacement
2470 \fl@trace{***Float placement parameters:}%
2471 \fl@traceval\@colnum
2472 \fl@traceval\@colroom

```

⁶⁴This is a somewhat questionable design.

```

2473 \fl@traceval\@topnum
2474 \fl@traceval\@toproom
2475 \fl@traceval\@botnum
2476 \fl@traceval\@botroom
2477 \fl@traceval\@fpmin
2478 \fl@trace{\string\textfraction = \textfraction}%
2479 \fl@traceval\@dbltopnum
2480 \fl@traceval\@dbltoproom
2481 \fl@trace{\string\textfraction = \textfraction}%
2482 \fl@trace{toplist: \@toplist}%
2483 \fl@trace{botlist: \@botlist}%
2484 \fl@trace{midlist: \@midlist}%
2485 \fl@trace{deferlist: \@deferlist}%
2486 \fl@trace{dbltoplist: \@dbltoplist}%
2487 % next line only in old releases
2488 \fl@trace{dbldeferlist: \@dbldeferlist}%
2489 \endgroup
2490 }
2491 \EndIncludeInRelease

```

We need to make sure that `fltrace` comes before `flafter` to make the tracing work.

```

2492 \ifpackageloaded{flafter}
2493 {\PackageWarningNoLine
2494   {fltrace}{Load 'fltrace' before 'flafter'\MessageBreak
2495   Attempting to recover by reloading 'flafter'}}%

```

Hide the fact that `flafter` was already loaded and then request it anew.

```

2496 \expandafter\let\csname ver@flafter.sty\endcsname\relax
2497 \def\reserved@a#1{%
2498   \expandafter\let\csname\string#1+flafter+IIR\endcsname\relax}%
2499 \reserved@a\@addtocurcol
2500 \reserved@a\@addtonextcol
2501 \RequirePackage{flafter}}{}
2502 </fltrace>

```

As the code for `flafter` will contain tracing calls so that it works in conjunction with `fltrace` we need to provide a dummy definition for `\fl@trace` in that package.

```

2503 <flafter>
2504 \providecommand\fl@trace[1]{}
2505 </flafter>

```

(End of definition for \fl@trace and others.)

`\suppressfloats` Float suppression commands: these set the relevant counter globally to zero. Thus they
`\@flstop` are overridden for a particular float by an `!` specifier.

```

2506 <*2kernel>
2507 \def \suppressfloats {%
2508   \ifnextchar [%
2509     \@flstop
2510     {\global \@colnum \z@}%
2511   }

```

Maybe this should be a loop over `#1`?

```

2512 \def \@flstop [#1] {%
2513   \if t#1%

```

```

2514     \global \@topnum \z@
2515     \fi
2516     \if b#1%
2517         \global \@botnum \z@
2518     \fi
2519 }

```

(End of definition for \suppressfloats and \flstop.)

Manipulation of float placement and type; both their strings and the corresponding count registers.

`\@fpstype` First a new count register to go with `\@currtype`.
`\@reqcolroom` Then a new skip register, for information needed to remove the `\@maxsep` conservatism: it is possible that this could use a temporary register.
`\@textfloatsheight`

Finally a dimension register to hold the total height of in-text floats on the current page. This is needed to implement a major change in the functionality of `\@addtocurcol` which is, nevertheless, a bug fix. It is not local and therefore cannot be a temporary register.

```

2520 \newcount \@fpstype
2521 \newdimen \@reqcolroom
2522 \newdimen \@textfloatsheight

```

(End of definition for \@fpstype, \@reqcolroom, and \@textfloatsheight.)

`\saved@reqcolroom` Saved value of `\@reqcolroom`; this is needed when making several tests.

```

2523 </2ekernel>
2524 < *2ekernel | latexrelease >
2525 < latexrelease > \IncludeInRelease{2025/06/01}%
2526 < latexrelease >           {\saved@reqcolroom}{float placement calculation}%
2527 \newdimen \saved@reqcolroom
2528 </2ekernel | latexrelease >
2529 < latexrelease > \EndIncludeInRelease
2530 < latexrelease > \IncludeInRelease{0000/00/00}%
2531 < latexrelease >           {\saved@reqcolroom}{float placement calculation}%
2532 < latexrelease > \let \saved@reqcolroom \@undefined
2533 < latexrelease > \EndIncludeInRelease

```

(End of definition for \saved@reqcolroom.)

`\@fpsadddefault` Adds the default placement to what is already there.
Should not need to change this, but could do it as follows:

```

def \@fpsadddefault {%
  \@temptokena \expandafter\expandafter\expandafter
    {\csname fps@\@capttype \endcsname}%
  \edef \reserved@a {\the\@temptokena}%
  \@onelevel@sanitize \reserved@a
  \edef \@fps {\@fps\reserved@a}%
}

```

```

2534 <*2ekernel | fltrace)
2535 \def \@fpsadddfault {%
2536 <*trace)
2537   \fl@trace{fps changed from: \@fps}%
2538 </trace)
2539   \edef \@fps {\@fps\csname fps@\@capytype \endcsname}%
2540   \@latex@warning {%
2541     No positions in optional float specifier.\MessageBreak
2542     Default added (so using ‘\@fps’)}%
2543 }

```

(End of definition for \@fpsadddfault.)

\@setfloattypcounts Sets counters \@fpstype and \@currtype.

BANG == bit4 of \count\@currbox = 0.

```

2544 \def \@setfloattypcounts {%
2545   \@currtype \count\@currbox
2546   \@fpstype \count\@currbox
2547   \divide\@currtype\@xxxii \multiply\@currtype\@xxxii
2548   \advance \@fpstype -\@currtype
2549 <*trace)
2550   \fl@trace{(mod 32) fpstype: \the \@fpstype}%
2551   \fl@trace{(mult of 32) currtype: \the \@currtype}%
2552 % Tracing only: but some should be changed into real errors/warnings?
2553   \ifnum \@fpstype<\sixt@@n
2554     \ifnum \@fpstype=\z@
2555       \fl@trace{ERROR: no PLACEMENT, fpstype = \the \@fpstype = 0?}%
2556     \fi
2557     \ifnum \@fpstype=\@ne
2558       \fl@trace{WARNING: only h, fpstype = \the \@fpstype = 1?}%
2559     \fi
2560     \fl@trace{BANG float}%
2561   \else
2562     \ifnum \@fpstype=\sixt@@n
2563       \fl@trace{ERROR: no PLACEMENT, fpstype = \the \@fpstype = 16?}%
2564     \fi
2565     \ifnum \@fpstype=17
2566       \fl@trace{WARNING: only h, fpstype = \the \@fpstype = 17?}%
2567     \fi
2568     \fl@trace{ORD float}%
2569   \fi
2570 </trace)
2571 }
2572 </2ekernel | fltrace)

```

(End of definition for \@setfloattypcounts.)

Macros for getting, testing and setting bits of the fps.

\@getfpsbit Sets \@tempcnta to required bit of \count\@currbox.

```

2573 <*2ekernel)
2574 \def \@getfpsbit {%
2575   \@boxfpsbit \@currbox
2576 }

```

(End of definition for \@getfpsbit.)

`\@boxfpsbit` Used above.

```
2577 \def \@boxfpsbit #1#2{%
2578   \@tempcnta \count#1%
2579   \divide \@tempcnta #2\relax
2580 }
```

(End of definition for \@boxfpsbit.)

`\@testfp` New definition of the float page test.

```
2581 \def \@testfp #1{%
2582   \@boxfpsbit #18\relax % Really '#1 8' for human readers!
2583   \ifodd \@tempcnta
2584   \else
2585     \@testtrue
2586   \fi
2587 }
```

(End of definition for \@testfp.)

`\@setfpsbit` Sets required bit of `\@tempcnta` (to 1).

```
2588 \def \@setfpsbit #1{%
2589   \@tempcntb \@tempcnta
2590   \divide \@tempcntb #1\relax
2591   \ifodd \@tempcntb
2592   \else
2593     \advance \@tempcnta #1\relax
2594   \fi
2595 }
2596 </2kernel>
```

(End of definition for \@setfpsbit.)

`\@resethfps` Globally adds `t` as a possible location for an `h` or `lh` only placement: this must be done using the count.

Although it will leave `\@fpstype` set to 17 even if it was originally 1, this does not matter since it is the last thing in `\@addtocurcol`.

```
2597 <*2kernel | fltrace>
2598 \def \@resethfps {%
2599   \let\reserved@a\@empty
2600   \ifnum \@fpstype=\@ne
2601     \def \reserved@a {!}%
2602     \@fpstype 17
2603   \fi
2604   \ifnum \@fpstype=17
2605     \global \advance \count\@currbox \tw@
2606     \@latex@warning@no@line {%
2607       '\reserved@a h' float specifier changed to '\reserved@a ht'}%
2608 <*trace>
2609     \fl@trace{%
2610       't' added to '\reserved@a h'- new Count: \the \count\@currbox}%
2611 </trace>
2612   \fi
2613 }
```


(End of definition for `\@resetfps`.)

Special stuff for BANG floats.

`\@flsetnum` Ignores any zero float counter value in case BANG.

It uses a local assignment to the normally global counter: a bit naughty, perhaps?

These assignments are safe so long as the counter involved is only consulted once (i.e. only for the ‘bang float’) with the changed value. This is the case within `\@addtocurcol` because it is used only once within a call of the output routine (which forms a group).

For `\@addtonextcol` this is achieved by putting a group around its code; this is needed because it is called (by `\@startcolumn`) for each float which was on the deferlist. Almost identical considerations pertain to `\@addtodblcol`. There may be more efficient ways to handle this, but the group seems to be the simplest.

```
2614 \def \@flsetnum #1{%
2615 <*trace>
2616   \fl@trace{fpstype: \the \@fpstype (flsetnum \string#1)}%
2617 </trace>
2618   \ifnum \@fpstype<\sift@n
2619     \ifnum #1=\z@
2620 <*trace>
2621     \fl@trace{BANG float resetting \string#1 to 1}%
2622 </trace>
2623     #1\@ne
2624   \fi
2625 \fi
2626 <*trace>
2627   \fl@trace{#1 (before) = \the #1}%
2628 </trace>
2629 }
```

(End of definition for `\@flsetnum`.)

`\@flsettextmin` This ignores `\textfraction` space restriction in case BANG.

```
2630 \def \@flsettextmin {%
2631 <*trace>
2632   \fl@trace{fpstype: \the \@fpstype (flsettextmin)}%
2633 </trace>
2634   \ifnum \@fpstype<\sift@n
2635 <*trace>
2636     \fl@trace{BANG ignoring textmin}%
2637 </trace>
2638     \@textmin \z@
2639   \else
2640     \@textmin \textfraction\@colht
2641 <*trace>
2642     \fl@trace{ORD textmin = \the \@textmin}%
2643 </trace>
2644   \fi
2645 }
```

(End of definition for `\@flsettextmin`.)

`\@flcheckspace` This ignores space restriction in case BANG; this is still slightly conservative since it does not allow for the fact that, if there is no text in the column then `\textfloatsep` is not needed. Sets `@tempswa` true if there is room for `\@currbox`.

```

2646 </2kernel | fltrace>
2647 <*2kernel | fltrace | latexrelease>
2648 <latexrelease | fltrace>\IncludeInRelease{2025/06/01}%
2649 <latexrelease | fltrace>          {\@flcheckspace}{float placement calculation}%
2650 \def \@flcheckspace #1#2{%

```

When this test is executed, the value of `\@reqcolroom` may no longer be correct due to a previous test that altered it. We therefore reset it to a value saved earlier.

```

2651   \@reqcolroom \saved@reqcolroom
2652   \advance \@reqcolroom
2653   \ifx #2\@empty \textfloatsep \else \floatsep \fi
2654 <*trace>
2655   \fl@trace{colroom = \the \@colroom
2656                                     (flcheckspace \string#1 \string#2)}%
2657   \fl@trace{reqcolroom = \the \@reqcolroom
2658                                     (flcheckspace \string#1 \string#2)}%
2659 </trace>
2660   \ifdim \@colroom>\@reqcolroom
2661     \ifdim #1>\ht\@currbox
2662       \@tempwattrue
2663 <*trace>
2664       \fl@trace{Space OK: #1 = \the #1 > \the \ht \@currbox
2665                                     (flcheckspace \string#1 \string#2)}%
2666 </trace>
2667     \else
2668 <*trace>
2669       \fl@trace{fpstype: \the \@fpstype
2670                                     (flcheckspace \string#1 \string#2)}%
2671 </trace>
2672     \ifnum \@fpstype<\sixt@n
2673 <*trace>
2674       \fl@trace{BANG float ignoring #1
2675                                     (flcheckspace \string#1 \string#2):}%
2676       \fl@trace{\@spaces #1 = \the #1. Ht float: \the \ht \@currbox
2677                                     \space BANG}%
2678 </trace>
2679     \else
2680 <*trace>
2681       \fl@trace{Fail---no room (flcheckspace \string#1 \string#2)
2682                                     (fpstype \the \@fpstype=ORD?):}%
2683       \fl@trace{\@spaces #1 = \the #1. Ht float: \the \ht \@currbox
2684                                     \space ORD?}%
2685 </trace>
2686     \fi
2687   \fi
2688 <*trace>
2689   \else
2690     \fl@trace{Fail---no room at 2nd test of colroom
2691               (flcheckspace \string#1 \string#2)}%
2692 </trace>
2693

```

```

2694 \fi
2695 }
2696 </2ekernel | fltrace | latexrelease>
2697 <latexrelease | fltrace>\EndIncludeInRelease
2698 <latexrelease | fltrace>\IncludeInRelease{0000/00/00}%
2699 <latexrelease | fltrace>          {\@flcheckspace}{float placement calculation}%
2700 <latexrelease | fltrace>
2701 <latexrelease | fltrace>\def \@flcheckspace #1#2{%
2702 <latexrelease | fltrace>  \advance \@reqcolroom
2703 <latexrelease | fltrace>    \ifx #2\@empty \textfloatsep \else \floatsep \fi
2704 <*trace>
2705 <latexrelease | fltrace>  \fl@trace{colroom = \the \@colroom
2706 <latexrelease | fltrace>                                (flcheckspace \string#1 \string#2)}%
2707 <latexrelease | fltrace>  \fl@trace{reqcolroom = \the \@reqcolroom
2708 <latexrelease | fltrace>                                (flcheckspace \string#1 \string#2)}%
2709 </trace>
2710 <latexrelease | fltrace>  \ifdim \@colroom>\@reqcolroom
2711 <latexrelease | fltrace>    \ifdim #1>\ht\@currbox
2712 <latexrelease | fltrace>      \@tempwattrue
2713 <*trace>
2714 <latexrelease | fltrace>      \fl@trace{Space OK: #1 = \the #1 > \the \ht \@currbox
2715 <latexrelease | fltrace>                                (flcheckspace \string#1 \string#2)}%
2716 </trace>
2717 <latexrelease | fltrace>  \else
2718 <*trace>
2719 <latexrelease | fltrace>    \fl@trace{fpstype: \the \@fpstype
2720 <latexrelease | fltrace>                                (flcheckspace \string#1 \string#2)}%
2721 </trace>
2722 <latexrelease | fltrace>  \ifnum \@fpstype<\sift@n
2723 <*trace>
2724 <latexrelease | fltrace>    \fl@trace{BANG float ignoring #1
2725 <latexrelease | fltrace>                                (flcheckspace \string#1 \string#2):}%
2726 <latexrelease | fltrace>  \fl@trace{\@spaces #1 = \the #1. Ht float: \the \ht \@currbox
2727 <latexrelease | fltrace>                                \space BANG}%
2728 </trace>
2729 <latexrelease | fltrace>    \@tempwattrue
2730 <*trace>
2731 <latexrelease | fltrace>  \else
2732 <latexrelease | fltrace>    \fl@trace{Fail---no room (flcheckspace \string#1 \string#2)
2733 <latexrelease | fltrace>                                (fpstype \the \@fpstype=ORD?):}%
2734 <latexrelease | fltrace>  \fl@trace{\@spaces #1 = \the #1. Ht float: \the \ht \@currbox
2735 <latexrelease | fltrace>                                \space ORD}%
2736 </trace>
2737 <latexrelease | fltrace>  \fi
2738 <latexrelease | fltrace>  \fi
2739 <*trace>
2740 <latexrelease | fltrace>  \else
2741 <latexrelease | fltrace>    \fl@trace{Fail---no room at 2nd test of colroom
2742 <latexrelease | fltrace>                                (flcheckspace \string#1 \string#2)}%
2743 </trace>
2744 <latexrelease | fltrace>  \fi
2745 <latexrelease | fltrace>}
2746 <latexrelease | fltrace>\EndIncludeInRelease

```

(End of definition for `\@flcheckspace`.)

`\@flupdates` This updates everything when a float is placed.

```
2747 <*2ekernel>
2748 \def \@flupdates #1#2#3{%
2749   \global \advance #1\m@ne
2750   \global \advance \@colnum \m@ne
2751   \@tempdima -\ht\@currbox
2752   \advance \@tempdima
2753   -\ifx #3\@empty \textfloatsep \else \floatsep \fi
2754   \global \advance #2\@tempdima
2755   \global \advance \@colroom \@tempdima
2756   \@cons #3\@currbox
2757 }
2758 </2ekernel>
```

(End of definition for `\@flupdates`.)

Interesting facts about float mechanisms past and present, together with a summary of various features, some unresolved:

1. The value `\textfraction` does not affect the processing of doublecol floats: this seems sensible, but should be documented.
2. `\twocolumn` floatplacement was wrong: `dbl` not needed, `ord` needed.
3. `\@floatplacement` was not called after `\startdblcol` or `\@topnewpage`. This has been changed; it is clearly a bug fix.
4. The use `\@topnewpage` when `\dblfigrule` is non-trivial produced a rule in the wrong place. This has been fixed by not using `\dblfigrule` when processing the ‘float’ from `\@topnewpage`.
5. If the specifier was just `h` and the float could not be put here, it went on the deferlist and stayed there until a clearpage. It now gets changed to a ‘`th`’: this is only an error-recovery action, putting just `h` or `!h` should be deprecated.
6. `\@dblmaxsep` was ‘the maximum of `\dblfloatsep` and `\dbltextfloatsep`’. But it was never used! Now gone completely, like `\@maxsep`.
7. After an `h` float is put on a page, it was counted as text when applying the `\textfraction` test; this is possibly too big a change although it is a bug fix?
8. Two consecutive `h` floats are separated by twice `\intextsep`: this could be changed to one by use of `\addvspace`, OK? Note that it would also mean that less space is put in if an `h` float immediately follows other spaces. This is also possibly too big a change, at least for compatibility mode? Or it may be simply wrong! It has not been changed.
9. Now `\@addtocurcol` checks first for just `p` fps. I think that this is an increase in efficiency, but maybe the coding should be made even more efficient.
10. `\@tryfcolumn` now tests if the list is empty first, otherwise lots of wasted time! Thus this test has been removed from `\@startcolumn`. As Frank pointed out, this makes `\@startcolumn` less efficient. But it is now the same as `\@startdblcolumn`: I can see no reason why they should be different, but which is best?

11. Why is `\@colroom` set in `\@docclearpage`?
12. Footnotes. Check what `\clearpage` does when footnotes are left over. Footnotes are not put on float pages and, also, `\@addtonextcol` ignores the existence of held-over footnotes in deciding what floats can go on the page. Not changed.
13. `\clearpage` can still lose non-boxes, at least when floats are involved. It also moves some to the ‘wrong page’, but this may be a coding problem.
14. The `!` option makes it necessary to check in `\output` that there is enough room left on the page after adding a float. (This would have been necessary anyway if anyone set `\@textmin` too close to zero! A similar danger existed also if the text in a `\twocolumn[text]` entity gets too large.) The current implementation of this also makes the normal case a little less efficient, OK? Not enough room means, at present, less than `\baselineskip`, with a warning: is this OK? Should it be made generic (another parameter)?
15. There are four possibilities for supporting this:
`\twocolumn[\maketitle more text]`
 One is to change `\maketitle` slightly to allow this. Another is to change `\@topnewpage` so that more than one `\twocolumn[]` command is allowed; in this case `\maketitle\twocolumn[more text]` will work. The former is more robust from the user’s viewpoint, but makes the code for `\maketitle` rather ad hoc (maybe it is already?). Another is to misuse the global `twocolumn` flag locally within `\@topnewpage`. Yet another is to move the column count register from the `multicol` package into the kernel. This has been done.
16. Where should the reinserts be put to maximise the probability that footnotes come out on the correct page? Or should we go for as much compatibility as possible (but see next item)?
17. Should we continue to support (as much as possible) `\samepage`? Some of its intended functionality is now advertised as being provided by `\enlargethispage`. Use of either is likely to result in wrongly placed footnotes, marginals, etc. Which should have priority: obeying the pagination instructions, or correct placement of notes/marginalia?
18. Is the adjustment of space to cause shrinking in the kludge-* case correct? Should it be limited to `0pt`?
19. Is the setting of `\boxmaxdepth` in `makecol` and friends needed? It only has any effect if `\@textbottom` ends with a box or rule, in which case the `vskip` to allow for its depth should also be added. If it is kept, it should probably be the last thing in the box. It has now been removed.

 It would perhaps be better to document that `\@textbottom` and `\@texttop` must have natural height `0pt`.
20. I cannot see why the `vskip` adjustment for the depth is needed if `boxmaxdepth` is used to ensure that there is never a too deep box.
21. The value of `\boxmaxdepth` should be explicitly set whenever necessary: it is too risky to assume that it has any particular value. Care is needed in deciding what to set it to.

It is interesting to note that the value of `\boxmaxdepth` is unique in being read before the local settings for the box group are reset; all other parameter settings which affect the box construction use their values outside the box group.

22. Should `\@maxdepth` store the setting of `\maxdepth` from `lplain`? Or should we provide a proper interface to class files for setting these?

An analysis of various other macros.

`\@opcol` should do `\@floatplacement`, but where? Right at the end, since it always occurs at the start of a column.

```
\def\@opcol{%
  % Why is this done first?
  \global \@mparbottom \z@
  \if@twocolumn
    \@outputdblcol
  \else
    \@outputpage
    % This is not needed since it is done at the end of
    %   |\@outputpage|:
    \global \@colht \textheight
  \fi}
```

Only tracing has been added to these.

```
2759 <latexrelease | fltrace>\IncludeInRelease{2017/01/01}%
2760 <latexrelease | fltrace> {\@makefcolumn}{negative height floats}%
2761 <*2ekernel | fltrace | latexrelease>
2762 \def\@makefcolumn #1{%
2763   \begingroup
2764     \@fpmin -\maxdimen
2765     \let \@testfp \@gobble
2766     \@tryfcolumn #1%
2767   \endgroup
2768 <*trace>
2769   \if@fcolmade
2770     \fl@trace{PAGE: in \string\clearpage
2771               \if@twocolumn ---twocolumn\fi---}%
2772     \fl@trace{----- float column/page completed from \string#1}%
2773   \fi
2774 </trace>
2775 }
2776 <latexrelease | fltrace>\EndIncludeInRelease
2777 <latexrelease | fltrace>\IncludeInRelease{0000/00/00}%
2778 <latexrelease | fltrace> {\@makefcolumn}{negative height floats}%
2779 <latexrelease | fltrace>\def\@makefcolumn #1{%
2780 <latexrelease | fltrace> \begingroup
2781 <latexrelease | fltrace>   \@fpmin \z@
2782 <latexrelease | fltrace>   \let \@testfp \@gobble
2783 <latexrelease | fltrace>   \@tryfcolumn #1%
2784 <latexrelease | fltrace> \endgroup
2785 <*trace>
```

```

2786 <latexrelease | fltrace> \if@fcolmade
2787 <latexrelease | fltrace> \fl@trace{PAGE: in \string\clearpage
2788 <latexrelease | fltrace> \if@twocolumn ---twocolumn\fi---}%
2789 <latexrelease | fltrace> \fl@trace{----- float column/page completed
2790 <latexrelease | fltrace> from \string#1}%
2791 <latexrelease | fltrace> \fi
2792 </trace>
2793 <latexrelease | fltrace>}
2794 <latexrelease | fltrace>\EndIncludeInRelease
2795 </2ekernel | fltrace | latexrelease>

```

This will line up the last baselines in the two columns provided they are constructed in the normal way: i.e. ending in a skip of minus the original depth, with `\@textbottom` adding nothing.

Thus again it is essential for `\@textbottom` to have depth 0pt.

```

2796 <latexrelease | fltrace>\IncludeInRelease{2025/06/01}%
2797 <latexrelease | fltrace> {\@outputdblcol}{Use new mark mechanism}%
2798 <*2ekernel | fltrace | latexrelease>

```

```

2799 \def\@outputdblcol{%
2800 \if@firstcolumn
2801 \global\@firstcolumnfalse

```

Save the left column

```

2802 \global\setbox\@leftcolumn\copy\@outputbox
2803 <fltrace> \fl@trace{PAGE: first column boxed}%
2804 \else
2805 \global\@firstcolumntrue
2806 \setbox\@outputbox\vbox{%
2807 \hb@xt@\textwidth{%
2808 \hb@xt@\columnwidth{\box\@leftcolumn \hss}%
2809 \hfil

```

The color of the `\vrule` should be `\normalcolor` as to not inherit the color from the column.

```

2810 {\normalcolor\vrule \@width\columnseprule}%
2811 \hfil
2812 \hb@xt@\columnwidth{\box\@outputbox \hss}}}%
2813 <fltrace> \fl@trace{PAGE: second column also boxed}%
2814 \@combinedblfloats
2815 \@outputpage
2816 <fltrace> \fl@trace{PAGE: two column page completed}%
2817 \begingroup
2818 \@dblfloatplacement
2819 \@startdblcolumn
2820 \@whiles\if@fcolmade \fi{\@outputpage
2821 <fltrace> \fl@trace{PAGE: double float page completed}%
2822 \@startdblcolumn}%
2823 \endgroup
2824 \fi}%
2825 <latexrelease | fltrace>\EndIncludeInRelease
2826 <latexrelease | fltrace>\IncludeInRelease{2015/01/01}%
2827 <latexrelease | fltrace> {\@outputdblcol}{2 column marks}%
2828 <latexrelease | fltrace>\def\@outputdblcol{%

```

```

2829 <latexrelease | fltrace> \if@firstcolumn
2830 <latexrelease | fltrace> \global\@firstcolumnfalse
2831 <latexrelease | fltrace> \global\setbox\@leftcolumn\copy\@outputbox
2832 <latexrelease | fltrace> \splitmaxdepth\maxdimen
2833 <latexrelease | fltrace> \vbadness\maxdimen
2834 <latexrelease | fltrace> \setbox\@outputbox\vbox{\unvbox\@outputbox\unskip}%
2835 <latexrelease | fltrace> \setbox\@outputbox\vsplit\@outputbox to\maxdimen
2836 <latexrelease | fltrace> \toks@\expandafter{\topmark}%
2837 <latexrelease | fltrace> \xdef\@firstcoltopmark{\the\toks@}%
2838 <latexrelease | fltrace> \toks@\expandafter{\splitfirstmark}%
2839 <latexrelease | fltrace> \xdef\@firstcolfirstmark{\the\toks@}%
2840 <latexrelease | fltrace> \ifx\@firstcolfirstmark\@empty
2841 <latexrelease | fltrace> \global\let\@setmarks\relax
2842 <latexrelease | fltrace> \else
2843 <latexrelease | fltrace> \gdef\@setmarks{%
2844 <latexrelease | fltrace> \let\firstmark\@firstcolfirstmark
2845 <latexrelease | fltrace> \let\topmark\@firstcoltopmark}%
2846 <latexrelease | fltrace> \fi
2847 <latexrelease | fltrace> \else
2848 <latexrelease | fltrace> \global\@firstcolumntrue
2849 <latexrelease | fltrace> \setbox\@outputbox\vbox{%
2850 <latexrelease | fltrace> \hb@xt@\textwidth{%
2851 <latexrelease | fltrace> \hb@xt@\columnwidth{\box\@leftcolumn \hss}%
2852 <latexrelease | fltrace> \hfil
2853 <latexrelease | fltrace> {\normalcolor\vrule \@width\columnseprule}%
2854 <latexrelease | fltrace> \hfil
2855 <latexrelease | fltrace> \hb@xt@\columnwidth{\box\@outputbox \hss}}}%
2856 <latexrelease | fltrace> \@combinedblfloats
2857 <latexrelease | fltrace> \@setmarks
2858 <latexrelease | fltrace> \@outputpage
2859 <latexrelease | fltrace> \begingroup
2860 <latexrelease | fltrace> \@dblfloatplacement
2861 <latexrelease | fltrace> \@startdblcolumn
2862 <latexrelease | fltrace> \@whilesw\if@colmade \fi{\@outputpage
2863 <latexrelease | fltrace> \@startdblcolumn}%
2864 <latexrelease | fltrace> \endgroup
2865 <latexrelease | fltrace> \fi}
2866 <latexrelease | fltrace> \EndIncludeInRelease

2867 <latexrelease | fltrace> \IncludeInRelease{0000/00/00}%
2868 <latexrelease | fltrace> {\@outputdblcol}{2 column marks}%
2869 <latexrelease | fltrace> \def\@outputdblcol{%
2870 <latexrelease | fltrace> \if@firstcolumn
2871 <latexrelease | fltrace> \global \@firstcolumnfalse
2872 <latexrelease | fltrace> \global \setbox\@leftcolumn \box\@outputbox
2873 <latexrelease | fltrace> (*trace)
2874 <latexrelease | fltrace> \fl@trace{PAGE: first column boxed}%
2875 <latexrelease | fltrace> (/trace)
2876 <latexrelease | fltrace> \else
2877 <latexrelease | fltrace> \global \@firstcolumntrue
2878 <latexrelease | fltrace> \setbox\@outputbox \vbox {%
2879 <latexrelease | fltrace> \hb@xt@\textwidth {%
2880 <latexrelease | fltrace> \hb@xt@\columnwidth {%
2881 <latexrelease | fltrace> \box\@leftcolumn \hss}%
2882 <latexrelease | fltrace> \hfil

```



```

2883 <latexrelease | fltrace>          {\normalcolor\vrule
2884 <latexrelease | fltrace>          \@width\columnseprule}%
2885 <latexrelease | fltrace>          \hfil
2886 <latexrelease | fltrace>          \hb@xt@\columnwidth {%
2887 <latexrelease | fltrace>          \box\@outputbox \hss}%
2888 <latexrelease | fltrace>          }%
2889 <latexrelease | fltrace>          }%
2890 <*trace>
2891 <latexrelease | fltrace>          \fl@trace{PAGE: second column also boxed}%
2892 </trace>
2893 <latexrelease | fltrace>          \@combinedblfloats
2894 <latexrelease | fltrace>          \@outputpage
2895 <*trace>
2896 <latexrelease | fltrace>          \fl@trace{PAGE: two column page completed}%
2897 </trace>
2898 <latexrelease | fltrace>          \begingroup
2899 <latexrelease | fltrace>          \dblfloatplacement
2900 <latexrelease | fltrace>          \startdblcolumn

```

This loop could be replaced by an `\expandafter` tail recursion in `\startdblcolumn`.

```

2901 <latexrelease | fltrace>          \@whilesw@if@fcolmade \fi
2902 <latexrelease | fltrace>          {\@outputpage
2903 <*trace>
2904 <latexrelease | fltrace>          \fl@trace{PAGE: double float page completed}%
2905 </trace>
2906 <latexrelease | fltrace>          \startdblcolumn}%
2907 <latexrelease | fltrace>          \endgroup
2908 <latexrelease | fltrace> \fi
2909 <latexrelease | fltrace>}%
2910 <latexrelease | fltrace>\EndIncludeInRelease
2911 </2ekernel | fltrace | latexrelease>

```

2.2.5 Float placement parameters

The main purpose of this section is to ensure that all the float-placement parameters which need to be set in a class file or package have been declared. It also describes their use and sets values for them which are reasonable for typical documents using US letter or A4 sized paper.

Limits for the placement of floating objects

`\c@topnumber` This counter holds the maximum number of floats that can appear at the top of a text page or column.

```

2912 <*2ekernel>
2913 \newcount\c@topnumber
2914 \setcounter{topnumber}{2}

```

(End of definition for \c@topnumber.)

`\topfraction` This macro holds the maximum proportion (as a decimal number) of a text page or column that can be occupied by floats at the top.

```

2915 \newcommand\topfraction{.7}

```

(End of definition for \topfraction.)

\c@bottomnumber This counter holds the maximum number of floats that can appear at the bottom of a text page or column.

2916 `\newcount\c@bottomnumber`

2917 `\setcounter{bottomnumber}{1}`

(End of definition for \c@bottomnumber.)

\bottomfraction This macro holds the maximum proportion (as a decimal number) of a text page or column that can be occupied by floats at the bottom.

2918 `\newcommand\bottomfraction{.3}`

(End of definition for \bottomfraction.)

\c@totalnumber This counter holds the maximum number of floats that can appear on any text page or column.

2919 `\newcount\c@totalnumber`

2920 `\setcounter{totalnumber}{3}`

(End of definition for \c@totalnumber.)

\textfraction This macro holds the minimum proportion (as a decimal number) of a text page or column that must be occupied by text.

2921 `\newcommand\textfraction{.2}`

(End of definition for \textfraction.)

\floatpagefraction This macro holds the minimum proportion (as a decimal number) of a page or column that must be occupied by floating objects before a ‘float page’ is produced.

2922 `\newcommand\floatpagefraction{.5}`

(End of definition for \floatpagefraction.)

\c@dbltopnumber This counter holds the maximum number of double-column floats that can appear on the top of a two-column text page.

2923 `\newcount\c@dbltopnumber`

2924 `\setcounter{dbltopnumber}{2}`

(End of definition for \c@dbltopnumber.)

\dbltopfraction This macro holds the maximum proportion (as a decimal number) of a two-column text page that can be occupied by double-column floats at the top.

2925 `\newcommand\dbltopfraction{.7}`

(End of definition for \dbltopfraction.)

\dblfloatpagefraction This macro holds the minimum proportion (as a decimal number) of a page that must be occupied by double-column floating objects before a ‘double-column float page’ is produced.

2926 `\newcommand\dblfloatpagefraction{.5}`

(End of definition for \dblfloatpagefraction.)

Floats on a text page

`\floatsep`
`\textfloatsep`
`\intertextsep` When a floating object is placed on a page with text, these parameters control the separation between the float and the other objects on the page. These parameters are used for both one-column mode and single-column floats in two-column mode. They are all rubber lengths.

`\floatsep` is the space between adjacent floats that are placed at the top or bottom of the text page or column.

`\textfloatsep` is the space between the main text and floats at the top or bottom of the page or column.

`\intertextsep` is the space between in-text floats and the text.

```
2927 \newskip\floatsep
2928 \newskip\textfloatsep
2929 \newskip\intertextsep
2930 \setlength\floatsep {12\p@ \@plus 2\p@ \@minus 2\p@}
2931 \setlength\textfloatsep{20\p@ \@plus 2\p@ \@minus 4\p@}
2932 \setlength\intertextsep {12\p@ \@plus 2\p@ \@minus 2\p@}
```

(End of definition for `\floatsep`, `\textfloatsep`, and `\intertextsep`.)

`\dblfloatsep`
`\dbltextfloatsep` When double-column floats (floating objects that span the whole `\textwidth`) are placed at the top of a text page in two-column mode, the separation between the float and the text is controlled by `\dblfloatsep` and `\dbltextfloatsep`. They are rubber lengths.

`\dblfloatsep` is the space between adjacent double-column floats placed at the top of the text page.

`\dbltextfloatsep` is the space between the main text and double-column floats at the top of the page.

```
2933 \newskip\dblfloatsep
2934 \newskip\dbltextfloatsep
2935 \setlength\dblfloatsep {12\p@ \@plus 2\p@ \@minus 2\p@}
2936 \setlength\dbltextfloatsep{20\p@ \@plus 2\p@ \@minus 4\p@}
```

(End of definition for `\dblfloatsep` and `\dbltextfloatsep`.)

Floats on their own page or column

`\@fptop`
`\@fpsep`
`\@fpbot` When floating objects are placed on a separate page or column, called a ‘float page’, the layout of the page is controlled by these parameters, which are rubber lengths.

At the top of the page `\@fptop` is inserted; typically this supplies some stretchable whitespace. At the bottom of the page `\@fpbot` is inserted. Between adjacent floats `\@fpsep` is inserted.

These parameters are used for all floating objects on a ‘float page’ in one-column mode, and for single-column floats in two-column mode.

Note that at least one of the two parameters `\@fptop` and `\@fpbot` should contain a `plus ...fil` so as to fill the remaining empty space.

```
2937 \newskip\@fptop
2938 \newskip\@fpsep
2939 \newskip\@fpbot
2940 \setlength\@fptop{0\p@ \@plus 1fil}
2941 \setlength\@fpsep{8\p@ \@plus 2fil}
2942 \setlength\@fpbot{0\p@ \@plus 1fil}
```

(End of definition for `\@fptop`, `\@fpsep`, and `\@fpbot`.)

`\@dblftop` Double-column ‘float pages’ in two-column mode use similar parameters.

`\@dblfpsep` 2943 `\newskip\@dblftop`

`\@dblfpbot` 2944 `\newskip\@dblfpsep`

2945 `\newskip\@dblfpbot`

2946 `\setlength\@dblftop{0\p@ \@plus 1fil}`

2947 `\setlength\@dblfpsep{8\p@ \@plus 2fil}`

2948 `\setlength\@dblfpbot{0\p@ \@plus 1fil}`

(End of definition for \@dblftop, \@dblfpsep, and \@dblfpbot.)

`\topfigrule` The macros can be used to put in rules between floats and text; whatever they insert
`\botfigrule` should be vertical mode material which takes up zero space.

`\dblfigrule` 2949 `\let\topfigrule=\relax`

2950 `\let\botfigrule=\relax`

2951 `\let\dblfigrule=\relax`

2952 `\</2kernel`

(End of definition for \topfigrule, \botfigrule, and \dblfigrule.)

File 55

ltagging.dtx

1 <*2ekernel | latexrelease>
2 \ExplSyntaxOn

1 General support for tagged output

`\SuspendTagging`

`\ResumeTagging`

`\tag_suspend:n`

`\tag_resume:n`

These are the places in code where it is important to stop any tagging activities, e.g., when we are doing trial typesetting that it is done several times. In such a case one must tag only the final version that is actually used, otherwise tagging structures are generated which then do not end up in the PDF and confuse the mechanism. For this we have two commands that can be used in packages: `\SuspendTagging` and `\ResumeTagging` (with corresponding L3 programming layer commands). They are available as part of the \LaTeX kernel, so that they can be safely used in packages whether or not tagging is requested. They all take a string argument that is used for debugging to easily identify why tagging was suspended or restarted, for example, in `tabularx` you find `\SuspendTagging{tabularx}`. By default these four commands do nothing.

The argument is used literally (in `\typeout` messages) without any expansion when debugging is turned on and otherwise it is not used at all. This means it is safe to write something like `\SuspendTagging{\foo}` or even `\SuspendTagging\foo` which means \LaTeX has to parse only a single token instead of putting a string of characters into the argument. This means a tiny speed improvement but with many such debugging strings...

`\UseTaggingSocket`

`\tag_socket_use:n`

`\tag_socket_use:nn`

`\tag_socket_use:nnn`

Given that we sometimes have to suspend tagging, it would be fairly inefficient to put different plugs into these sockets whenever that happens. We therefore offer `\UseTaggingSocket` which is like `\UseSocket` except that it expects a socket starting with `tagsupport/` but the socket name is specified without this prefix, i.e.,

`\UseTaggingSocket{foo} → \UseSocket{tagsupport/foo}`

Beside being slightly shorter, the big advantage is that this way we can change `\UseTaggingSocket` to do nothing by switching a boolean instead of changing the plugs of the tagging support sockets back and forth.

Usually, these sockets have (beside the default plug defined for every socket) one additional plug defined and directly assigned. This plug is used when tagging is active. There may be more plugs, e.g., tagging with special debugging or special behavior depending on the class or PDF version etc., but right now it is usually just on or off.

When tagging is suspended they all have the same predefined behavior: The sockets with zero arguments do nothing. The sockets with one argument gobble their argument. The sockets with two arguments will drop their first argument and pass the second unchanged.

It is possible to use the tagging support sockets with `\UseSocket` directly, but in this case the socket remains active if `\SuspendTagging` is in force. There may be reasons for doing that but in general we expect to always use `\UseTaggingSocket`.

`\UseExpandableTaggingSocket`

`\tag_socket_use_expandable:nof`

For special cases like in some `\halign` contexts we need a fully expandable version of the command. For these cases, `\UseExpandableTaggingSocket` can be used. To allow

being expandable, it does not output any debugging information if `\DebugSocketsOn` is in effect and therefore should be avoided whenever possible.

The L3 programming layer versions `\tag_socket_use_expandable:n`, `\tag_socket_use:n`, `\tag_socket_use:nn`, and `\tag_socket_use:nnn` are slightly more efficient than `\UseTaggingSocket` because they do not have to determine how many arguments the socket takes when disabling it.

`\MathCollectTrue` The tagging of math has to collect/grab the math first. This is not wanted for all
`\MathCollectFalse` uses of $\$$. These command allow to control the behavior of the math shift token. Without the math tagging code they do nothing. Their behavior with the math tagging code is documented in latex-lab-math.pdf

2 Implementation

`\tag_suspend:n` In the kernel, these commands get dummy definitions so that they can be used without
`\tag_resume:n` harm in packages. The real definition is used when tagging gets enabled.
`\SuspendTagging` 3 `\cs_new_eq:NN \tag_suspend:n \use_none:n`
`\ResumeTagging` 4 `\cs_new_eq:NN \tag_resume:n \use_none:n`
 5 `\cs_new_protected:Npn \SuspendTagging #1 { \tag_suspend:n {#1} }`
 6 `\cs_new_protected:Npn \ResumeTagging #1 { \tag_resume:n {#1} }`

(End of definition for `\tag_suspend:n` and others.)

`\tag_socket_use:n` Again this is not the final definition for the kernel; it is just a version to get going while
`\tag_socket_use:nn` some parts of the kernel support are still missing.
`\tag_socket_use:nnn` 7 `\prg_new_conditional:Npnn \tag_if_active: { p , T , TF, F }`
`\tag_socket_use_expandable:n` 8 `{ \prg_return_false: }`

`\UseTaggingSocket` Dummy definitions in the kernel. These definitions will get updated in tagpdf. The
`\UseExpandableTaggingSocket` default in the kernel is simply to get rid of the first argument, while the second argument is preserved if present:

```

9 \cs_new:Npn \tag_socket_use_expandable:n #1 { }
10 \cs_new_protected:Npn \tag_socket_use:n #1 { }
11 \cs_new_protected:Npn \tag_socket_use:nn #1#2 { }
12 \cs_new_protected:Npn \tag_socket_use:nnn #1#2#3 { #3 }
13 \cs_new_protected:Npn \UseTaggingSocket #1 {
14   \int_case:nnF
15     { \int_use:c { c__socket_tagsupport/#1_args_int } }
16     {
17       0 \prg_do_nothing:
18       1 \use_none:n
19       2 \use_ii:nn

```

We do not expect tagging sockets with more than one or two arguments, so for now we only provide those.

```

20     }
21     \ERRORusetaggingsocket % that should get a proper error message
22   }

```

The same as an expandable command:

```

23 \cs_new:Npn \UseExpandableTaggingSocket #1 {
24   \int_case:nnF
25     { \int_use:c { c__socket_tagsupport/#1_args_int } }
26     {

```

```

27         0 \prg_do_nothing:
28         1 \use_none:n
29         2 \use_ii:nn
30     }
31     \ERRORusetaggingsocket % that should get a proper error message
32 }

```

(End of definition for `\tag_socket_use:n` and others.)

2.1 Math collection

The documentation is in latex-lab-math.

```

\MathCollectTrue
\MathCollectFalse 33 \cs_new_protected:Npn\MathCollectTrue{}
34 \cs_new_protected:Npn\MathCollectFalse{}

```

(End of definition for `\MathCollectTrue` and `\MathCollectFalse`.)

2.2 Tagging sockets

This collects tagging sockets that should be generally available so that they can also be used even if the tagging code is not loaded.

2.2.1 Tagging support for paragraph setup

Paragraphs are tagged through the code in the para/hooks. This code is sometimes adjusted, e.g. to produce a “flattened” paragraph or to use a different tag. Sockets related to such code parts are collected here.

```

\l_tag_block_flattened_level_int

```

The block code needs to know if they are nested blockenvs inside a flattened environment. For this it uses a counter. Inside some contexts, e.g. at the begin of a minipage or a footnote this counter must be reset. We therefore define the counter here so that we can use it in the following socket.

```

35 \int_new:N \l__tag_block_flattened_level_int

```

(End of definition for `\l__tag_block_flattened_level_int`.)

`tagsupport/para/restore` (socket) This socket restores the para related settings to their default. It should be used in places where “normal” paragraph tagging must be ensured, for example at the begin of a footnote.

```

36 \NewSocket{tagsupport/para/restore}{0}

```

default (plug)

```

37 \NewSocketPlug{tagsupport/para/restore}{default}
38 {
39     \tl_set:Nn \l__tag_para_main_tag_tl {text-unit}
40     \tl_set_eq:NN \l__tag_para_tag_tl \l__tag_para_tag_default_tl
41     \bool_set_false:N \l__tag_para_flattened_bool
42     \int_zero:N \l__tag_block_flattened_level_int
43     \bool_set_true:N \l__tag_para_bool
44 }
45 \AssignSocketPlug{tagsupport/para/restore}{default}

```

`tagsupport/para/begin` (*socket*) These sockets are currently defined in tagpdf. They overwrite definitions in the latex-lab-
`tagsupport/para/end` (*socket*) block code. There is also a simpler definition that probably should be a general socket
too. TODO: move this into ltagging.

2.2.2 Tagging socket for targets

`tagsupport/refstepcounter` (*socket*) When tagging is active we want to track the current structure number when targets are
set. This will be mostly used in `\refstepcounter` but also if targets are set manually.

```
46 \NewSocket{tagsupport/recordtarget}{0}
```

`tagsupport/recordtarget` (*plug*)

```
47 %
48 \NewSocketPlug{tagsupport/recordtarget}{kernel}
49 {
50   \t1_if_blank:VF \@currentHref
51   {
52     \prop_gput:Nee
53       \g__tag_struct_dest_num_prop
54       {\@currentHref}
55       {\tag_get:n{struct_num}}
56   }
57 }
58 \AssignSocketPlug{tagsupport/recordtarget}{kernel}
59 \ExplSyntaxOff
```

2.2.3 Tagging sockets for toc

`tagsupport/toc/contentsline/before` (*socket*) Tagging sockets at the begin and end of contentsline. They receive *all* contentsline
`tagsupport/toc/contentsline/after` (*socket*) arguments as one argument in four brace groups. The socket code should then use the
parts it needs.

```
60 \NewSocket{tagsupport/toc/contentsline/before}{1}
61 \NewSocket{tagsupport/toc/contentsline/after}{1}

62 \AssignSocketPlug{tagsupport/toc/contentsline/before}{noop}
63 \AssignSocketPlug{tagsupport/toc/contentsline/after}{noop}
```

`tagsupport/toc/starttoc/before` (*socket*) Tagging sockets for the begin and end of start of `\@starttoc`. They take one argument,
`tagsupport/toc/starttoc/after` (*socket*) the extension.

```
64 \NewSocket{tagsupport/toc/starttoc/before}{1}
65 \NewSocket{tagsupport/toc/starttoc/after}{1}

66 \AssignSocketPlug{tagsupport/toc/starttoc/before}{noop}
67 \AssignSocketPlug{tagsupport/toc/starttoc/after}{noop}
```

`tagsupport/toc/leaders/before` (*socket*) Tagging sockets to make the dot leaders an artifact. They do not take an argument.

`tagsupport/toc/leaders/after` (*socket*)

```
68 \NewSocket{tagsupport/toc/leaders/before}{0}
69 \NewSocket{tagsupport/toc/leaders/after}{0}
```

2.2.4 Tagging support for marginpar

`tagsupport/marginpar/begin` (*socket*)

`tagsupport/marginpar/end` (*socket*)

```
70 \NewSocket{tagsupport/marginpar/begin}{0}
71 \NewSocket{tagsupport/marginpar/end}{0}
```


2.2.5 Tagging support for table/tabular packages

The code uses a number of sockets to inject the tagging commands. These can be easily set to a noop-plug in case the automated tagging is not wanted.

<code>tagsupport/tbl/cell/begin</code> (socket)	At first sockets for the begin and end of cells and table rows:
<code>tagsupport/tbl/cell/end</code> (socket)	72 <code>\NewSocket{tagsupport/tbl/cell/begin}{0}</code>
<code>tagsupport/tbl/pcell/end</code> (socket)	73 <code>\NewSocket{tagsupport/tbl/cell/end}{0}</code>
<code>tagsupport/tbl/pcell/end</code> (socket)	74 <code>\NewSocket{tagsupport/tbl/row/begin}{0}</code>
<code>tagsupport/tbl/row/begin</code> (socket)	75 <code>\NewSocket{tagsupport/tbl/row/end}{0}</code>
<code>tagsupport/tbl/row/end</code> (socket)	Multi-line cells have their own sockets (as they start out in vertical mode and need different treatment).
	76 <code>\NewSocket{tagsupport/tbl/pcell/begin}{0}</code>
	77 <code>\NewSocket{tagsupport/tbl/pcell/end}{0}</code>
<code>tagsupport/tbl/init</code> (socket)	This socket should be at the begin of the table, inside a group. It is used for settings such as disabling para-tagging inside the table. This socket can perhaps be merged later into the begin-sockets.
	78 <code>\NewSocket{tagsupport/tbl/init}{0}</code>
<code>tagsupport/tbl/init/celldata</code> (socket)	This socket is used in <code>\tbl_init_cell_data_for_table</code> , the command that stores and initialize cell data to handle nested tables. It can be used to restore similar tagging related values
	79 <code>\NewSocket{tagsupport/tbl/init/celldata}{0}</code>
<code>tagsupport/tbl/finalize</code> (socket)	To fine tune the structure (change cells to header cells, remove unwanted structures, move a foot to the end, etc.). We also need a socket that is executed at the end of the table but <i>before</i> all the variables are restored to the outer or default values. The code in the socket can make assignments, but probably shouldn't do typesetting and not write whatsits.
	80 <code>\NewSocket{tagsupport/tbl/finalize}{0}</code>
<code>tagsupport/tbl/restore/celldata</code> (socket)	This socket is used in <code>\tbl_restore_outer_cell_data:</code> , the command that restores cell data when quitting a nested table. It can be used to restore similar tagging related values
	81 <code>\NewSocket{tagsupport/tbl/restore/celldata}{0}</code>
<code>tagsupport/tbl/colspan</code> (socket)	This socket is used to manage spanning cells, e.g., a <code>\multicolumn</code> . It expects one argument (the number of cells spanned) and if tagging is enabled set appropriate tag attributes in the background. We probably need a similar socket for row spans eventually.
	82 <code>\NewSocket{tagsupport/tbl/colspan}{1}</code>
	83 <code>\AssignSocketPlug{tagsupport/tbl/colspan}{noop}</code>
<code>tagsupport/tbl/hmode/begin</code> (socket)	These sockets are used in the begin and end code of environments, to allow a fast enabling and disabling of the tagging. We distinguish between tables that can be used inside paragraphs and standalone tables such as <code>longtable</code> that are always in vertical mode.
<code>tagsupport/tbl/hmode/end</code> (socket)	
<code>tagsupport/tbl/vmode/begin</code> (socket)	
<code>tagsupport/tbl/vmode/end</code> (socket)	
	84 <code>\NewSocket{tagsupport/tbl/hmode/begin}{0}</code>
	85 <code>\NewSocket{tagsupport/tbl/hmode/end}{0}</code>
	86 <code>\NewSocket{tagsupport/tbl/vmode/begin}{0}</code>
	87 <code>\NewSocket{tagsupport/tbl/vmode/end}{0}</code>

`port/tbl/longtable/init (socket)` `longtable` needs its own sockets to fine tune the structure. Simply switching the plug in the previous socket interferes with enabling/disabling the tagging.

```
88 \NewSocket{tagsupport/tbl/longtable/init}{0}
89 \NewSocket{tagsupport/tbl/longtable/finalize}{0}
```

`port/tbl/longtable/head (socket)` Header and footer boxes need special handling because they are repeatedly used.

```
port/tbl/longtable/foot (socket) 90 \NewSocket{tagsupport/tbl/longtable/head}{0}
91 \NewSocket{tagsupport/tbl/longtable/foot}{0}
```

`port/tbl/leaders/begin (socket)` Sockets around leaders such as rules or dotted lines, that should be tagged as artifacts, `support/tbl/leaders/end (socket)` used, for example, in `\cline`.

```
92 \NewSocket{tagsupport/tbl/leaders/begin}{0}
93 \NewSocket{tagsupport/tbl/leaders/end}{0}
```

2.2.6 Tagging Support for floats

`port/float/hmode/begin (socket)` These sockets are used if the float is called in `hmode`.

```
support/float/hmode/end (socket) 94 \NewSocket{tagsupport/float/hmode/begin}{0}
95 \NewSocket{tagsupport/float/hmode/end}{0}
```

`tagsupport/float/begin (socket)` These sockets start and stop the float structure.

```
tagsupport/float/end (socket) 96 \NewSocket{tagsupport/float/begin}{0}
97 \NewSocket{tagsupport/float/end}{0}
```

`tagsupport/caption/begin (socket)` These sockets are used in `\@makecaption`. They open and close the `Caption` structure.

`tagsupport/caption/end (socket)` Their default plugs assume that they are used in `vmode`. The argument of the `begin` socket is the structure number of the parent float. If it is empty the current structure number is used.

```
98 \NewSocket{tagsupport/caption/begin}{1}
99 \NewSocket{tagsupport/caption/end}{0}

100 \AssignSocketPlug{tagsupport/caption/begin}{noop}
```

`port/caption/label/begin (socket)` These sockets are used in `\@makecaption` around the label. Their default plugs ensure `port/caption/label/end (socket)` that the label is outside the paragraph and that the rest of the caption uses flattened para mode. If the caption is not in a `hbox`, the `para/begin` socket should follow to properly start the paragraph.

```
101 \NewSocket{tagsupport/caption/label/begin}{0}
102 \NewSocket{tagsupport/caption/label/end}{0}
```

2.3 Tagging support for output routines

`port/build/page/header (socket)` These sockets receive the formatted running header/footer in its second argument (the `port/build/page/footer (socket)` first is not used) following the convention of tagging sockets, i.e., only the second argument is processed if tagging is not active

```
103 \NewSocket{tagsupport/build/page/header}{2}
104 \NewSocket{tagsupport/build/page/footer}{2}

105 \NewSocketPlug{tagsupport/build/page/header}{transparent}{#2}
106 \NewSocketPlug{tagsupport/build/page/footer}{transparent}{#2}

107 \AssignSocketPlug{tagsupport/build/page/header}{transparent}
108 \AssignSocketPlug{tagsupport/build/page/footer}{transparent}
```

`\build/column/outputbox` (*socket*) This socket is used to add any missing tagging structures to the `\@outputbox` box, if necessary.

```
109 \NewSocket{tagsupport/build/column/outputbox}{0}
```

`\t/build/column/footins` (*socket*) This socket is used to add any missing tagging structures to the `\footins` box, if necessary.

```
110 \NewSocket{tagsupport/build/column/footins}{0}
```

`tagsupport/page@sofar` (*socket*) This socket is declared and used in the multicol output routines. Only listed for reference.

```
111 %\NewSocket{tagsupport/page@sofar}{0}
```

2.4 Tagging support for math

2.4.1 General sockets

The following sockets are the main math sockets.

```
112 \NewSocket{tagsupport/math/inline/begin}{0}
```

```
113 \NewSocket{tagsupport/math/inline/end}{0}
```

```
114 \NewSocket{tagsupport/math/inline/formula/begin}{2}
```

```
115 \NewSocket{tagsupport/math/inline/formula/end}{0}
```

```
116 \NewSocket{tagsupport/math/display/begin}{0}
```

```
117 \NewSocket{tagsupport/math/display/end}{0}
```

```
118 \NewSocket{tagsupport/math/display/formula/begin}{2}
```

```
119 \NewSocket{tagsupport/math/display/formula/end}{0}
```

```
120 \NewSocket{tagsupport/math/display/tag/begin}{0}
```

```
121 \NewSocket{tagsupport/math/display/tag/end}{0}
```

Assign transparent socket to the sockets with two arguments:

```
122 \NewSocketPlug{tagsupport/math/inline/formula/begin}{transparent}{#2}
```

```
123 \AssignSocketPlug{tagsupport/math/inline/formula/begin}{transparent}
```

```
124 \NewSocketPlug{tagsupport/math/display/formula/begin}{transparent}{#2}
```

```
125 \AssignSocketPlug{tagsupport/math/display/formula/begin}{transparent}
```

2.4.2 Sockets specific for luamml

Save sockets These sockets are wrappers around the `\luamml_save:...` commands. They take an argument which should contain the argument of the save command.

`\t/math/luamml/save/nn` (*socket*) The argument should contain the two arguments of the command.

```
126 \NewSocket{tagsupport/math/luamml/save/nn}{1}
```

```
127 \AssignSocketPlug{tagsupport/math/luamml/save/nn}{noop}
```

`\t/math/luamml/save/nNn` (*socket*) The argument should contain the three arguments of the command.

```
128 \NewSocket{tagsupport/math/luamml/save/nNn}{1}
```

```
129 \AssignSocketPlug{tagsupport/math/luamml/save/nNn}{noop}
```

Socket to annotate

`\luamml/annotate/false` (*socket*) These socket can be used for content that should be annotated with `core=false`

```
130 \NewSocket{tagsupport/math/luamml/annotate/false}{2}
```

```
131 \NewSocketPlug{tagsupport/math/luamml/annotate/false}{transparent}{#2}
```

```
132 \AssignSocketPlug{tagsupport/math/luamml/annotate/false}{transparent}
```

Array sockets These sockets will be used in `array` to add `luamml` support to the array environment.

`\math/luamml/array/save` (*socket*) This socket will be used in `\endarray`. The plug is set by `luamml`.

```
133 \NewSocket{tagsupport/math/luamml/array/save}{0}
```

`\luamml/array/finalize` (*socket*) This socket will be used in `\endarray`. The plug is set by `luamml`.

```
134 \NewSocket{tagsupport/math/luamml/array/finalize}{0}
```

`\math/luamml/array/initcol` (*socket*) This socket will be used in `\@classz`. The plug is set by `luamml`.

```
135 \NewSocket{tagsupport/math/luamml/array/initcol}{0}
```

`\luamml/array/finalizecol` (*socket*) This socket will be used in `\@classz`. The plug is set by `luamml`. The argument sets the type of the column.

```
136 \NewSocket{tagsupport/math/luamml/array/finalizecol}{1}
```

```
137 \AssignSocketPlug{tagsupport/math/luamml/array/finalizecol}{noop}
```

Alignment environments Multiline environments like `align`, `multline` or `gather` are tagged as `mtable`.

`\luamml/mtable/finalizecol` (*socket*) This sockets is used at the end of alignment cells and adds them to the row. The argument passes a type like `last` or `box`.

```
138 \NewSocket{tagsupport/math/luamml/mtable/finalizecol}{1}
```

```
139 \AssignSocketPlug{tagsupport/math/luamml/mtable/finalizecol}{noop}
```

`\luamml/mtable/finalize` (*socket*) This sockets is used at the end of alignment environment to finalize the `mtable` code. It should be used normally with `\UseExpandableTaggingSocket`.

```
140 \NewSocket{tagsupport/math/luamml/mtable/finalize}{1}
```

```
141 \AssignSocketPlug{tagsupport/math/luamml/mtable/finalize}{noop}
```

`\luamml/mtable/aligncol` (*socket*) This sockets is used in `multline` to add attributes describing the alignment to the left and right. It takes an argument, the alignment.

```
142 \NewSocket{tagsupport/math/luamml/mtable/aligncol}{1}
```

```
143 \AssignSocketPlug{tagsupport/math/luamml/mtable/aligncol}{noop}
```

`\mtable/innertable/save` (*socket*) This socket is used in `\endaligned` to save the table. It takes no argument.

```
144 \NewSocket{tagsupport/math/luamml/mtable/innertable/save}{0}
```

`\mtable/smallmatrix/save` (*socket*) This socket is used in `\endsmallmatrix` to save the table. It takes no argument. TODO: Check if this socket and the `innertable` socket can/should be merged into a more generic version.

```
145 \NewSocket{tagsupport/math/luamml/mtable/smallmatrix/save}{0}
```

`\mtable/innertable/finalize` (*socket*) This socket is used e.g. in `\endsmallmatrix` and `\gathered` to finalize the table. It takes no argument.

```
146 \NewSocket{tagsupport/math/luamml/mtable/innertable/finalize}{0}
```

`\luamml/mtable/tag/save` (*socket*) This socket is used to save a tag for later use. has been save before.

```
147 \NewSocket{tagsupport/math/luamml/mtable/tag/save}{0}
```

`\luamml/mtable/tag/set` (*socket*) This socket should be used when a tag is placed. It inserts a tag that has been save before.

```
148 \NewSocket{tagsupport/math/luamml/mtable/tag/set}{0}
```

mbox socket

`tagsupport/math/luamml/hbox` (*socket*) This socket is used around `\hbox` inside an `\mbox`, `\makebox` and `\text` and annotates the content if the `\hbox` is used inside math. The real plug is set by `luamml` but a default plug is defined here so that the socket can also be used if `luamml` is not used.

```
149 \NewSocket{tagsupport/math/luamml/hbox}{2}
150 \NewSocketPlug{tagsupport/math/luamml/hbox}{transparent}{#2}
151 \AssignSocketPlug{tagsupport/math/luamml/hbox}{transparent}
```

math phantom sockets

`tagsupport/math/luamml/finph@nt` (*socket*) This socket handles the annotation of `\finph@nt`

```
152 \NewSocket{tagsupport/math/luamml/finph@nt}{2}
153 \NewSocketPlug{tagsupport/math/luamml/finph@nt}{transparent}{#2}
154 \AssignSocketPlug{tagsupport/math/luamml/finph@nt}{transparent}
```

`tagsupport/math/luamml/finsh` (*socket*) This socket handles the annotation of `\finsh`

```
155 \NewSocket{tagsupport/math/luamml/finsh}{2}
156 \NewSocketPlug{tagsupport/math/luamml/finsh}{transparent}{#2}
157 \AssignSocketPlug{tagsupport/math/luamml/finsh}{transparent}
```

Artifact root sign

`tagsupport/math/luamml/artifact` (*socket*) Unicode characters like a root sign should be marked as artifacts to avoid duplication, e.g., in derivation if `mathml` structure elements are used that imply the meaning.

```
158 \NewSocket{tagsupport/math/luamml/artifact}{0}
```

3 For `ltagging.dtx` parked here for now

```
159 <@=tbl>
160 \ExplSyntaxOn
```

3.1 Variables for row, column and span counting

This part needs a decision on names for various integer registers as well as a decision if those should be also made available for $\text{\LaTeX} 2_{\epsilon}$ -style packages in form of 2e names and or as non-internals for the L3 programming layer.

At the moment they are all internal but this probably has to change.

`\g__tbl_col_int` `\g__tbl_row_int` holds the current row number in the table. The value 0 means we haven't yet processed the table preamble (or in case of `longtable` are just in front of the next chunk to be processed). It is incremented by every `\cr` including the one ending the table preamble.

TODO: due to the gymnastics needed inside the `longtable` code the row counter is directly exposed there rather than hidden by interfaces. This needs changing when it is decided how to manage these counters.

`\g__tbl_col_int` holds the current column number. The value 0 means we have not yet started the table or just finished a table row (with `\` typically); any other positive value means we are currently typesetting a cell in that column in some row (denoted by the `\g__tbl_row_int`).

In a `\multicolumn` it holds the column number of the first spanned column and `\g__tbl_span_tl` the info how many cells are spanned.

`\g__tbl_span_tl` is normally 1 except in a `\multicolumn` cell.

```

161 \int_new:N \g__tbl_col_int
162 \int_new:N \g__tbl_row_int
163 \tl_new:N \g__tbl_span_tl
164 \tl_new:N \g__tbl_table_cols_tl
165
166 \tl_gset:Nn \g__tbl_span_tl {1}
167 \tl_gset:Nn \g__tbl_table_cols_tl {0} % indicates outer level

```

(End of definition for `\g__tbl_col_int` and others.)

`\l__tbl_saved_col_tl` `\l__tbl_saved_row_tl` `\l__tbl_saved_span_tl` `\l__tbl_saved_table_cols_tl` Saving the outer values if we are nesting tables is necessary (as the above variables are globally altered). For this we always use token lists because they don't change and we do not need to blow additional integer registers.

```

168 \tl_new:N \l__tbl_saved_col_tl
169 \tl_new:N \l__tbl_saved_row_tl
170 \tl_new:N \l__tbl_saved_span_tl
171 \tl_new:N \l__tbl_saved_table_cols_tl
172
173 \tl_set:Nn \l__tbl_saved_col_tl{0}
174 \tl_set:Nn \l__tbl_saved_row_tl{0}
175 \tl_set:Nn \l__tbl_saved_span_tl{1}
176 \tl_set:Nn \l__tbl_saved_table_cols_tl{0} % indicates outer level

```

(End of definition for `\l__tbl_saved_col_tl` and others.)

`\g__tbl_missingcells_int` This will contain the number of missing cells in a row:

```

177 \int_new:N \g__tbl_missing_cells_int

```

(End of definition for `\g__tbl_missingcells_int`.)

3.2 Tracing/debugging

`\DebugTablesOn`
`\DebugTablesOff`

```

178 \def\DebugTablesOn{
179   \cs_set_eq:NN \__tbl_trace:n \typeout
180 }
181 \def\DebugTablesOff{
182   \cs_set_eq:NN \__tbl_trace:n \use_none:n
183 }
184 \cs_new_eq:NN \__tbl_trace:n \use_none:n

```

(End of definition for `\DebugTablesOn` and `\DebugTablesOff`.)

3.3 Interface commands

All interface commands for the cell number determination have to be public on some level because they are needed in other packages as well, e.g., longtable. We may or may not also want to provide 2e style names for them.

`\tbl_update_cell_data:` Updating cell data in columns after the first means we have to increment the `\g__tbl_col_int` by the span count of the previous cell (in case it was a `\multicolumn`) and then reset the `\g__tbl_span_tl` to one (as the default).

```

185 \cs_new_protected:Npn \tbl_update_cell_data: {
186     \int_gadd:Nn \g__tbl_col_int { \g__tbl_span_tl }
187     \tl_gset:Nn \g__tbl_span_tl {1}
188 }

```

(End of definition for \tbl_update_cell_data:.)

`\tbl_count_table_cols:` Current implementation of `\@mkpream` uses the scratch counter `\count@` to keep track of the number of toks registers it needs (2 per column), but this can't be used as it counts also insertions made with `!{}` and `@{}`. So similar as does longtable for `\LT@cols` we count the numbers of ampersands instead.

```

189 \cs_new:Npn \tbl_count_table_cols: {
190     \seq_set_split:NnV\l__tbl_tmpa_seq {&}\@preamble
191     \tl_gset:Ne \g__tbl_table_cols_tl { \seq_count:N \l__tbl_tmpa_seq }
192     \__tbl_trace:n { ==>~ Table~ has~ \g__tbl_table_cols_tl \space columns }
193 }

```

(End of definition for \tbl_count_table_cols:.)

`\l__tbl_tmpa_seq`

```

194 \seq_new:N \l__tbl_tmpa_seq

```

(End of definition for \l__tbl_tmpa_seq.)

`\tbl_count_missing_cells:n` We might have the situation that some table package has not implemented the `\tbl_count_table_cols:` in which case `\g__tbl_table_cols_tl` would always be zero and we would get an error below when we try to determine the missing cells, so bypass that calculation if we aren't doing tagging (there the packages should have the proper code added). Recall that this is code, that is called by `\` and an old table packagee might rely on whatever the L^AT_EX kernel offers here.

```

195 \cs_new:Npn \tbl_count_missing_cells:n #1 {
196     \tag_if_active:T {
197         \int_compare:nNnT \g__tbl_col_int > 0
198         {
199             \int_gset:Nn \g__tbl_missing_cells_int
200             {
201                 \g__tbl_table_cols_tl
202                 - \g__tbl_col_int
203                 - \g__tbl_span_tl
204                 + 1
205             }
206             \int_compare:nNnT \g__tbl_missing_cells_int < 0 \ERRORmissingcells % should not happen
207             \__tbl_trace:n{==>~
208                 (#1)~
209                 This~ row~ needs~

```

```

210         \int_use:N \g__tbl_missing_cells_int \space
211         additional~ cell(s)
212     }
213 }
214 }
215 }

```

(End of definition for \tbl_count_missing_cells:n.)

\tbl_save_outer_table_cols:

```

216 \cs_new_protected:Npn \tbl_save_outer_table_cols: {
217   \tl_set_eq:NN \l__tbl_saved_table_cols_tl \g__tbl_table_cols_tl
218 }

```

(End of definition for \tbl_save_outer_table_cols:.)

\tbl_init_cell_data_for_table:

```

219 \cs_new_protected:Npn \tbl_init_cell_data_for_table: {
220   \tl_set:No \l__tbl_saved_col_tl {\int_use:N \g__tbl_col_int }
221   \tl_set:No \l__tbl_saved_row_tl {\int_use:N \g__tbl_row_int }
222   \tl_set_eq:NN \l__tbl_saved_span_tl \g__tbl_span_tl
223   %
224   \__tbl_trace:n { ==>~ saved~cell~data:~
225                   \l__tbl_saved_row_tl,
226                   \l__tbl_saved_col_tl,
227                   \l__tbl_saved_span_tl \space
228                   (
229                     \int_compare:nNnTF \l__tbl_saved_table_cols_tl = 0
230                       { outer~ level }
231                       { max:~ \l__tbl_saved_table_cols_tl }
232                   )
233   }

```

Tagging has to initialize cell data too.

```

234   \UseTaggingSocket{tbl/init/celldata}

```

These are the initial values when starting a table:

```

235   \int_gzero:N \g__tbl_row_int
236   \int_gzero:N \g__tbl_col_int
237   \tl_gset:Nn \g__tbl_span_tl {1}
238 }

```

(End of definition for \tbl_init_cell_data_for_table:.)

\tbl_update_cell_data_for_next_row:

```

239 \cs_new_protected:Npn \tbl_update_cell_data_for_next_row: {
240   \int_gincr:N \g__tbl_row_int           % this row about to start
241   \int_gzero:N \g__tbl_col_int          % we are before first col
242 }

```

(End of definition for \tbl_update_cell_data_for_next_row:.)

`\tbl_init_cell_data_for_row:` If we start processing a cell in the first column we set `\g__tbl_col_int` to 1 as we are no longer "at" but "in" the first column. We also set `\g__tbl_span_tl` to its default value (not spanning cells).

```
243 \cs_new_protected:Npn \tbl_init_cell_data_for_row: {
244     \int_gset:Nn \g__tbl_col_int {1}
245     \tl_gset:Nn \g__tbl_span_tl {1}
246 }
```

(End of definition for \tbl_init_cell_data_for_row:.)

`\tbl_if_row_was_started:T` We use `\g__tbl_col_int` equal zero to indicate that we are just after a TR (i.e. between rows or at the very beginning of the table). Using the row count is not so good as longtable may split the table in chunks.

These conditionals have to be expandable (i.e., unprotected) as they are sometimes executed when \TeX is scanning inside a table.

```
247 \cs_new:Npn \tbl_if_row_was_started:T {
248     \int_compare:nNnT \g__tbl_col_int > 0
249 }
250 \cs_new:Npn \tbl_if_row_was_started:TF {
251     \int_compare:nNnTF \g__tbl_col_int > 0
252 }
```

(End of definition for \tbl_if_row_was_started:T and \tbl_if_row_was_started:TF.)

`\tbl_gzero_row_count:` This here is basically a temporary interface. What it will be in the end depends on what we decide concerning exposing row and column counters, if they stay internal we need something like this here (perhaps using `gincr` etc, or perhaps some other names in the first place).

```
253 \cs_new_protected:Npn \tbl_gzero_row_count: {
254     \int_gzero:N \g__tbl_row_int
255 }
256 \cs_new_protected:Npn \tbl_gincr_row_count: {
257     \int_gincr:N \g__tbl_row_int
258 }
259 \cs_new_protected:Npn \tbl_gdecr_row_count: {
260     \int_gdecr:N \g__tbl_row_int
261 }
```

(End of definition for \tbl_gzero_row_count:, \tbl_gincr_row_count:, and \tbl_gdecr_row_count:.)

`\tbl_inbetween_rows:` Again name is not really brilliant so far.

```
262 \cs_new_protected:Npn \tbl_inbetween_rows: {
263     \int_gzero:N \g__tbl_col_int
264 }
```

(End of definition for \tbl_inbetween_rows:.)

`\tbl_restore_outer_cell_data:`

```
265 \cs_new_protected:Npn \tbl_restore_outer_cell_data: {
266     \int_gset:Nn \g__tbl_col_int { \l__tbl_saved_col_tl }
267     \int_gset:Nn \g__tbl_row_int { \l__tbl_saved_row_tl }
268     \tl_gset_eq:NN \g__tbl_span_tl \l__tbl_saved_span_tl
269     \tl_gset_eq:NN \g__tbl_table_cols_tl \l__tbl_saved_table_cols_tl
270     \UseTaggingSocket{tbl/restore/celldata}
```

```

271 \tbl_restore_outer_cell_data:n { ==>~ restored~cell~data:~
272     \int_use:N \g__tbl_row_int,
273     \int_use:N \g__tbl_col_int,
274     \l__tbl_saved_span_tl \space
275     (
276     \int_compare:nNnTF \g__tbl_table_cols_tl = 0
277         { outer~ level }
278         { max:~ \g__tbl_table_cols_tl }
279     )
280     }
281 }

```

(End of definition for `\tbl_restore_outer_cell_data:.`)

`\tbl_update_multicolumn_cell_data:n` This macro updates `\g__tbl_col_int` and `\g__tbl_span_tl` inside a `\multicolumn` and possibly calls the tagging socket `tbl/row/begin`.

```

282 \cs_new_protected:Npn \tbl_update_multicolumn_cell_data:n #1 {

```

We execute socket for tagging only if this `\multicolumn` replaces the preamble of the first column. In that case we also have to set `\g__tbl_col_int` to 1 because this is no longer done in the preamble for the cell either.

```

283     \int_compare:nNnTF \g__tbl_col_int = 0
284     {
285         \UseTaggingSocket{tbl/row/begin}
286         \int_gset:Nn \g__tbl_col_int {1}
287     }

```

If we are in a later column we use `\g__tbl_span_tl` from the previous column to update.

```

288     {
289         \int_gadd:Nn \g__tbl_col_int { \g__tbl_span_tl }
290     }

```

Then we set the span value so that it can be use in the next column.

```

291     \tl_gset:Nn \g__tbl_span_tl {#1}
292 }

```

(End of definition for `\tbl_update_multicolumn_cell_data:n.`)

`\tbl_crcr:n` This macro is used instead of the usual `\crcr` at the end of a table. It is deliberately defined without protection because it may get expanded by the scanning mechanism of low-level T_EX after a final `\cr` (aka `\`) in the table. In that case it shouldn't stop the expansion and the conditional inside will be false, thus it just vanishes without doing anything. If there are missing cells (in which case we also haven't see `\cr` yet) the macro `\tbl_count_missing_cells:n` is executed and then the row is finished with a final `\cr`.

```

293 \cs_new:Npn \tbl_crcr:n #1 {
294     \int_compare:nNnT \g__tbl_col_int > 0
295     {
296         \tbl_count_missing_cells:n {#1}
297     }

```

Even if we are at the start of a row we may have to do a `\cr`, so we do a `\crcr` always at the end.

```

298     \crcr
299 }

```

(End of definition for \tbl_crcr:n.)

```
300 \ExplSyntaxOff
301 <@@=>
```

This is needed for `longtable` because `\refstepcounter` is setting up a target when `hyperref` is loaded and we don't want that in `longtable`. Prevent `longtable` patching by `hyperref` until `hyperref` does so automatically:

```
302 \def\hyper@nopatch@longtable{}
    Should there be a module?
303 <latexrelease>\NewModuleRelease{2024/06/01}{ltagging}
304 <latexrelease>          {Tagging support}
305 <latexrelease>\IncludeInRelease{0000/00/00}{ltagging}%
306 <latexrelease>          {Undo tagging support}
307 <latexrelease>
308 <latexrelease>
309 <latexrelease>
310 <latexrelease>\EndModuleRelease
311 </2ekernel | latexrelease>
```

File 56

lthyphen.dtx

This file contains the code for loading hyphenation patterns into L^AT_EX. Most of this will end up in a file called `hyphen.ltx`. If you wish to customize your L^AT_EX system in respect of hyphenation patterns, write a file `hyphen.cfg`. If this file exists, it will be loaded instead of `hyphen.ltx`. See the comments below for additional information.

To produce the printed version of this file the following code is used. It can be extracted with the DOCSTRIP program, or one can run this file directly through L^AT_EX 2_ε.

```
1 <*driver>
2 \documentclass{ltxdoc}
3 \begin{document}
4 \DocInput{lthyphen.dtx}
5 \end{document}
6 </driver>
```

The default file `hyphen.ltx` loads hyphenation patterns for US english. If you want to load additional or other hyphenation patterns, you should create a file `hyphen.cfg`. This is best done by starting from `hyphen.ltx`.

For backward compatibility, the default file, `hyphen.ltx`, first tries to load the file `hyphen.tex`. If this file exists, an information message is issued and the appropriate defaults for T_EX's internal parameters are set: `\language` is initialized to 0, and `\lefthyphenmin` and `\righthyphenmin` to 2 and 3, respectively, to disallow x- or -xx breaks.

```
7 <*default>
8 \InputIfFileExists{hyphen.tex}%
9   {\message{Loading hyphenation patterns for US english.}%
10    \language=0
11    \lefthyphenmin=2 \righthyphenmin=3 }%
```

Otherwise, since we cannot do anything without any hyphenation patterns, an error message is printed and the IniT_EX run is terminated by invoking `\@@end` (which is the L^AT_EX 2_ε name for T_EX's `\end` primitive).

```
12   {\errhelp{The configuration for hyphenation is incorrectly
13             installed.^^J}%
14             If you don't understand this error message you need
15             to seek^^Jexpert advice.}%
16   \errmessage{OOPS! I can't find any hyphenation patterns for
17              US english.^^J \space Think of getting some or the
18              latex2e setup will never succeed}\@@end}
19 </default>
```

The following example describes the possible contents of a file `hyphen.cfg` that will load both US English and German hyphenation patterns, making the former the default. It sets `\language` to 0 for the US patterns and to 1 for the German patterns. Then `\language` is set to 0 to make this the default and the default values of `\lefthyphenmin` and `\righthyphenmin` are set.

```
language=0
input hyphen % (or \input ushyphen1 if the file has been renamed)
language=1
input ghyph31
```

```
language=0
lefthyphenmin=2
righthyphenmin=3
endinput
```

Another possibility is to use the package `babel`, by Johannes Braams. That package is distributed with a suitable `hyphen.cfg` file.

File 57

ltxfinal.dtx

1 Final settings

This section contains the final settings for L^AT_EX. It initializes some debugging and typesetting parameters, sets the default `\catcodes` and `uc/lc` codes, and inputs the hyphenation file.

1.1 Debugging

By default, L^AT_EX shows statistics:

```
1 <*2ekernel>
2 \tracingstats1
```

1.2 Typesetting parameters

```
\@lowpenalty These are penalties used internally.
\@medpenalty 3 \newcount\@lowpenalty
\@highpenalty 4 \newcount\@medpenalty
5 \newcount\@highpenalty
```

(End of definition for \@lowpenalty, \@medpenalty, and \@highpenalty.)

```
\newmarks Allocate extended marks types if etex is active. Placed here at the end of the format to
increase compatibility with count allocations in earlier releases.
```

```
6 </2ekernel>
7 <*2ekernel | latexrelease>
8 <latexrelease>\IncludeInRelease{2015/01/01}%
9 <latexrelease>           {\newmarks}{Extended Allocation}%
10 \ifx\marks\@undefined\else
11 \def\newmarks{%
12   \e@alloc\marks \e@alloc@chardef{\count256}\m@ne\e@alloc@top}
13 \fi
14 </2ekernel | latexrelease>
15 <latexrelease>\EndIncludeInRelease
16 <latexrelease>\IncludeInRelease{0000/00/00}%
17 <latexrelease>           {\newmarks}{Extended Allocation}%
18 <latexrelease>\let\newmarks\@undefined
19 <latexrelease>\EndIncludeInRelease
20 <*2ekernel>
```

(End of definition for \newmarks.)

Allocate 3 mark classes to be used in `\markboth` and `\markright`. Should be done earlier but for that definition of `\newmarks` needs moving (which it should I guess).

```
21 </2ekernel>
22 <*2ekernel | latexrelease>
23 <latexrelease>\IncludeInRelease{2022/06/01}%
24 <latexrelease>           {2e-left}{Delayed legacy marks}%
25 \NewMarkClass {2e-left}
```

```

26 \NewMarkClass {2e-right}
27 \NewMarkClass {2e-right-nonempty}

```

No rollback really, the marks will remain.

```

28 </2ekernel | latexrelease>
29 <latexrelease>\EndIncludeInRelease
30 <latexrelease>\IncludeInRelease{0000/00/00}%
31 <latexrelease>          {2e-left}{Delayed legacy marks}%
32 <latexrelease>
33 <latexrelease>\EndIncludeInRelease
34 <*2ekernel>

```

```

\newXeTeXintercharclass Allocate \XeTeXintercharclass types if xetex is active. previously defined in xetex.ini.
\xe@alloc@intercharclass
\xe@alloc@intercharclass@top
35 </2ekernel>
36 <*2ekernel | latexrelease>
37 <latexrelease>\IncludeInRelease{2015/01/01}%
38 <latexrelease>          {\newXeTeXintercharclass}{Extended Allocation}%

```

Classes allocated 1 to 4094 (or 254 on older xetex) (In earlier XeLaTeX versions 1, 2 and 3 were pre-set for CJK).

```

39 \ifx\XeTeXcharclass\@undefined
40 \else
41 \ifdim\the\XeTeXversion\XeTeXrevision\p@>0.99993\p@
42 \chardef\xe@alloc@intercharclass@top=4095
43 \else
44 \chardef\xe@alloc@intercharclass@top=255
45 \fi
46 \def\newXeTeXintercharclass{%
47 \e@alloc\XeTeXcharclass
48 \chardef\xe@alloc@intercharclass\m@ne\xe@alloc@intercharclass@top}
49 \fi
50 </2ekernel | latexrelease>
51 <latexrelease>\EndIncludeInRelease
52 <latexrelease>\IncludeInRelease{0000/00/00}%
53 <latexrelease>          {\newXeTeXintercharclass}{Extended Allocation}%
54 <latexrelease> \ifx\XeTeXcharclass\@undefined
55 <latexrelease> \else
56 <latexrelease> \def\xe@alloc@#1#2#3#4#5{\global\advance#1\@ne
57 <latexrelease> \xe@ch@ck#1#4#2%
58 <latexrelease> \allocationnumber#1%
59 <latexrelease> \global#3#5\allocationnumber
60 <latexrelease> \wlog{\string#5=\string#2\the\allocationnumber}}
61 <latexrelease> \def\xe@ch@ck#1#2#3{%
62 <latexrelease> \ifnum#1<#2\else
63 <latexrelease> \errmessage{No room for a new #3}%
64 <latexrelease> \fi}
65 <latexrelease> \def\newXeTeXintercharclass{%
66 <latexrelease> \xe@alloc@\xe@alloc@intercharclass
67 <latexrelease> \XeTeXcharclass\chardef\@cclv}
68 <latexrelease> \fi
69 <latexrelease>\EndIncludeInRelease
70 <*2ekernel | latexrelease>
71 <latexrelease>\IncludeInRelease{2016/02/01}%

```

```

72 <latexrelease> {\xe@alloc@intercharclass}{Start of XeTeX class allocator}%
73 \ifx\XeTeXcharclass\@undefined
74 \else
75   \countdef\xe@alloc@intercharclass=257
76   \xe@alloc@intercharclass=\z@
77 \fi
78 </2ekernel | latexrelease>
79 <latexrelease>\EndIncludeInRelease
80 <latexrelease>\IncludeInRelease{2015/01/01}%
81 <latexrelease> {\xe@alloc@intercharclass}{Start of XeTeX class allocator}%
82 <latexrelease> \ifx\XeTeXcharclass\@undefined
83 <latexrelease> \else
84 <latexrelease>   \xe@alloc@intercharclass=\thr@@
85 <latexrelease> \fi
86 <latexrelease>\EndIncludeInRelease
87 <latexrelease>\IncludeInRelease{0000/00/00}%
88 <latexrelease> {\xe@alloc@intercharclass}{Start of XeTeX class allocator}%
89 <latexrelease> \ifx\XeTeXcharclass\@undefined
90 <latexrelease> \else
91 <latexrelease>   \newcount\xe@alloc@intercharclass
92 <latexrelease>   \xe@alloc@intercharclass=\thr@@
93 <latexrelease> \fi
94 <latexrelease>\EndIncludeInRelease
95 <*2ekernel>

```

(End of definition for \newXeTeXintercharclass, \xe@alloc@intercharclass, and \xe@alloc@intercharclass@top.)

`trace_stack_levels` Now define the Lua function to emulate `\tracingstacklevels` and install it in the `input_level_string` callback.

```

96 </2ekernel>
97 <*2ekernel | latexrelease>

```

In `latexrelease` mode we always remove the function from the callback, then add the correct version later.

```

98 <latexrelease>\ifx\directlua\@undefined
99 <latexrelease>\else
100 <latexrelease> \directlua{%
101 <latexrelease>   if luatexbase.callbacktypes['input_level_string'] and %
102 <latexrelease>     luatexbase.in_callback('input_level_string', 'tracingstacklevels') then
103 <latexrelease>     luatexbase.remove_from_callback('input_level_string', 'tracingstacklevels')
104 <latexrelease>   end}%
105 <latexrelease>\fi
106 <latexrelease>\IncludeInRelease{2021/06/01}{trace_stack_levels}%
107 <latexrelease>           {Lua trace_stack_levels function}%
108 \ifx\directlua\@undefined
109 \else
110 <*2ekernel>
111   \expanded{%
112     \everyjob{\the\everyjob
113       \noexpand%\directlua
114 </2ekernel>
115     \directlua{%
116       local function trace_stack_levels (input_ptr)
117         local tracingstacklevels = tex.count.tracingstacklevels

```



```

118     if tex.tracingmacros > 0 or input_ptr < tracingstacklevels then
119         if tracingstacklevels > 0 then
120             if input_ptr < tracingstacklevels then
121                 return "\string\n\string~" .. string.rep(".",input_ptr)
122             else
123                 return "\string~\string~"
124             end
125         else
126             return "\string\n"
127         end
128     else
129         return ""
130     end
131 end
132 <latexrelease>   if luatexbase.callbacktypes['input_level_string'] then
133     luatexbase.add_to_callback('input_level_string',
134         trace_stack_levels,'tracingstacklevels')
135 <latexrelease>   end
136 }%
137 <*2ekernel>
138 }}%
139 </2ekernel>
140 \fi
141 <latexrelease>\EndIncludeInRelease
142 <latexrelease>

```

Then for the full rollback, just do nothing, since the function was already taken out of the rollback above.

```

143 <latexrelease>\IncludeInRelease{0000/00/00}{trace_stack_levels}%
144 <latexrelease>           {Lua trace_stack_levels function}%
145 <latexrelease>% Nothing here
146 <latexrelease>\EndIncludeInRelease
147 </2ekernel | latexrelease>
148 <*2ekernel>

```

(End of definition for trace_stack_levels.)

The default values of the picture and \fbox parameters:

```

149 \unitlength = 1pt
150 \fboxsep = 3pt
151 \fboxrule = .4pt

```

The saved value of T_EX's \maxdepth:

```

152 \@maxdepth = \maxdepth

```

\vsize initialized because a \clearpage with \vsize < \topskip causes trouble. \@colroom and \@colht also initialized because \vsize may be set to them if a \clearpage is done before the \begin{document}

```

153 \vsize = 1000pt
154 \@colroom = \vsize
155 \@colht = \vsize

```

Initialise \textheight \textwidth and page style, to avoid internal errors if they are not set by the class.

```

156 \textheight=.5\maxdimen
157 \textwidth=\textheight
158 \ps@empty

```

1.3 Lccodes for hyphenation

For 7- and 8-bit engines the assumption of T1 encodings is the basis for the hyphenation patterns. That's not the case for the Unicode engines, where the assumption is engine-native working. The common loader system provides access to data from the Unicode Consortium covering not only `\lccode` but also other related data. The `\lccode` part of that at least needs to be loaded before hyphenation is tackled: XeTeX follows the standard TeX route of building patterns into the format. LuaTeX doesn't require this data be loaded *here* but it does need to be loaded somewhere. Rather than test for the Unicode engines by name, the approach here is to look for the extended math mode handling both provide: any other engine developed in this area will presumably also provide `\Umathcode`.

```
159 \ifnum 0%
160   \ifx\Umathcode\@undefined\else 1\fi
161   \ifx\XeTeXmathcode\@undefined\else 1\fi
162   >\z@
163   \message{ Unicode character data,}
164   \input{load-unicode-data}
165 \endkernel)
166 \langle\latexrelease\rangle\IncludeInRelease{2016/02/01}%
167 \langle\latexrelease\rangle {\XeTeXintercharclasses}{XeTeX character classes}%
168 \langle\latexrelease\rangle \ifx\XeTeXinterchartoks\undefined
169 \langle\latexrelease\rangle \else
170 \langle\latexrelease\rangle \begingroup
171 \langle\latexrelease\rangle \chardef\XeTeXcharclassID = 0 %
172 \langle\latexrelease\rangle \chardef\XeTeXcharclassOP = 0 %
173 \langle\latexrelease\rangle \chardef\XeTeXcharclassCL = 0 %
174 \langle\latexrelease\rangle \chardef\XeTeXcharclassEX = 0 %
175 \langle\latexrelease\rangle \chardef\XeTeXcharclassIS = 0 %
176 \langle\latexrelease\rangle \chardef\XeTeXcharclassNS = 0 %
177 \langle\latexrelease\rangle \chardef\XeTeXcharclassCM = 0 %
178 \langle\latexrelease\rangle \input{load-unicode-xetex-classes}
179 \langle\latexrelease\rangle \endgroup
180 \langle\latexrelease\rangle \global\let\xtxHanGlue\undefined
181 \langle\latexrelease\rangle \global\let\xtxHanSpace\undefined
182 \langle\latexrelease\rangle \global\XeTeXinterchartoks 0 1 = {}
183 \langle\latexrelease\rangle \global\XeTeXinterchartoks 0 2 = {}
184 \langle\latexrelease\rangle \global\XeTeXinterchartoks 0 3 = {}
185 \langle\latexrelease\rangle \global\XeTeXinterchartoks 1 0 = {}
186 \langle\latexrelease\rangle \global\XeTeXinterchartoks 2 0 = {}
187 \langle\latexrelease\rangle \global\XeTeXinterchartoks 3 0 = {}
188 \langle\latexrelease\rangle \global\XeTeXinterchartoks 1 1 = {}
189 \langle\latexrelease\rangle \global\XeTeXinterchartoks 1 2 = {}
190 \langle\latexrelease\rangle \global\XeTeXinterchartoks 1 3 = {}
191 \langle\latexrelease\rangle \global\XeTeXinterchartoks 2 1 = {}
192 \langle\latexrelease\rangle \global\XeTeXinterchartoks 2 2 = {}
193 \langle\latexrelease\rangle \global\XeTeXinterchartoks 2 3 = {}
194 \langle\latexrelease\rangle \global\XeTeXinterchartoks 3 1 = {}
195 \langle\latexrelease\rangle \global\XeTeXinterchartoks 3 2 = {}
196 \langle\latexrelease\rangle \global\XeTeXinterchartoks 3 3 = {}
197 \langle\latexrelease\rangle \fi
198 \langle\latexrelease\rangle\EndIncludeInRelease
199 \langle\latexrelease\rangle\IncludeInRelease{0000/00/00}%
```

```

200 <latexrelease> {\XeTeXintercharclasses}{XeTeX character classes}%
201 <latexrelease> \ifxXeTeXinterchartoks\undefined
202 <latexrelease> \else
203 <latexrelease> \input{load-unicode-xetex-classes}
204 <latexrelease> \gdef\xtxHanGlue{\hskip0pt plus 0.1em\relax}
205 <latexrelease> \gdef\xtxHanSpace{\hskip0.2em plus 0.2em minus 0.1em\relax}
206 <latexrelease> \global\XeTeXinterchartoks 0 1 = {\xtxHanSpace}
207 <latexrelease> \global\XeTeXinterchartoks 0 2 = {\xtxHanSpace}
208 <latexrelease> \global\XeTeXinterchartoks 0 3 = {\nobreak\xtxHanSpace}
209 <latexrelease> \global\XeTeXinterchartoks 1 0 = {\xtxHanSpace}
210 <latexrelease> \global\XeTeXinterchartoks 2 0 = {\nobreak\xtxHanSpace}
211 <latexrelease> \global\XeTeXinterchartoks 3 0 = {\xtxHanSpace}
212 <latexrelease> \global\XeTeXinterchartoks 1 1 = {\xtxHanGlue}
213 <latexrelease> \global\XeTeXinterchartoks 1 2 = {\xtxHanGlue}
214 <latexrelease> \global\XeTeXinterchartoks 1 3 = {\nobreak\xtxHanGlue}
215 <latexrelease> \global\XeTeXinterchartoks 2 1 = {\nobreak\xtxHanGlue}
216 <latexrelease> \global\XeTeXinterchartoks 2 2 = {\nobreak\xtxHanGlue}
217 <latexrelease> \global\XeTeXinterchartoks 2 3 = {\xtxHanGlue}
218 <latexrelease> \global\XeTeXinterchartoks 3 1 = {\xtxHanGlue}
219 <latexrelease> \global\XeTeXinterchartoks 3 2 = {\xtxHanGlue}
220 <latexrelease> \global\XeTeXinterchartoks 3 3 = {\nobreak\xtxHanGlue}
221 <latexrelease> \fi
222 <latexrelease>\EndIncludeInRelease
223 <*2ekernel>

```

There is one over-ride that makes sense here (see below for the same for 8-bit engines): setting the lccode for - to itself.

```
224 \lccode'\- ='\- % default hyphen char
```

The alternative is that a “traditional” engine is in use.

```
225 \else
```

We set things up so that hyphenation files can assume that the default (T1) lccodes are in use (at present this also sets up the uccodes). We temporarily define `\reserved@a` to apply `\reserved@c` to all the numbers in the range of its arguments.

```

226 \def\reserved@a#1#2{%
227   \@tempcnta#1\relax
228   \@tempcntb#2\relax
229   \reserved@b
230 }
231 \def\reserved@b{%
232   \ifnum\@tempcnta>\@tempcntb\else
233     \reserved@c\@tempcnta
234     \advance\@tempcnta\@ne
235     \expandafter\reserved@b
236   \fi
237 }

```

Depending on the T_EX version, we might not be allowed to do this for non-ASCII characters.

```

238 \def\reserved@c#1{%
239   \count@=#1\advance\count@ by -"20
240   \uccode#1=\count@
241   \lccode#1=#1
242 }

```

```

243 \reserved@a{'\a}{'\z}
244 \reserved@a{"A0}{"BC}
245 \reserved@a{"E0}{"FF}

```

The upper case characters need their `\uccode` and `\lccode` values set, and their `\sfcode` set to 999.

```

246 \def\reserved@c#1{%
247   \count@=#1\advance\count@ by "20
248   \uccode#1=#1
249   \lccode#1=\count@
250   \sfcode#1=999
251 }
252 \reserved@a{'\A}{'\Z}
253 \reserved@a{"80}{"9C}
254 \reserved@a{"C0}{"DF}

```

Well, it would be nice if that were correct, but unfortunately, the Cork encoding contains some odd slots whose `uccode` or `lccode` isn't quite what you'd expect.

```

255 \uccode'\^^Y='\I    % dotless i
256 \lccode'\^^Y='\^^Y % dotless i
257 \uccode'\^^Z='\J    % dotless j, ae in OT1
258 \lccode'\^^Z='\^^Z % dotless j, ae in OT1
259 \lccode'\^^9d='\i   % dotted I
260 \uccode'\^^9d='\^^9d % dotted I
261 \lccode'\^^9e='\^^9e % d-bar
262 \uccode'\^^9e='\^^d0 % d-bar

```

Finally here is one that helps hyphenation in the OT1 encoding.

```

263 \lccode'\^^[='\^^[ % oe in OT1

```

And we also set the `\lccode` of `\-` and `\textcompwordmark` so that they do not prevent hyphenation in the remainder of the word (as suggested by Lars Helström).

```

264 \lccode'\- ='\- % default hyphen char
265 \lccode 127=127 % alternate hyphen char
266 \lccode 23 =23 % textcompwordmark in T1

```

End of the conditional to select either Unicode or T1 encoding defaults.

```

267 \fi

```

At this stage, we can install any last-minute `expl3` set-up.

```

268 \@expl@finalise@setup@@
269 \def\@expl@finalise@setup@@{}

```

This is as good a place as any to active a few XeTeX-specific settings

```

270 \ifx\XeTeXuseglyphmetrics\@undefined
271 \else
272   \XeTeXuseglyphmetrics=1 %
273   \XeTeXdashbreakstate=1 %
274 \fi

```

1.4 Hyphenation

The following code will be compiled into the format file. It checks for the existence of `hyphen.cfg` in inputs that file if found. Otherwise it inputs `hyphen.ltx`. Note that these are loaded in *before* the `\catcodes` are set, so local hyphenation files can use 8-bit input.

We try to load the customized hyphenation description file.

```

275 \InputIfFileExists{hyphen.cfg}
276     {\typeout{=====^J%
277             Local configuration file hyphen.cfg used^^J%
278             =====}%
279     \def\@addtofilelist##1{\xdef\@filelist{\@filelist,##1}}%
280     }
281     {\input{hyphen.ltx}}
282 \let\@addtofilelist\@gobble

```

`\l@nohyphenation`

```

283 \ifx\l@nohyphenation \@undefined
284 \newlanguage\l@nohyphenation
285 \fi

```

(End of definition for \l@nohyphenation.)

`\document@default@language` Default document language. -1 acts as language 0, but used as a flag in `\document` to see if it has been set in the preamble.

```

286 </2ekernel>
287 <*2ekernel | latexrelease>
288 <latexrelease>\IncludeInRelease{2017/04/15}%
289 <latexrelease>    {\document@default@language}{Save language for hyphenation}%
290 \let\document@default@language\m@ne
291 </2ekernel | latexrelease>
292 <latexrelease>\EndIncludeInRelease
293 <latexrelease>\IncludeInRelease{0000/00/00}%
294 <latexrelease>    {\document@default@language}{Save language for hyphenation}%
295 %
296 <latexrelease>\let\document@default@language\@undefined
297 <latexrelease>\EndIncludeInRelease
298 <*2ekernel>

```

(End of definition for \document@default@language.)

1.5 Font loading

Fonts loaded during the formatting process might already have changed the `\font@submax` from `Opt` to something higher. If so, we put out a bold warning.

```

299 \ifdim \font@submax >\z@
300     \@font@warning{Size substitutions with differences\MessageBreak
301                 up to \font@submax\space have occurred.\MessageBreak
302                 \MessageBreak
303                 Please check the transcript file
304                 carefully\MessageBreak
305                 and redo the format generation if necessary!
306                 \@gobbletwo}%
307     \errhelp{Only stopped, to give you time to
308             read the above message.}
309     \errmessage{}

```

We reset the macro. Otherwise every user will get a warning on every job.

```

310 \def\font@submax{Opt}
311 \fi

```

For pdfT_EX preload and enable automatic glyph to Unicode mapping for more reliable copy and paste support.

```

312 </2ekernel>
313 <*2ekernel | latexrelease>
314 <latexrelease>\IncludeInRelease{2021/06/01}%
315 <latexrelease>          {\pdfgentounicode}{Preload glyphtounicode}%
316 \ifx \pdfgentounicode \@undefined \else
317 <*2ekernel>
318   \ifnum 0=0%
319     \ifdefined\pdftexversion
320 % \pdftexversion<140 does not have \pdfgentounicode, so we only check higher values
321   \ifnum \pdftexversion=140 \ifnum\pdftexrevision<22 1\fi\fi
322   \fi
323   \relax
324 </2ekernel>
325   \input glyphtounicode
326 <*2ekernel>
327   \else
328     \begingroup
329       \everyeof{\noexpand}\endlinechar-1
330       \edef\x{\endgroup
331         \everyjob{\the\everyjob\@@input glyphtounicode }%
332       }\x
333   \fi
334 </2ekernel>
335   \pdfgentounicode=1
336 \fi
337 </2ekernel | latexrelease>
338 <latexrelease>\EndIncludeInRelease

```

When rolling back we can't unload the glyphtounicode mappings, but we can reset `\pdfgentounicode` to ensure that they aren't used.

```

339 <latexrelease>\IncludeInRelease{0000/00/00}%
340 <latexrelease>          {\pdfgentounicode}{Preload glyphtounicode}%
341 <latexrelease>\ifx \pdfgentounicode \@undefined \else
342 <latexrelease> \pdfgentounicode=0
343 <latexrelease>\fi
344 <latexrelease>\EndIncludeInRelease
345 <*2ekernel>

```

1.6 Input encoding

Starting with the 2018 L^AT_EX release default the inputencoding to UTF-8. Unless the format is being used with luatex, xetex, enctex or mltx.

This is done in a way largely compatible with older releases: `utf8.def` is input just as if

```
\usepackage[utf8]{inputenc}
```

had been used, however rather than input the whole package a minimal core part just enough to support loading the UTF-8 encoding files is defined here.

If a document re-specifies UTF-8 this is silently ignored.

```

346 </2ekernel>
347 <*2ekernel | latexrelease>

```

Check that a classic 8-bit tex engine is being used (LaTeX or PDFLaTeX).

```
348 <latexrelease>\IncludeInRelease{2018/04/01}%
349 <latexrelease>          {UTFviii@invalid}{UTF-8 default}%

Skip this section in Unicode TeX, or if MLTeX and EncTeX are enabled.

350 \ifnum0%
351   \ifx\Umathcode\@undefined\else 1\fi
352   \ifx\mubyte\@undefined\else 1\fi
353   \ifx\charsubdef\@undefined\else 1\fi
354   =\z@

355 \def\saved@space@catcode{10}
356 \let\@inpenc@test\relax
357 \def\IeC{%
358   \ifx\protect\@typeset@protect
359     \expandafter\@firstofone
360   \else
361     \noexpand\IeC
362   \fi
363 }
```

Make characters active for UTF-8 input formats

```
364 \@tempcnta=1
365 \loop
366   \catcode\@tempcnta=13 %
367   \advance\@tempcnta\@ne %
368   \ifnum\@tempcnta<32 %
369   \repeat %
370 \catcode0=15 % null
371 \catcode9=10 % tab
372 \catcode10=12 % ctrl J
373 \catcode12=13 % ctrl L
374 \catcode13=5 % newline
375 \@tempcnta=128
376 \loop
377   \catcode\@tempcnta=13
378   \advance\@tempcnta\@ne
379   \ifnum\@tempcnta<256
380   \repeat
```

\UseRawInputEncoding Reset 8 bit characters to catcode 12 so the input encoding matches the “Raw” font encoding. Useful for special behaviours, or for compatibility with older L^AT_EX formats.

```
381 \def\UseRawInputEncoding{%
382   \let\inputencodingname\@undefined % revert
383   \let\DeclareFontEncoding\DeclareFontEncoding@saved % revert
384   \let\DeclareUnicodeCharacter\@undefined % revert
385   \@tempcnta=1
386   \loop
387     \catcode\@tempcnta=15 %
388     \advance\@tempcnta\@ne %
389     \ifnum\@tempcnta<32 %
390     \repeat %
391     \catcode0=15 % null
392     \catcode9=10 % tab
```

```

393 \catcode10=12 % ctrl J
394 \catcode12=13 % ctrl L
395 \catcode13=5 % newline
396 \@tempcnta=128
397 \loop
398   \catcode\@tempcnta=12
399   \advance\@tempcnta\@ne
400 \ifnum\@tempcnta<256
401 \repeat
402 }

```

(End of definition for \UseRawInputEncoding.)

\DeclareFontEncoding@saved Saved version of \DeclareFontEncoding@ before utf8.def modifies it for use in \UseRawInputEncoding above.

```

403 \let\DeclareFontEncoding@saved\DeclareFontEncoding@

```

(End of definition for \DeclareFontEncoding@saved.)

```

404 \edef\inputencodingname{utf8}%
405 \input{utf8.def}
406 \let\UTFviii@undefined@err@@\UTFviii@undefined@err
407 \let\UTFviii@invalid@err@@\UTFviii@invalid@err
408 \let\UTFviii@two@octets@@\UTFviii@two@octets
409 \let\UTFviii@three@octets@@\UTFviii@three@octets
410 \let\UTFviii@four@octets@@\UTFviii@four@octets
411 <2kernel>\def\UTFviii@undefined@err#1{\@gobble#1}%
412 <2kernel>\let\UTFviii@invalid@err\string
413 <2kernel>\let\UTFviii@two@octets\string
414 <2kernel>\let\UTFviii@three@octets\string
415 <2kernel>\let\UTFviii@four@octets\string
416 <2kernel>\everyjob\expandafter{\the\everyjob
417 <2kernel>\let\UTFviii@undefined@err\UTFviii@undefined@err@@
418 <2kernel>\let\UTFviii@invalid@err\UTFviii@invalid@err@@
419 <2kernel>\let\UTFviii@two@octets\UTFviii@two@octets@@
420 <2kernel>\let\UTFviii@three@octets\UTFviii@three@octets@@
421 <2kernel>\let\UTFviii@four@octets\UTFviii@four@octets@@
422 <2kernel>}
423 \let\@inpenc@test\@undefined
424 \let\@saved@space@catcode\@undefined

```

For formats not set up for UTF-8 default, set the C0 controls to catcode 15.

```

425 \else
426 \@tempcnta=0
427 \loop
428   \catcode\@tempcnta=15 %
429   \advance\@tempcnta\@ne %
430   \ifnum\@tempcnta<32 %
431   \repeat %
432 \catcode0=15 % null
433 \catcode9=10 % tab
434 \catcode10=12 % ctrl J
435 \catcode12=13 % ctrl L
436 \catcode13=5 % newline
437 \let\UseRawInputEncoding\relax

```


This ends the skipped code in Unicode engines:

```

438 \fi
439 </2ekernel | latexrelease>
440 <latexrelease>\EndIncludeInRelease
441 <latexrelease>\IncludeInRelease{0000/00/00}%
442 <latexrelease>          {\UTFviii@invalid}{UTF-8 default}%

The first block of commands got only introduced in 2019 but we revert all of Unicode
support in one go not jump to the intermediate version.

443 <latexrelease> \let\UTFviii@two@octets@combine\@undefined
444 <latexrelease> \let\UTFviii@three@octets@combine\@undefined
445 <latexrelease> \let\UTFviii@four@octets@combine\@undefined
446 <latexrelease> \let\UTFviii@two@octets@string\@undefined
447 <latexrelease> \let\UTFviii@three@octets@string\@undefined
448 <latexrelease> \let\UTFviii@four@octets@string\@undefined
449 <latexrelease> \let\UTFviii@two@octets@noexpand\@undefined
450 <latexrelease> \let\UTFviii@three@octets@noexpand\@undefined
451 <latexrelease> \let\UTFviii@four@octets@noexpand\@undefined

452 <latexrelease>\@tempcnta=0
453 <latexrelease>\loop
454 <latexrelease> \catcode\@tempcnta=15
455 <latexrelease> \advance\@tempcnta\@ne
456 <latexrelease>\ifnum\@tempcnta<32
457 <latexrelease>\repeat      %
458 <latexrelease>\catcode9=10 % tab
459 <latexrelease>\catcode10=12 % ctrl J
460 <latexrelease>\catcode12=13 % ctrl L
461 <latexrelease>\catcode13=5 % newline
462 <latexrelease>\@tempcnta=128
463 <latexrelease>\loop
464 <latexrelease>\catcode\@tempcnta=12
465 <latexrelease>\advance\@tempcnta\@ne
466 <latexrelease>\ifnum\@tempcnta<256
467 <latexrelease>\repeat
468 <latexrelease>\let\IeC\@undefined
469 <latexrelease>\def\DeclareFontEncoding@#1#2#3{%
470 <latexrelease> \expandafter
471 <latexrelease> \ifx\csname T@#1\endcsname\relax
472 <latexrelease> \def\cdp@elt{\noexpand\cdp@elt}%
473 <latexrelease> \xdef\cdp@list{\cdp@list\cdp@elt{#1}%
474 <latexrelease>          {\default@family}{\default@series}%
475 <latexrelease>          {\default@shape}}%
476 <latexrelease> \expandafter\let\csname#1-cmd\endcsname\@changed@cmd
477 <latexrelease> \else
478 <latexrelease> \font@info{Redeclaring font encoding #1}%
479 <latexrelease> \fi
480 <latexrelease> \global\@namedef{T@#1}{#2}%
481 <latexrelease> \global\@namedef{M@#1}{\default@M#3}%
482 <latexrelease> \xdef\LastDeclaredEncoding{#1}%
483 <latexrelease> }
484 <latexrelease> \let\UseRawInputEncoding\@undefined
485 <latexrelease> \let\DeclareFontEncoding@saved\@undefined
486 <latexrelease> \let\inputencodingname\@undefined
487 <latexrelease>\EndIncludeInRelease

```

488 (*2kernel)

We temporarily define `\reserved@a` to apply `\reserved@c` to all the numbers in the range of its arguments.

```
489 \def\reserved@a#1#2{%
490   \@tempcnta#1\relax
491   \@tempcntb#2\relax
492   \reserved@b
493 }
494 \def\reserved@b{%
495   \ifnum\@tempcnta>\@tempcntb\else
496     \reserved@c\@tempcnta
497     \advance\@tempcnta\@ne
498     \expandafter\reserved@b
499   \fi
500 }
```

Set the special catcodes (although some of these are useless, since an error will have occurred if the catcodes have changed). Note that `^^J` has catcode ‘other’ for use in warning messages.

```
501 \catcode'\ =10
502 \catcode'\#=6
503 \catcode'\$=3
504 \catcode'\%=14
505 \catcode'\&=4
506 \catcode'\'=0
507 \catcode'\^=7
508 \catcode'\_ =8
509 \catcode'\{=1
510 \catcode'\}=2
511 \catcode'\-=13
512 \catcode'\@=11
513 \catcode'\^^I=10
514 \catcode'\^^J=12
515 \catcode'\^^L=13
516 \catcode'\^^M=5
```

Set the ‘other’ catcodes.

```
517 \def\reserved@c#1{\catcode#1=12\relax}
518 \reserved@c{'\!}
519 \reserved@c{'\"}
520 \reserved@a{'\'}{'\'?}
521 \reserved@c{'\[]}
522 \reserved@c{'\]}
523 \reserved@c{'\''}
524 \reserved@c{'\|}
```

Set the ‘letter’ catcodes.

```
525 \def\reserved@c#1{\catcode#1=11\relax}
526 \reserved@a{'\A}{'\Z}
527 \reserved@a{'\a}{'\z}
```

All the characters in the range 0–31 and 127–255 are illegal, *except* `tab` (`^^I`), `nl` (`^^J`), `ff` (`^^L`) and `cr` (`^^M`).

1.7 Lccodes and uccodes

We now again set up the default (T1) uc/lccodes. The lower case characters need their `\uccode` and `\lccode` values set. Some of this is a repeat of the set-up before loading hyphenation files. Depending on the T_EX version, we might not be allowed to do this for non-ASCII characters. For the Unicode engines (XeT_EX and LuaT_EX) there is no need to do any of this: they use hyphenation data which does not alter any of the set up and so this entire block is skipped.

```
528 \ifnum 0%
529   \ifx\Umathcode\@undefined\else 1\fi
530   \ifx\XeTeXmathcode\@undefined\else 1\fi
531   >\z@
532 \else
533 \def\reserved@c#1{%
534   \count@=#1\advance\count@ by -"20
535   \uccode#1=\count@
536   \lccode#1=#1
537 }
538 \reserved@a{\a}{\z}
539 \reserved@a{"A0}{"BC}
540 \reserved@a{"E0}{"FF}
```

The upper case characters need their `\uccode` and `\lccode` values set, and their `\sfcode` set to 999.

```
541 \def\reserved@c#1{%
542   \count@=#1\advance\count@ by "20
543   \uccode#1=#1
544   \lccode#1=\count@
545   \sfcode#1=999
546 }
547 \reserved@a{\A}{\Z}
548 \reserved@a{"80}{"9C}
549 \reserved@a{"C0}{"DF}
```

Well, it would be nice if that were correct, but unfortunately, the Cork encoding contains some odd slots whose uccode or lccode isn't quite what you'd expect.

```
550 \uccode'\^^Y=' \I    % dotless i
551 \lccode'\^^Y=' \^^Y  % dotless i
552 \uccode'\^^Z=' \J    % dotless j, ae in OT1
553 \lccode'\^^Z=' \^^Z  % dotless j, ae in OT1
554 \lccode'\^^9d=' \i   % dotted I
555 \uccode'\^^9d=' \^^9d % dotted I
556 \lccode'\^^9e=' \^^9e % d-bar
557 \uccode'\^^9e=' \^^d0 % d-bar
```

Finally here is one that helps hyphenation in the OT1 encoding.

```
558 \lccode'\^^[=' \^^[  % oe in OT1
559 \fi % End of reset block for 8-bit engines
```

`\BCPdata` A stub for use by babel, polyglossia, etc.

```
560 \ExplSyntaxOn
561 \newcommand*\BCPdata[1]{
562   \str_case:nn {#1}
563   {
564     { language } { en }
565   }
566 \ExplSyntaxOff
```

```

565     { region } { US }
566     { script } { Latn }
567     { tag }    { en-US }
568   }
569 }
570 \ExplSyntaxOff

```

(End of definition for \BCPdata.)

1.8 Case changing

`\MakeUppercase` And whilst we're doing things with `uc/lc` tables, here are two commands to upper- and lower-case a string.

`\MakeLowercase` Wrappers around the L3 case changing functions. `\protected` to make them mostly safe as replacements for `\uppercase` and `\lowercase`.

`\MakeTitlecase` In

`\NoCaseChange` `\CaseSwitch`

`\AddToNoCaseChangeList` `\markboth{\MakeUppercase\contentsname}`

`\CaseSwitch` `{\MakeUppercase\contentsname}`

`\DeclareCaseChangeEquivalent`

`\DeclareLowercaseMapping`

`\DeclareTitlecaseMapping`

`\DeclareUppercaseMapping`

`\@uclclist`

then the uppercasing is only done to the first letter of the contents name, since the `mark` expands out to:

```

\mark{\MakeUppercase Table of Contents}
{\MakeUppercase Table of Contents}

```

In order to get round this, we redefine `\MakeUppercase` and `\MakeLowercase` to grab their argument and brace it.

Earlier versions needed to process `\@uclclist` in an `\edef` to handle legacy input encodings, but recent (2022) `expl3` versions handle non-UTF8 text natively so we simply call the `\text_...case:n` functions.

```

571 \ExplSyntaxOn
572 \keys_define:nn { __kernel }
573   {
574     lang .str_set:N = \reserved@a ,
575     locale .str_set:N = \reserved@a ,
576     words .choices:nn =
577       { all , first }
578       { \str_set:Nn \reserved@b {#1} }
579   }
580 \cs_new_protected:Npn \@@text@case@aux #1#2#3
581   {
582     \cs_set_nopar:Npn \reserved@a { }
583     \cs_set_nopar:Npn \reserved@b { }
584     \tl_if_blank:nTF {#2}
585       {
586         \str_set:Nx \reserved@a
587           { \BCPdata { casing } }
588         \str_if_empty:NT \reserved@a
589           {
590             \str_set:Nx \reserved@a
591               { \BCPdata { language } }
592           }
593       }

```

```

594     { \keys_set:nm { __kernel } {#2} }
595     \use:c { text_ #1 :Vn } \reserved@a {#3}
596   }

```

The odd use of *three* spaces here is needed as `ltxcmd` uses the name with one and two spaces to give a ‘friendly’ error message for a runaway argument: that means we can’t use it here.

```

597 \exp_args_generate:n { cnx }
598 \cs_set_protected:Npn \reserved@a #1
599   {
600     \cs_generate_variant:cn { text_ \str_lowercase:n {#1} case:nm } { V }
601     \ExpandArgs { cnx } \NewExpandableDocumentCommand
602       { Make#1case }
603       { O{} +m }
604       { \exp_not:c { Make#1case \c_space_tl \c_space_tl \c_space_tl } [####1] {####2} }
605   }
606 \reserved@a { Upper }
607 \reserved@a { Lower }
608 \reserved@a { Title }

```

These don’t get covered above so we do them manually.

```

609 \cs_generate_variant:Nn \text_titlecase_all:nm { V }
610 \cs_generate_variant:Nn \text_titlecase_first:nm { V }

```

Currently, `babel` uses the equivalence of `\oe` and `\OE` to force casing of some material, most notably in `\today`. To enable that to work, we have to set those commands equal even though the current case changing code does not work using this approach.

```

611 \cs_new_protected:cpn { MakeLowercase \c_space_tl \c_space_tl \c_space_tl } [#1] #2
612   {{
613     \let \OE \oe
614     \@@text@case@aux { lowercase } {#1} {#2}
615   }}
616 \cs_new_protected:cpn { MakeUppercase \c_space_tl \c_space_tl \c_space_tl } [#1] #2
617   {{
618     \let \oe \OE
619     \@@text@case@aux { uppercase } {#1} {#2}
620   }}
621 \cs_new_protected:cpn { MakeTitlecase \c_space_tl \c_space_tl \c_space_tl } [#1] #2
622   {{
623     \let \oe \OE
624     \@@text@case@aux { titlecase_ \reserved@b } {#1} {#2}
625   }}

```

`\NoCaseChange` protects its argument from the case change functions.

`\AddToNoCaseChangeList` Allows new commands to protect their arguments, eg `\AddToNoCaseChangeList{\eqref}` would protect the argument of `\eqref` in the same way as the argument of `\ref`.

```

626 \cs_new_protected_nopar:Npn \AddToNoCaseChangeList
627   {\tl_put_right:Nn \l_text_case_exclude_arg_tl}
628 \AddToNoCaseChangeList{ \NoCaseChange }
629 \cs_new_protected:Npn \NoCaseChange #1 {#1}
630 \cs_new_eq:NN \CaseSwitch \text_case_switch:nmmn
631 \cs_new_eq:NN \DeclareCaseChangeEquivalent
632   \text_declare_case_equivalent:Nn
633 \NewDocumentCommand \DeclareLowercaseMapping { o m m }

```

```

634 {
635   \IfNoValueTF {#1}
636     { \text_declare_lowercase_mapping:nn }
637     { \text_declare_lowercase_mapping:nnn {#1} }
638     {#2} {#3}
639 }
640 \NewDocumentCommand \DeclareTitlecaseMapping { o m m }
641 {
642   \IfNoValueTF {#1}
643     { \text_declare_titlecase_mapping:nn }
644     { \text_declare_titlecase_mapping:nnn {#1} }
645     {#2} {#3}
646 }
647 \NewDocumentCommand \DeclareUppercaseMapping { o m m }
648 {
649   \IfNoValueTF {#1}
650     { \text_declare_uppercase_mapping:nn }
651     { \text_declare_uppercase_mapping:nnn {#1} }
652     {#2} {#3}
653 }
654 \ExplSyntaxOff

655 \def\@uclclist{\oe\OE\o\O\ae\AE
656   \dh\DH\dj\DJ\l\L\ng\NG\ss\SS\ij\IJ\th\TH}

```

(End of definition for \MakeUppercase and others.)

1.9 Applying Patch files

Between major releases, small patches will be distributed in files `ltpatch.ltx` which must be added at this point.

Patch file code removed.

```

657 %\IfFileExists{ltpatch.ltx}
658 % {\typeout{=====^J%
659 %           Applying patch file ltpatch.ltx^J%
660 %           =====}
661 % \def\fmtversion@topatch{unknown}
662 % \input{ltpatch.ltx}
663 % \ifx\fmtversion\fmtversion@topatch
664 % \ifx\patch@level\@undefined
665 % \typeout{^J^J^J%
666 %           !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!^J%
667 %           !! Patch file 'ltpatch.ltx' not suitable for this^J%
668 %           !! version of LaTeX.^J^J%
669 %           !! Please check if initex found an old patch file:^J%
670 %           !! --- if so, rename it or delete it, and redo the^J%
671 %           !! initex run.^J%
672 %           !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!^J}%
673 % \batchmode \@@end
674 % \else

```

The code below adds the ‘patch level’ string to the first `\typeout` in the startup banner.

```

675 % \def\fmtversion@topatch{0}%
676 % \ifx\fmtversion@topatch\patch@level\else

```


1.11 Initialise file list

`\@providesfile` Initialise for use in the document. During `initex` a modified version has been used which leaves debugging information for `latexbug.tex`.

```
710 \def\@providesfile#1[#2]{%  
711   \wlog{File: #1 #2}%  
712   \expandafter\xdef\csname ver@#1\endcsname{#2}%  
713   \endgroup}
```

(End of definition for \@providesfile.)

`\@filelist` Reset `\@filelist` so files input while making the format are not listed. The list built up so far may take up a lot of memory and so it is moved to `\reserved@a` where it will be overwritten as soon as almost any `LATEX` command is issued in a class file. However the `latexbug.tex` program will be able to access this information and insert it into a bug report.

```
714 \let\@filelist\@gobble  
715 \def\@addtofilelist#1{\xdef\@filelist{\@filelist,#1}}%
```

(End of definition for \@filelist and \@addtofilelist.)

1.12 Preparation for supporting PDF in backends

At the current point in time, basic support for PDF in backends is not part of `LATEX` core; it is provided by external packages. At some time in the future that work will be placed into the kernel but for now it is separate and has to be explicitly loaded in the document.

In that code there is a command `\IfPDFManagementActiveTF` which can be used by packages in order to execute different code depending on the whether this basic backend support is loaded.

To make this also work properly when this external package is not loaded at all, we here add this command already in the kernel (with a trivial definition); thus any package can query this loading state in all circumstances. Once this basic PDF backend support gets moved to the kernel, this definition will vanish again from here or, rather, it will be replaced by a real test.

`\IfPDFManagementActiveTF` So long as the code for the basic backend support for PDF is not loaded, the test that is implicit here will always return the false branch. Once this code is loaded, this definition will get replaced by a real test (as it is then possible that the management code is either activated or not activated).

```
716 \let \IfPDFManagementActiveTF \@secondoftwo
```

(End of definition for \IfPDFManagementActiveTF.)

1.13 Do some temporary work for pre-release

This is a good place to load code that hasn't yet been integrated into the other files ...

1.14 Some last minute initializations ...

Load the first aid set of definitions for external packages that await updates.

```
717 \@input{latex2e-first-aid-for-external-files.ltx}
```


1.15 Dumping the format

Finally we make @ into a letter, ensure the format will be in the ‘normal’ error mode, and dump everything into the format file.

```
718 \makeatother
719 \errorstopmode
720 \dump
721 </2kernel>
```

Change History

- 1985-11-04 ltmath.dtx LaTeX2.09
 General: produce warning message if line extends into margin. Doesn't warn about formula overprinting equation number. 854
- 1989-04-10 ltfssbas.dtx v1.0a
 General: Starting with version numbers! `\ifmmode` added in `\math@group` 533
- 1989-04-10 ltfssbas.dtx v1.0b
 General: `\preload@sizes` added. 533
`\wrong@fontshape` changed to define substitution font/shape macro. 533
- 1989-04-10 ltfssini.dtx v1.0a
 General: Starting with version numbers `\newif` for `\@tempswa` added since this switch is unknown at the time when this file is read in. (latex.tex is loaded later.) `\math@famname` changed to `\math@version`. 695
- 1989-04-14 ltfssbas.dtx v1.0c
 General: More documentation added. 533
- 1989-04-15 ltfssini.dtx v1.0b
 General: `\mathfontset` renamed to `\mathversion`. 695
- 1989-04-19 ltfssbas.dtx v1.0d
 General: Even more doc. 533
- 1989-04-21 ltfssbas.dtx v1.0e
 General: Documentation is fun! Parameters of `\define@mathalphabet` changed. 533
- 1989-04-21 ltfssini.dtx v1.0c
 General: Changed to conform to fam.tex. 695
- 1989-04-23 ltfssbas.dtx v1.0f
 General: % in `\getanddefinefonts` added. 533
- 1989-04-26 ltfssini.dtx v1.0d
 General: `\xpt` added. 695
- 1989-04-27 ltfssbas.dtx v1.0g
 General: Documentation revised. 533
- 1989-04-27 ltfssini.dtx v1.0e
 General: Definitions of L^AT_EX symbols corrected. 695
- 1989-04-29 ltfssbas.dtx v1.0h
 General: Documented problem with `\halign`, and `\noalign` 533
- `\mathversion`: Test if version defined added. 546
- 1989-04-29 ltfssbas.dtx v1.0i
 General: Removed the `\halign` `\noalign` correction (wasn't bugfree) 533
- 1989-04-29 ltfssini.dtx v1.0f
 General: Corrections to L^AT_EX tabular env. added. 695
- 1989-05-01 ltfssbas.dtx v1.0j
 General: Default for `\baselinestretch` added. 533
- 1989-05-22 ltfssbas.dtx v1.0k
 General: Lines longer than 72 characters folded. 533
- 1989-05-22 ltfssini.dtx v1.0g
 General: Lines shortened to 72 characters 695
- 1989-09-14 ltfssbas.dtx v1.0m
 General: Global replacement: `\group` to `\mathgroup` 533
`\mathversion`: Corrected typo: `\endcsname` to `\endcsname`. 546
- 1989-11-07 ltfssini.dtx v1.0i
 General: All family, series, and shape names abbreviated. 695
- 1989-11-08 ltfssbas.dtx v1.0o
 General: First parameter of `\define@mathalphabet` and `\define@mathgroup` changed from string to control sequence. 533
- 1989-11-14 ltfssbas.dtx v1.0p
`\math@version`: Math version prefix 'mv@' added. 546
- 1989-11-19 ltfssbas.dtx v1.0q
`\define@newfont`: Group added. 549
`\wrong@fontshape`: Instead of calling `\family\default@family`, etc. we directly set `\f@family`, etc. 554
- 1989-11-22 ltfssbas.dtx v1.0r
`\math@version`: `\def` → `\edef` for `\math@version`. 546
- 1989-11-25 ltfssbas.dtx v1.0s
 General: All `\edef\font@name` changed to `\xdef\font@name`. Necessary after introduction of `\begingroup/\endgroup` in v1.0q. 533
 extra// → + in `\extra@def`. 533

1989-11-26 ltfssbas.dtx v1.0t	1990-01-25 ltfssini.dtx v1.1e
<code>\select@group</code> : <code>\bgroup</code> / <code>\egroup</code>	<code>\nfss@text</code> : Macro added. 718
changed to	1990-01-27 ltfssbas.dtx v1.2d
<code>\begingroup</code> / <code>\endgroup</code> to avoid	<code>\DeclarePreloadSizes</code> : Font identifier
empty Ord atom on math list. . . 557	set to <code>\relax</code> 541
1989-12-02 ltfssini.dtx v1.1b	1990-01-28 ltfssbas.dtx v1.2e
General: <code>\rmmath</code> renamed to	<code>\mathgroup</code> : <code>\newfam</code> let to
<code>\mathrm</code> 695	<code>\new@mathgroup</code> 533
1989-12-03 ltfssini.dtx v1.1c	1990-01-28 ltfssbas.dtx v1.2f
General: Some internal macros	<code>\define@newfont</code> : Added call to
renamed to make them	<code>\curr@fontshape</code> macro to allow
inaccessible. 695	substitution. 550
1989-12-05 ltfssbas.dtx v1.0u	<code>\wrong@fontshape</code> : Warning message
<code>\addto@hook</code> : <code>\addto@hook</code> added. . 561	slightly changed. 554
1989-12-05 ltfsstrc.dtx v1.0u fam.dtx	1990-01-28 ltfssini.dtx v1.2b
<code>\every@math@size</code> : Hook <code>\every@size</code>	<code>\em</code> : Call to <code>\@nomath</code> added. 715
added. 642	1990-02-08 ltfssini.dtx v1.1g
1989-12-13 ltfsstrc.dtx v1.0f	General: Protected the commands
<code>\use@mathgroup</code> : <code>\expandafter</code> added	<code>\family</code> , <code>\series</code> , <code>\shape</code> , <code>\size</code> ,
before final <code>\fi</code> 645	<code>\selectfont</code> , and <code>\mathversion</code> . 695
1989-12-16 ltfssbas.dtx v1.1a	1990-02-16 ltfssbas.dtx v1.2g
<code>\select@group</code> : <code>\relax</code> in front	General: Support for changes of
added. 557	<code>\baselineskip</code> without changing
Now four arguments. 557	the size. 533
Redefinition of alphabet now	<code>\math@version</code> : <code>\@nomath</code> added. . . 546
simpler. 557	1990-02-18 ltfsstrc.dtx v1.0j
Usage of ‘=’ macro added. 557	<code>\selectfont</code> : Redefine unprotected
1989-12-16 ltfsstrc.dtx v1.1a	version <code>\p@selectfont</code> instead of
<code>\selectfont</code> : Changed order of calls. 637	<code>\selectfont</code> 637
<code>\use@mathgroup</code> : Redefinition of	1990-03-14 ltfsstrc.dtx v1.0k
alphabet now simpler. 644	General: Added code for TeX3. 633
Usage of ‘=’ macro added. 644	<code>\extract@font</code> : Added code for
1990-01-18 ltfsstrc.dtx v1.0h	TeX3. 636
General: <code>\tracingfonts</code> meaning	1990-03-30 ltfssbas.dtx v1.2h
changed. 633	<code>\math@egroup</code> : Changed to have one
1990-01-20 ltfssbas.dtx v1.2a	arg. 559
<code>\math@bgroup</code> : Def. placed in this	1990-03-30 ltfsstrc.dtx v1.2h
file. 559	<code>\use@mathgroup</code> : Third argument
<code>\math@egroup</code> : Def. placed in this	removed (see <code>\math@egroup</code>). . . . 644
file. 559	1990-04-01 ltfssbas.dtx v1.2i
<code>\select@group</code> : Def for alph id	General: Code added from
changed. 557	<code>tracefmt.dtx</code> 533
1990-01-21 ltfssbas.dtx v1.2b	Support for TeX3. 533
<code>\select@group</code> : Code moved to	1990-04-01 ltfsstrc.dtx v1.0l
<code>\use@mathgroup</code> 557	General: Part of code moved to
1990-01-21 ltfsstrc.dtx v1.2b	<code>fam.dtx</code> 633
<code>\use@mathgroup</code> : Macro added to	<code>\tracingfonts</code> : Check if
allow cleaner interface. 644	<code>\tracingfonts</code> already defined. . 634
1990-01-23 ltfssbas.dtx v1.2c	1990-04-01 ltfsstrc.dtx v1.0o
General: <code>\no@version@warning</code>	<code>\tracingfonts</code> : Check if
renamed to <code>\no@alphabet@error</code> . 533	<code>\tracingfonts</code> defined removed
Macro <code>\no@alphabet@help</code> added 533	again. 634
<code>\no@alphabet@error</code> : Changed to	
error call 533	

1990-04-02 ltfssini.dtx v1.li	1991-08-14 ltpictur.dtx LaTeX2.09
General: <code>\input</code> of files now handled by <code>docstrip</code> 695	General: (RmS) inserted extra braces around entry for NFSS 927
1990-04-05 ltfsstrc.dtx v1.0m	1991-08-14 ltthm.dtx LaTeX2.09
<code>\selectfont</code> : Call <code>\tracingon</code> only if <code>\tracingfonts</code> greater than 3. . . 637	<code>\endtheorem</code> : Moved <code>\itshape</code> after <code>\item</code> to make it work with NFSS 957
1990-05-05 ltfsstrc.dtx v1.0n	1991-08-26 ltfssini.dtx v1.1n
<code>\selectfont</code> : <code>\tracingon</code> with new syntax. 637	<code>\reset@font</code> : Macro introduced . . . 718
1990-06-23 ltfssini.dtx v1.1k	1991-08-26 ltmiscen.dtx LaTeX2.09
<code>\nfss@text</code> : Changed to <code>\mbox</code> 718	<code>\verbatim</code> : <code>\@@par</code> added 837
1990-06-24 ltfssbas.dtx v1.2j	1991-08-26 ltpictur.dtx LaTeX2.09
<code>\DeclarePreloadSizes</code> : Missing percent added. 541	<code>\endpicture</code> : (RmS & FMI) extra boxing level around <code>\@picbox</code> to guard against unboxing in math mode (proposed by John Hobby) 925
1990-06-24 ltfsstrc.dtx v1.0o	1991-08-26 ltplain.dtx LaTeX2.09
<code>\baselinestretch</code> : Moved to <code>tracefnt.dtx</code> 642	<code>\tracingall</code> : Added
<code>\getanddefine@fonts</code> : <code>\Adding</code> tracing code. 646	<code>\errorcontextlines=\maxdimen</code> , suggested by J. Schrod 33
<code>\Macro</code> moved from <code>fam.dtx</code> 645	1991-09-29 ltboxes.dtx LaTeX2.09
Adding debug code. 646	<code>\mpfootnotetext</code> : (RmS) added
<code>\use@mathgroup</code> : Tracing code added. 645	<code>\reset@font</code> 892
1990-06-30 ltfssbas.dtx v1.2l	1991-09-29 ltfloat.dtx LaTeX2.09
<code>\showhyphens</code> : Macro added. 560	<code>\@footnotetext</code> : (RmS) added
1990-06-30 ltfsstrc.dtx v1.0p	<code>\reset@font</code> 991
<code>\use@mathgroup</code> : Added <code>\relax</code> after math group number. 645	1991-09-29 ltmath.dtx LaTeX2.09
1990-07-07 ltfsstrc.dtx v1.0q	<code>\@eqnnum</code> : RmS: <code>\reset@font</code> added. 853
<code>\getanddefine@fonts</code> : Group number added to tracing. 646	1991-09-29 ltsect.dtx LaTeX2.09
<code>\math@egroup</code> : Tracing code added. 645	<code>\@dottedtocline</code> : (RmS) added
<code>\use@mathgroup</code> : Group number added to tracing. 645	<code>\reset@font</code> for page number . . 970
1990-08-27 ltfsstrc.dtx 1.0r	1991-10-17 ltcntrl.dtx LaTeX2.09
<code>\type@restoreinfo</code> : Some extra tracing info. 641	<code>\@tfor</code> : (RmS) <code>\xdef</code> replaced by <code>\def</code> (See FMI's <code>array.doc</code>) 396
1990-08-27 ltfsstrc.dtx v1.0r	1991-10-25 ltbibl.dtx LaTeX2.09
<code>\getanddefine@fonts</code> : Correcting missing name after <code>\tracingon</code> . . 646	<code>\@citex</code> : added <code>\reset@font</code> , suggested by Bernd Raichle. . . 1000
1991-03-28 ltfssini.dtx v1.1m	1991-11-01 ltfloat.dtx LaTeX2.09
<code>\copyright</code> : Extra braces added. . . 718	<code>\footnote</code> : (RmS) Added
1991-03-30 ltfssini.dtx v1.2g	<code>\let\protect\noexpand</code> in <code>\footnote</code> , <code>\footnotemark</code> , and <code>\footnotetext</code> , since <code>\xdef</code> is used 991
<code>\newfont</code> : Definition added. 717	1991-11-04 ltlists.dtx LaTeX2.09
<code>\symbol</code> : Definition added. 717	<code>\makelabel</code> : (RmS) added default definition for <code>\makelabel</code> , to produce an error message. 874
1991-07-24 ltmiscen.dtx LaTeX2.09	1991-11-04 ltplain.dtx RmS
<code>\verbatim</code> : Added	General: Removed <code>\itemitem</code> since never needed/useful with L ^A T _E X. . 31
<code>\penalty\interlinepenalty</code> to definition of <code>\par</code> so that <code>\samepage</code> works 837	1991-11-06 ltbibl.dtx LaTeX2.09
1991-08-14 ltmath.dtx LaTeX2.09	<code>\@citex</code> : added code to remove a leading blank 1000
<code>\cases</code> : (RmS) inserted extra braces around entry for NFSS 848	

1991-11-13 ltbibl.dtx LaTeX2.09	avoid conflicts with other channels allocated by <code>\newread</code>	82
<code>\@bibitem</code> : Changed counter <code>enumi</code> to <code>enumiv</code> , as it says in the comment above		999
1991-11-21 ltfssini.dtx v1.1o		
<code>\reset@font</code> : Added extra braces for robustness.		718
Changed to protected version of macro.		718
1991-11-22 ltfloat.dtx LaTeX2.09		
<code>\footnote</code> : (RmS) Added <code>\let\protect\noexpand</code> in <code>\xfootnote</code> , <code>\xfootnotemark</code> , and <code>\xfootnotetext</code>		991
1991-11-22 ltlists.dtx LaTeX2.09		
<code>\@item</code> : (RmS) Changed second call to <code>\makelabel</code> to <code>\unhbox\@tempboxa</code> . Avoids problems with side effects in <code>\makelabel</code> and is more efficient.		874
1991-11-27 ltfssbas.dtx v1.3a		
General: All <code>\family</code> , <code>\shape</code> etc. renamed to <code>\fontfamily</code> etc.		533
1991-11-27 ltfssini.dtx v1.2a		
General: All <code>\family</code> , <code>\shape</code> etc. renamed to <code>\fontfamily</code> etc.		695
1992-01-06 ltfssini.dtx v1.2c		
General: added <code>slitex</code> code		695
1992-01-10 ltbibl.dtx LaTeX2.09		
<code>\@bibitem</code> : Changed <code>\c@enumiv</code> to value of <code>\@listctr</code>		999
1992-01-10 ltmath.dtx LaTeX2.09		
<code>equation</code> : (RmS): put <code>\hbox</code> around <code>\@eqnnum</code> to typeset the equation number in text mode (as in the <code>eqnarray</code> env.)		853
1992-01-10 lthm.dtx LaTeX2.09		
<code>\@othm</code> : (RmS) Check for existence of theorem environment		956
1992-01-14 ltbibl.dtx LaTeX2.09		
<code>\@biblabel</code> : removed <code>\hfill</code>		1002
1992-01-14 ltsect.dtx 0.0		
<code>\@starttoc</code> : (RmS) added <code>\immediate</code> to <code>\openout</code> as all <code>\write</code> commands are also executed <code>\immediate</code>		967
1992-02-26 ltbibl.dtx LaTeX2.09		
<code>\@lbibitem</code> : Added <code>\hfill</code> to restore left-alignment of bibliography labels in alpha style		999
1992-03-18 ltdefs.dtx LaTeX2.09		
General: (RMS) changed input channel from 0 to <code>\@inputcheck</code> to		
1992-03-18 ltfloat.dtx LaTeX2.09		
<code>\@xympar</code> : (RmS) added <code>\global\@ignorefalse</code>		986
<code>\end@float</code> : (RmS) changed <code>\@esphack</code> to <code>\@Esphack</code>		980
1992-03-18 ltlists.dtx 0.0		
<code>\trivlist</code> : (RmS): added <code>\@nmbrlistfalse</code>		869
1992-03-18 ltmiscen.dtx LaTeX2.09		
<code>\begin</code> : Changed <code>\@ignoretrue</code> to <code>\@ignorefalse</code> (as documented)		828
1992-03-21 ltfssini.dtx v1.2d		
General: Renamed <code>\text</code> to <code>\nfss@text</code> to make it internal.		695
1992-05-12 ltfssbas.dtx v1.3c		
<code>\extract@alph@from@version</code> : Macro added.		558
<code>\select@group</code> : Added call to <code>\extract@alph@from@version</code>		557
1992-07-26 ltfssbas.dtx v1.9a		
<code>\curr@fontshape</code> :		549
<code>\DeclareFontShape</code> : Introduced <code>\DeclareFontShape</code>		534
<code>\define@newfont</code> :		549
<code>\math@fonts</code> :		556
<code>\select@group</code> :		557
<code>\split@name</code> : Added splitting into <code>\f@encoding</code>		549
<code>\wrong@fontshape</code> :		554
1992-07-26 ltfstrc.dtx v2.0b		
<code>\s@fct@</code> :		654
<code>\s@fct@sub</code> : documentation fixes		655
<code>\selectfont</code> :		638
<code>\try@simple@size</code> :		648
<code>\try@size@range</code> :		652
<code>\use@mathgroup</code> :		645
1992-08-14 ltbibl.dtx LaTeX2.09		
<code>\@citex</code> : added missing argument braces around <code>\hbox</code> , found by Ed Szynter		1000
1992-08-14 ltboxes.dtx LaTeX2.09		
<code>\endminipage</code> : (RmS) replaced <code>\vskip-\lastskip</code> by <code>\unskip</code> (proposed by FMI)		892
1992-08-17 ltbibl.dtx LaTeX2.09		
<code>\@citex</code> : simplified code for removing leading blanks in citation key (proposed by Frank Jensen and Kresten Krab Thorup)		1000
1992-08-19 ltsect.dtx 0.0		
<code>\@xsect</code> : (RmS) corrected bug: stretch and shrink in argument to <code>\hskip</code>		

previously not negated	963	<code>\footnote:</code> (RmS) Changed all to 'def'protect'noexpand'protect'noexpand	991
1992-08-19 lthm.dtx LaTeX2.09		1992-12-03 ltfssini.dtx v?	
<code>\@othm:</code> (RmS) Changed error message to complain about undefined counter	956	<code>\hexnumber@:</code> Make it accept counters.	718
1992-08-20 ltfssini.dtx v1.4b		1993-03-08 preload.dtx v2.0b	
<code>\@setsize:</code> Added <code>\@currsize.</code> . . .	717	General: Added 12pt preloads	744
1992-08-24 ltdefs.dtx LaTeX2.09		1993-03-18 ltfssbas.dtx v2.0c	
<code>\@ifnextchar:</code> (Rms) <code>\@ifnextchar</code> didn't work if its first argument was an equal sign.	107	General: Changed all <code>\@tempdima</code> in <code>\@tempdimb</code> to avoid killing <code>\numberline</code>	533
1992-08-24 ltmiscen.dtx LaTeX2.09		1993-03-18 ltfstrc.dtx v2.1b	
<code>\@begin:</code> Added code to <code>\@begin</code> to remember line number. Used by <code>\@badend</code> to display position of non-matching <code>\@begin.</code>	828	General: Changed all <code>\@tempdima</code> in <code>\@tempdimb</code> to avoid killing <code>\numberline</code>	633
<code>\@verb:</code> Changed <code>\@verb</code> and <code>\@sverb</code> to work correctly in math mode . . .	842	Changed all <code>\@tempdimb</code> in <code>\@tempdimx</code> to avoid killing <code>\numberline</code>	633
1992-08-25 ltsect.dtx LaTeX2.09		1993-03-18 ltfstrc.dtx v2.1c	
<code>\@sect:</code> (FMi) replaced explicit setting of <code>\@svsec</code> by call to <code>\@seccntformat</code>	962	<code>\DeclareSizeFunction:</code> Added all args to avoid blanks problems . .	651
1992-09-18 ltlists.dtx LaTeX2.09		1993-04-09 lterror.dtx v1.0e	
<code>\@item:</code> (RmS) Added warning if <code>\@item</code> is used in math mode	872	<code>\@latexerr:</code> Mention The Companion	403
1992-09-18 lttab.dtx LaTeX2.09		1993-04-11 lterror.dtx v1.0f	
<code>\@array:</code> Changed <code>\@par</code> to <code>\@empty</code> to avoid starting new row e.g. after <code>\hline</code>	910	<code>\@latexerr:</code> Remove setting of errorcontextlines	403
1992-09-19 ltfstrc.dtx v2.0c		1993-05-05 lftncmd.dtx v2.0b	
<code>\@try@simple@size:</code>	648	General: Removed all LaTeX related cmds	747
1992-09-21 ltfssini.dtx v1.4d		1993-05-16 ltfssbas.dtx v2.0e	
<code>\@not@math@alphabet:</code> Macro defined. .	716	<code>\showhyphens:</code> Use <code>\reset@font</code> . .	560
1992-09-22 ltfssbas.dtx v1.91a		1993-07-16 ltfstrc.dtx v2.1h	
General: Introduced <code>\@tf@size</code> for math size.	533	General: Changed layout of info messages	633
1992-09-22 ltfstrc.dtx v2.1a		1993-07-17 ltoutenc.dtx 1.0d	
<code>\@getanddefine@fonts:</code> Introduced <code>\@tf@size</code> for math size.	646	General: changed <code>\@catcoding @</code> . . .	478
1992-11-13 ltfssini.dtx v?		1993-08-03 ltmiscen.dtx LaTeX2.09	
<code>\hexnumber@:</code> Made expandable. . . .	718	<code>\@enddocument:</code> Changed redefinition of <code>\global</code> to redefinition of <code>\@setckpt.</code>	820
1992-11-23 ltcounts.dtx LaTeX2.09		1993-08-05 ltpictur.dtx LaTeX2.09	
<code>\@stepcounter:</code> Replaced <code>{}</code> in <code>\@stepcounter</code> by <code>\@begingroup</code> <code>\@endgroup</code> to avoid adding an empty ord in math mode	523	<code>\@circle:</code> (RMS) Added error message if <code>\@circle</code> is used in math mode. .	947
1992-11-26 ltboxes.dtx LaTeX2.09		1993-08-05 ltsect.dtx LaTeX2.09	
<code>\@mpfootnotetext:</code> (RmS) added protection for <code>\edef</code>	892	<code>\@sect:</code> (RmS) Made sure that <code>\protect</code> works correctly in expansion of <code>\the</code> counter	962
1992-11-26 ltfloat.dtx LaTeX2.09		1993-08-05 ltspc.dtx LaTeX2e	
<code>\@footnotetext:</code> (RmS) added protection for <code>\edef</code>	991	<code>\@hspace:</code> (RmS) Removed superfluous <code>\leavevmode</code> in <code>\@hspace</code> and <code>\@hspacer,</code> as suggested by CAR.	448

1993-08-05 lttab.dtx latex2e	
<code>\tabular*</code> : Replaced	
<code>\expandafter\def</code> by <code>\@namedef</code> .	909
1993-08-06 ltbibl.dtx LaTeX2.09	
<code>\@citex</code> : Moved writing to .aux file in	
loop over citation keys so that	
leading blanks are removed there	
as well.	1000
1993-08-13 ltoutenc.dtx 1.0f	
General: Protected against active @	
sign.	478
1993-08-13 preload.dtx v2.0c	
General: Added <code>\relax</code> at end of font	
names.	745
1993-08-16 ltoutenc.dtx 1.0g	
General: Needs space after <code>\string</code>	478
1993-08-18 ltfsdcl.dtx v2.0e	
<code>\new@mathversion</code> : Exchanged names	
of encodings in warning message of	
<code>\SetSymbolFont</code>	676
1993-09-02 ltfsstrc.dtx v2.1i	
General: Corrected name of sgen size	
function.	633
1993-09-03 ltmiscen.dtx LaTeX2.09	
<code>\verbatim@nolig@list</code> : Replaced	
<code>\@noligs</code> by extensible list	842
1993-09-07 ltmiscen.dtx LaTeX2.09	
<code>\verb@balance@group</code> : (RmS)	
Changed definition of <code>\verb</code> so	
that it detects a missing second	
delimiter.	842
1993-09-08 ltmiscen.dtx LaTeX2.09	
<code>\enddocument</code> : Added warning in case	
of undefined references.	820
1993-09-15 ltfsbas.dtx v2.0g	
<code>\DeclareFontEncoding</code> : Corrected:	
<code>\default@T</code> to <code>\default@M</code>	537
1993-09-15 ltfsstrc.dtx v2.1j	
General: Corrected spelling of	
<code>\noexpand</code>	633
1993-09-19 lterror.dtx LaTeX2.09	
<code>\@invalidchar</code> : (RmS) Error message	
for invalid input characters. . . .	407
1993-11-02 ltmath.dtx LaTeX2.09	
General: RmS: Corrected description	
of <code>\@eqnset</code> , moved <code>\@eqnset</code>	
accordingly and removed extra	
<code>\tabskip</code> assignment.	854
1993-11-03 ltmath.dtx LaTeX2e	
General: RmS: Initialized <code>\everycr</code> to	
empty	854
1993-11-03 ltpictur.dtx LaTeX2.09	
General: (RmS) changed <code>\halign</code> to	
<code>\ialign</code> to initialize <code>\tabskip</code> and	
<code>\everycr</code>	927
1993-11-11 ltfsini.dtx v2.1a	
<code>\normalfont</code> : Macro added	718
1993-11-11 ltfsstrc.dtx v2.2a	
General: Option concept added for	
LaTeX2e	633
1993-11-14 ltclass.dtx v0.2a	
<code>\@current</code> : Name changed from	
<code>\@currentextension</code>	1043
<code>\@reset@ptions</code> : macro added . . .	1071
<code>\AtEndDocument</code> : Included extension	
in the generated macro name for	
package and class hooks.	1071
<code>\documentstyle</code> : Added	
<code>\RequirePackage</code>	
<code>\@unusedoptionlist</code> stuff. . . .	1059
<code>\load@onefilewithoptions</code> : Moved	
resetting of <code>\default@ds</code> , <code>\ds@</code> and	
<code>\@declareoptions</code> here, from the	
end of <code>\ProcessOptions</code>	1065
<code>\NeedsTeXFormat</code> : made more robust	
for alternative syntax for other	
formats.	1061
<code>\ProcessOptions*</code> : Optimize ‘empty	
option’ code.	1056
Stop adding the global option list	
inside class files.	1056
1993-11-14 ltdefs.dtx v0.2a	
<code>\g@addto@macro</code> : Made global	112
1993-11-15 ltclass.dtx v0.2b	
<code>\documentstyle</code> : Modified to match	
<code>\ProcessOption*</code>	1059
<code>\ProcessOptions*</code> : Star form	
added.	1056
1993-11-17 ltclass.dtx v0.2c	
<code>\@fileswith@ptions</code> : Macro added	1071
<code>\@badrequireerror</code> : Macro added	1073
<code>\@twoloadclasserror</code> : Macro added	1073
<code>\CurrentOption</code> : Name changed from	
<code>\@curroption</code>	1043
<code>\DeclareOption*</code> : Error checking	
added	1055
<code>\load@onefilewithoptions</code> : Added	
trap for two <code>\LoadClass</code>	
commands.	1067
<code>\NeedsTeXFormat</code> : Name changed from	
<code>\NeedsFormat</code>	1061
<code>\ProcessOptions*</code> : restoring	
<code>\@fileswith@ptions</code> added. . .	1056
1993-11-18 ltclass.dtx v0.2d	
<code>\documentstyle</code> : Modified	
<code>\RequirePackage</code> stuff.	1059

\ExecuteOptions: Use		\@newcommand: Macro added	84
\CurrentOption not		\@newenv: Macro interface changed . .	87
\reserved@a	1059	\@xargdef: Macro interface changed .	84
\NeedsTeXFormat: \fmtname		\@yargd@f: Avoid \@?@? token	85
\fmtversion not \@.	1061	Macro interface changed	85
1993-11-21 ltfiles.dtx LaTeX2e		\newcommand: Macro reimplemented	
\@missingfileerror: Stop infinite		and extended	83
looping on \@er@ext	470	\renewcommand: Macro reimplemented	
1993-11-21 ltmiscen.dtx v0.9a		and extended	86
\@verbatim: use \@verbatim@font		\renewenvironment: Macro	
instead of \tt	837	reimplemented and extended	87
\verb: Use \@verbatim@font instead of		\two@digits: Macro added	80
\tt.	842	1993-11-23 ltoutput.dtx v0.1a	
\verbatim@font: Macro added	838	\paperheight: Register added	1164
1993-11-22 ltclass.dtx v0.2f		\paperwidth: Register added	1164
\@fileswithoptions: Made the		1993-11-23 ltoutput.dtx v0.1c	
default [] not		\@enlargepage: Command added	1224
[\@unknownversion]	1063	\@kludgeins: Insert added	1223
\@ifl@ter: Added //00 so parsing		\@makecol: Command changed	1178
never produces a runaway		\@specialoutput: Command	
argument.	1048	changed	1171
General: \@unknownversion		\enlargethispage*: Commands	
removed	1038	added	1223
\load@onefilewithoptions: Made the		1993-11-24 ltfntcmd.dtx v2.1a	
initial version [] not		\maybe@ic: Use \@st@ic	752
[\@unknownversion]	1065	\t@st@ic: Macro added	753
1993-11-22 ltdefs.dtx LaTeX2e		1993-11-24 ltfssini.dtx v2.1a	
\@minus: Macro added	81	General: Removed \xpt stuff	718
\@plus: Macro added	81	1993-11-24 ltlogos.dtx LaTeX2e	
\@checkcommand: Macro added	88	\LaTeX: Macro changed	450
\@providecommand: Macro added	88	1993-11-28 ltclass.dtx v0.2h	
1993-11-22 lterror.dtx LaTeX2e		\@twoclasseserror: Macro added	1074
\@errorcontextlines: Macro added	403	General: Assorted commands now in	
1993-11-22 ltfiles.dtx LaTeX2e		the kernel removed.	1042
\@listfiles: Removed checking for		Directory syntax checking moved to	
\@unknownversion	472	dircheck.dtx	1042
1993-11-22 ltlength.dtx LaTeX2e		Primitive filenames now terminated	
\@settodim: Macro added	531	by space not \@relax.	1042
\@settopoint: Macro added	532	\@endfilecontents: Don't globally	
\@settodepth: Macro added	531	allocate a write stream (always use	
\@settoheight: Macro added	531	15)	1074
1993-11-22 ltlogos.dtx LaTeX2e		1993-11-28 ltfiles.dtx LaTeX2e	
\LaTeXe: Macro added	450	\@missingfileerror: Use filename	
1993-11-23 ltclass.dtx v0.2g		parser from dircheck	470
\@use@option: Name changed from		1993-11-29 ltoutput.dtx v1.0b	
\@executeoption	1058	\@makecol: \@makespecialcolbox	
General: Various macros now moved		added	1178
to latex.tex.	1042	1993-11-29 ltplain.dtx LaTeX2e	
Warnings and errors now directly		General: All accents in decimals;	
coded.	1042	suggested by Paul Taylor	32
1993-11-23 ltdefs.dtx LaTeX2e		1993-11-30 ltoutput.dtx v1.0c	
\@argdef: Macro added	84	\@fl@tracemessage: Commands	
\@ifundefined: Redefined to remove a		added	1226
trailing \fi	106		

1993-12-01 fontdef.dtx v2.1a		1993-12-04 ltfiles.dtx v0.9b	
General: Update for LaTeX2e	723	\@iinput: Macro reimplemented . . .	469
1993-12-01 ltoutput.dtx v1.0e		\@input: Macro reimplemented . . .	470
\@reinserts: Command added . . .	1187	\IfFileExists@: Macro added	466
1993-12-03 ltboxes.dtx v0.1a		\input: Macro reimplemented	469
\@argrbox: macro removed	895	1993-12-05 ltfloat.dtx LaTeX2e	
\@begin@tempboxa: macro added . . .	880	\@dblfloatplacement: Command	
\@end@tempboxa: macro added	880	changed	982
\@iirsbox: redefined to support		\@xfloat: Command changed	976
\height	895	1993-12-05 ltoutput.dtx v1.0f	
\@imakebox: macro modified	881	\@addtobot: Command changed . . .	1200
\@irsbox: redefined to support		\@addtocurcol: Command changed . .	1202
\height	895	\@addtodblcol: Command changed . .	1217
\@isavebox: color support	884	\@addtonextcol: Command changed . .	1212
extra group	884	\@addtotoporbot: Command	
\@isavepicbox: extra group	884	changed	1201
\@makebox: default changed from x to		\@boxfpsbit: Command added . . .	1230
c	880	\@flcheckspace: Command added . . .	1232
\@makepicbox: macro modified	881	\@flsetnum: Command added	1231
\@savebox: default c not x	884	\@flsettextmin: Command added . . .	1231
\bm@b: macros added	880	\@flstop: Commands added	1227
\endlrbox: macro added	885	\@flupdates: Command added	1234
\fbbox: extra group	885	\@fpsaddddefault: Command added . .	1228
\lrbox: color support	884	\@getfpsbit: Command added	1229
macro added	884	\@opcol: Command changed	1177
\makebox: modified	879	Hook added	1177
\mbox: extra group	880	\@outputpage: Command changed . . .	1188
\minipage: Redefined to support extra		\@resethfps: Command added	1230
optional arguments	891	\@setfloattypecounts: Command	
\newsavebox: Pass the whole of arg 1		added	1229
to \@ifdefinable	883	\@setfpsbit: Command added	1230
\parbox: Redefined to support extra		\@shipoutsetup: Command added . . .	1188
optional arguments	888	\@startcolumn: Command changed . . .	1196
\raisebox: redefined to support		\@startdblcolumn: Command	
\height	894	changed	1196
\sbox: color support	884	\@testfp: Command added	1230
extra group	884	\@textfloatsheight: Commands	
\set@color: color support	882	added	1228
macro added	882	\@topnewpage: Commands changed . . .	1167
1993-12-03 ltclass.dtx v0.2i		\@tryfcolumn: Command changed . . .	1197
\@cls@pkg: Name changed to avoid		\@writesetup: \@startpagehook	
clash with output routine.	1072	added	1188
General: \@onlypreamble: Many		\output: Command changed	1171
commands declared.	1042	1993-12-06 ltclass.dtx v0.2k	
Removed obsolete		\ExecuteOptions: Preserve	
\@documentclass	1042	\CurrentOption.	1059
1993-12-03 lterror.dtx v1.0b		1993-12-06 ltoutput.dtx v1.0f	
\@latexerr: Set		\@specialoutput: Unboxing of 255	
\@errorcontextlines to -1	403	added to rescue writes	1171
1993-12-03 ltfssini.dtx v2.1a		1993-12-06 ltoutput.dtx v1.0g	
General: update for LaTeX2e	695	\@topnewpage: \@floatplacement	
1993-12-04 ltfilehook.dtx v0.9b		placement bug fixed	1167
\unqu@tefilef@und: Macro added	1110	1993-12-07 ltclass.dtx v0.2l	
		\ProvidesFile: Macro added	1053

1993-12-07 ltclass.dtx v0.2m		<code>\fix@penalty</code> : Macro added	753
<code>\load@onefilewithoptions</code> : Reset		<code>\maybe@ic</code> : Macro name changed . .	752
<code>\CurrentOption</code>	1065	<code>\maybe@ic@</code> : Macro and name	
1993-12-07 ltoutenc.dtx 1.1		changed	752
General: Protected all special		<code>\sw@slant</code> : Macro changed	753
characters with <code>\string</code>	478	<code>\textup</code> : Macros changed	750
1993-12-07 ltoutenc.dtx v1.1		1993-12-11 ltmath.dtx v0.9g	
General: Made all character numbers		General: Added a group around the	
decimal.	475	first argument of <code>\frac</code> to prevent	
Removed a lot of equal signs and		changes (for example font changes)	
the like.	475	from modifying the contents of the	
1993-12-08 ltboxes.dtx v0.1b		second argument.	854
<code>\@begin@tempboxa</code> : Extra braces for		1993-12-11 ltoutenc.dtx v1.2a	
color support (braces removed		General: Corrected for <code>t1enc</code> , <code>math</code>	475
from other macros)	880	1993-12-11 ltsect.dtx LaTeX2e	
<code>\@irsbox</code> : fix typo	895	<code>\author</code> : Added default	958
<code>\@parboxto</code> : <code>\endgraf</code> added due to		<code>\title</code> : Added default	958
extra group in <code>\@begin@tempboxa</code>	888	1993-12-11 ltxref.dtx LaTeX2e	
<code>\lrbox</code> : move <code>\@endpefalse</code> out of the		<code>\setref</code> : Macro added	799
inner group	884	<code>\pageref</code> : Macro reimplemented . . .	799
1993-12-08 ltfntcmd.dtx v2.1b		<code>\ref</code> : Macro reimplemented	799
General: Macros <code>\rm</code> , <code>\bf</code> and <code>\sf</code>		1993-12-12 ltoutput.dtx v1.0h	
moved to <code>classes.dtx</code>	755	<code>\@cflb</code> : <code>boxmaxdepth</code> setting moved	1194
1993-12-08 ltlists.dtx LaTeX2e		defs changed to <code>lets</code>	1194
<code>\@item</code> : use <code>\sbox</code> to support colour	874	<code>\@cflt</code> : name changed	1194
1993-12-08 ltspace.dtx LaTeX2e		<code>\@doclearpage</code> : defs changed to <code>lets</code>	1177
<code>\@bsphack</code> : Command reimplemented	437	<code>\@makecol</code> : defs changed to <code>lets</code> . .	1178
Command reimplemented; late		<code>\@resetfyps</code> : Warnings added:	
birthday present for Chris	437	minimal	1230
<code>\@vbsphack</code> : Command added	440	<code>\@startdblcolumn</code> : defs changed to	
1993-12-09 ltboxes.dtx v0.1c		<code>lets</code>	1196, 1197
<code>\@irsbox</code> : fix another typo	895	<code>\@topnewpage</code> : braces removed . . .	1167
1993-12-09 ltclass.dtx v0.2n		<code>\@tryfcolumn</code> : defs changed to <code>lets</code>	1197
<code>\documentstyle</code> : input 209		<code>\fl@tracemessage</code> : Commands	
compatibility file.	1059	changed	1226
1993-12-09 ltfiles.dtx v0.9e		1993-12-13 ltclass.dtx v0.2o	
<code>\document</code> : Hook added	453	General: Removed setting	
1993-12-09 ltmiscen.dtx v0.9e		<code>\errorcontextlines</code> (now in	
<code>\enddocument</code> : Hook added	820	<code>latex.tex</code>)	1042
1993-12-10 ltoutenc.dtx v1.2		<code>\documentstyle</code> : compatibility file	
General: Added source code for		now <code>latex209.sty</code>	1059
<code>t1enc.sty</code>	475	<code>\usepackage</code> : Fixed error handling	1061
1993-12-11 ltfntcmd.dtx v3.0a		1993-12-13 ltdirchk.dtx v0.2a	
General: Complete reworking of all		General: on the ‘ <code>docstrip</code> ’ pass, do not	
text commands, using just one		check <code>openin</code> path	10
creator function	747	<code>\IfFileExists</code> : Removed interactive	
italic correction now put in front of		prompting for current directory	
penalty before <code>glue</code>	747	syntax	10
newcommands replaced by defs . .	747	<code>\strip@prefix</code> : modified, name	
newfontswitch command corrected		changed from <code>\stripmeaning</code>	5
and changed	747	1993-12-13 ltlists.dtx latex2e	
<code>\DeclareTextFontCommand</code> : Macro		<code>\trivlist</code> : Initialised <code>\@itemlabel</code>	869
changed	749	1993-12-13 ltmiscen.dtx v0.9h	
<code>\emph</code> : Macro changed	750	<code>\noligs</code> : Readded <code>\@noligs</code>	843

- `\@verbatim`: Readded `\@noligs` . . . 837
- Removed optional argument of `\item` 837
- `center`: Removed optional argument of `\item` 834
- `flushleft`: Removed optional argument of `\item` 835
- `flushright`: Removed optional argument of `\item` 836
- 1993-12-13 ltoutenc.dtx v1.2b
 - General: Corrected file name in driver code. 475
- 1993-12-13 lttab.dtx latex2e
 - `\tabbing`: Removed optional argument of `\item` 904
- 1993-12-14 ltoutput.dtx v1.0i
 - General: Section added to declare all parameters 1239
- 1993-12-15 ltboxes.dtx v0.1d
 - `\@iminipage`: Changed default from ‘c’ to ‘s’ 891
 - `\@iparbox`: Changed default from ‘c’ to ‘s’ 888
 - `\minipage`: Changed default from ‘c’ to ‘s’ 891
 - extra space removed. 891
 - `\parbox`: Changed default from ‘c’ to ‘s’ 888
- 1993-12-15 ltclass.dtx v0.2p
 - General: Removed extra ‘s’ from `\@@warnings` 1042
- 1993-12-16 ltlogos.dtx LaTeX2e
 - `\LaTeXe`: Extended logo by DPC . . 450
- 1993-12-16 ltmath.dtx v0.9i
 - `\@eqnocr`: use `\refstepcounter` instead of shortcut 855
 - General: use `\refstepcounter` instead of shortcut 854
- 1993-12-16 ltmiscen.dtx v0.9i
 - General: `\literal` added 843
- 1993-12-16 ltpage.dtx LaTeX2e
 - `\mark`: Init `\mark` at begin document 1036
- 1993-12-16 ltspace.dtx LaTeX2e
 - `\obsphack`: Corrected optimisation . 437
- 1993-12-16 lttab.dtx latex2e
 - `\@xhline`: Measure from middle of vertical rules 919
- 1993-12-17 ltclass.dtx v0.2q
 - `\@documentclasshook`: Macro added 1042
 - `\@fileswithoptions`: Add `\@compatibility` hook 1063
 - `\documentstyle`: Match Alan’s new code. 1059
- 1993-12-17 ltoutenc.dtx 1.3
 - General: Added this section 479
 - Removed all the hackery for use in `\DeclareFontEncoding`, and redid everything using `\DeclareTextFoo`. 491, 494
 - Removed the catcode hackery, since the file is only read as a package in the preamble, and removed all the messages on the screen, which just confuse users. Replaced them by the appropriate `\ProvidesPackage` commands. Added XXXenc. . . . 478
- 1993-12-17 ltoutenc.dtx v1.3
 - General: Added `\EncodingSpecificAccent`, `\EncodingSpecificAccentedLetter` and `\EncodingSpecificCommand`. 475
 - Made Rokicki’s encoding a proper encoding scheme rather than a variant of OT1. 475
- 1993-12-17 ltoutput.dtx v1.0j
 - `\@opcol`: Hook removed 1177
 - `\@specialoutput`: Page room test added 1172
 - `\@topnewpage`: check for vsize too small added 1167
 - Page room test added 1169
 - `\@writesetup`: —and then removed 1188
 - `\fl@tracemessage`: tracefloatvals made a document command . . 1226
- 1993-12-17 ltpage.dtx LaTeX2e
 - `\mark`: Removed init `\mark` at begin document, since it doesn’t work. 1036
 - `\rightmark`: Stopgap solution to mark `\leftmark` and `\rightmark` work without initializing mark until the problem is solved. 1036
- 1993-12-18 ltoutenc.dtx 1.3b
 - General: Fixed typos with `\ProvidesPackage` lines. Added the `\NeedsTeXFormat` line. Added the last argument to `\DeclareEncoding`. Moved the use of the encodings to after their declaration. 478
 - Replaced the missing last argument to `\DeclareFontEncoding`. 491, 494
- 1993-12-18 ltoutenc.dtx 1.3c
 - General: Rewrote for the new syntax of `\EncodingSpecific`. . . . 491, 494
 - Split `\EncodingSpecificAccent` up into `\EncodingSpecific` and `\DeclareAccent`. 479

1993-12-18 ltoutenc.dtx v1.3a		1994-01-14 ltdirchk.dtx v0.2d	
General: Replaced OT3 by XXX	475	\IfFileExists: Close the texsys.aux output stream	10
1993-12-18 ltoutenc.dtx v1.3b		1994-01-15 ltfiles.dtx v0.9o	
General: Corrected typos.	475	\document: move \@preamblecmds after document hook	455
Replaced the missing last argument to \DeclareFontEncoding.	475	1994-01-17 ltclass.dtx v0.2s	
1993-12-18 ltoutenc.dtx v1.3c		\@fileswithoptions: Modify to reduce parameter stack usage	1063
General: A new syntax, separating accent-definitions from encoding-specific definitions, and allowing encoding-specific \chardef, \let, etc.	475	General: Added many more \@onlypreamble commands	1042
Rewrote for the new syntax of \EncodingSpecific.	475	Wrapped long lines to column 72	1042
1993-12-18 ltoutenc.dtx v1.3d		\load@onefile@withoptions: Modify to reduce parameter stack usage	1068
General: Some T1 stuff had drifted into the OT1 file.	475	1994-01-17 ltfiles.dtx LaTeX2e	
1993-12-18 ltpage.dtx LaTeX2e		\listfiles: New Version, adds 'tex' if needed, and lines up columns	472
\slippy: Added \emergencystretch	1037	1994-01-17 ltfssbas.dtx v2.1a	
1993-12-19 ltclass.dtx v0.2r		General: New math font setup	533
\endfilecontents: Different message when ignoring a file	1074	\curr@math@size: New math font setup	548
1993-12-19 ltfntcmd.dtx v3.0b		\everydisplay: New math font setup	547
General: \@pdef command added	747	\everymath: New math font setup	547
Added by ASAJ.	755	\frozen@everydisplay: New math font setup	548
Made \@newfontswitch produce an error if command already exists, and added \@renewfontswitch,		\frozen@everymath: New math font setup	547, 548
ASAJ	747	\math@version: New math font setup	546
Other tidying	747	1994-01-17 ltfssini.dtx v2.1e	
Some more tidying done	747	\not@math@alphabet: Message changed	716
Untidying added, so this is now a TEMPORARY version.	747	1994-01-17 ltfsstrc.dtx v2.3a	
Wording changes by CAR.	755	General: New math font setup	633
\DeclareOldFontCommand: Corrected and tidied	754	\check@mathfonts: New math font setup	643
\DeclareTextFontCommand: Corrected and tidied	749	\glb@currsize: New math font setup	641
1993-12-19 ltspace.dtx LaTeX2e		\restglb@settings: New math font setup	644
\@bsphack: There seem to be problems with selfmade birthday presents	438	1994-01-18 ltbibl.dtx LaTeX2e	
1993-12-20 ltdefs.dtx LaTeX2e		\bibliography: Use \@input@ so include files are listed.	1000
\@reargdef: Kept old version of \@reargdef, for array.sty	85	1994-01-18 ltclass.dtx v0.2t	
1993-12-20 ltfiles.dtx v0.9m		\@ifclassloaded: Fix typo \@pkgetension	1047
\@obsoletefile: Added this command, removed \@oldfilewarning	472	1994-01-18 ltfilehook.dtx v0.9p	
1994-01-05 fontdef.dtx v2.1d		\unqu@tefilef@und: New Definition	1110
General: Removed nf prefix from file names.	725	1994-01-18 ltfiles.dtx v0.9p	
1994-01-13 ltmath.dtx v0.9o		\@iffileonpath: Macro added	468
\@eqnocr: correcting 0.9i	855	\@input: do not use a different definition for \input@path	470
General: correcting 0.9i	854	\@input@: Macro added	470
		\IfFileExists@: New Definition	466
		\includeonly: Use \@input@ so include files are listed.	459

1994-01-18 ltfssini.dtx v2.1f	1994-01-25 ltfssbas.dtx v2.1b
\backslash not@math@alphabet: Message corrected	\backslash math@version: Corrections for math setup
1994-01-18 ltmiscen.dtx v0.9p	1994-01-25 ltmath.dtx LaTeX2e
\backslash verbatim: Add	\backslash bordermatrix: Removed \backslash p@renwd.
\backslash global \backslash @inlabeledfalse	1994-01-26 ltfstrc.dtx v2.3c
Only add \backslash penalty if in hmode	\backslash check@mathfonts: Correct trace info placement
1994-01-19 fontdef.dtx v2.1e	\backslash restglb@settings: Correct trace info placement
General: Added missing setting for symbols in bold version.	1994-01-27 ltfntcmd.dtx v3.1a
1994-01-19 ltdirchk.dtx v0.2e	\backslash nocorrlist: Only ., used as default for cm fonts
\backslash IfFileExists: name changed from \backslash test	1994-01-29 ltclass.dtx v0.2v
\backslash input@path: No longer check that an empty group is in the path	\backslash @unprocessedoptions: Macro added.
\backslash strip@prefix: name changed from \backslash strip@meaning, to match NFSS.	\backslash load@onefile@withoptions: All options raise error if no \backslash ProcessOptions appears
1994-01-19 ltmath.dtx v1.0n classes	1994-01-31 ltdefs.dtx v0.2w
\backslash mathindent: Deferred setting of \backslash mathindent	\backslash g@addto@macro: Use toks register to avoid ‘hash’ problems
1994-01-20 ltdirchk.dtx v0.2f	1994-01-31 ltfiles.dtx v0.9t
General: \backslash @copytexsys and the texsys.new file removed	\backslash document: set \backslash @normalsize or \backslash normalsize if necessary
Modify all of ltxcheck	1994-01-31 ltfntcmd.dtx v3.1b
\backslash IfFileExists: \backslash @copytexsys removed	General: \backslash @normalsize no longer defined
1994-01-21 ltclass.dtx v0.2u	1994-02-01 ltpage.dtx LaTeX2e
\backslash documentstyle: compatibility file now latex209.def.	\backslash pagestyle: (DPC) Modify to get nicer error message
1994-01-21 ltdirchk.dtx v0.2g	\backslash thispagestyle: (DPC) Modify to get nicer error message
General: Improve documentation, reorganize docstrip module	1994-02-02 ltclass.dtx v0.2x
\backslash filename@parse: Minor changes, and add Mac version (:).	\backslash load@onefile@withoptions: Only run the hook and options check if the file was loaded.
\backslash today: Name changed from \backslash stamp, to save memory	1994-02-03 ltoutput.dtx v1.0k
1994-01-21 ltfloat.dtx LaTeX2e	\backslash @make@specialcolbox: correct mistakes in the documentation
\backslash xfloat: Added missing percent characters.	1994-02-07 ltclass.dtx v0.2y
1994-01-21 ltmiscen.dtx v0.9s	\backslash @fileswithoptions: Run \backslash @compatibility on the first class to start (not the first to finish)
\backslash verbatim@font: Removed unnecessary category code hackery.	\backslash @ifclasswith: Add extra ,s so ‘two’ is not matched with ‘twocolumn’
1994-01-24 ltdirchk.dtx v0.2h	\backslash ProcessOptions*: Add extra ,s so ‘two’ is not matched with ‘twocolumn’
\backslash IfFileExists: Stop testing once texsys.aux has been found	1994-02-07 ltfssbas.dtx v2.1c
1994-01-24 ltpage.dtx LaTeX2e	\backslash DeclareFontEncoding: revert catcode settings earlier
\backslash pagestyle: (DPC) Complain if pagestyle is undefined.	\backslash DeclareFontShape@: revert catcode settings earlier
1994-01-25 ltdirchk.dtx v0.2i	
General: Protect against looping on \backslash @@input and \backslash @@end.	

1994-02-08 ltoutput.dtx v1.0k	1994-03-04 ltvers.dtx v1.0a
General: Documentation and tasks	General: Initial version, split from
tidied. 1152	latex.dtx 37
1994-02-10 ltclass.dtx v0.2z	1994-03-07 ltboxes.dtx v0.1a
\@documentclasshook: Changed the	\@mpfootnotetext: Extra group for
name from \@compatibility to	color 892
\@documentclasshook, and added	1994-03-07 ltboxes.dtx v1.0a
the check for whether	General: Unify format with other
\@normalsize has been defined.	Kernel files 879
ASAJ. 1042	1994-03-07 ltdefs.dtx v1.0a
\@fileswithoptions: Renamed	\@italiccorr: Macro added 81
\@compatibility to	1994-03-07 ltfiles.dtx v1.0a
\@documentclasshook. ASAJ. 1063	General: Initial version, split from
1994-02-10 ltffsbas.dtx v2.1d	latex.dtx 451
\addto@hook: Made \addto@hook	Long lines wrapped to 72 columns 451
long. 561	1994-03-07 ltfinal.dtx v0.1a
1994-02-10 ltffscmp.dtx v2.1d	General: Add code from the old
\scan@fontshape: scan away stuff	dump.dtx 1276
after pt 659	Initial version, split from latex.dtx 1260
1994-02-22 ltffsini.dtx v2.1g	move code here from lhyphen.dtx 1266
General: Correct error message 720	Remove oldcomments
1994-02-24 ltffsbas.dtx v2.1e	environment 1260
\DeclareFontShape: Separate	use \InputIfFileExists not
restoration of catcodes for fd	\IfFileExists 1266
cmds 534	1994-03-07 ltfloat.dtx v1.0a
\define@newfont: Separate	\@endfloatbox: (DPC) Extra group
restoration of catcodes for fd	for colour 981
cmds 550	\@footnotetext: (DPC) Extra group
\nfss@catcodes: Separate restoration	for colour 991
of catcodes for fd cmds 550	\@xfloat: (DPC) Extra group for
1994-02-25 ltdirchk.dtx v0.2j	colour 977
General: Remove need for drv file 1	1994-03-07 lthyphen.dtx v0.1c
1994-03-01 ltdirchk.dtx v0.2k	General: move the 2kernel code to
General: Add unstripped module, so	ltfinal.dtx 1258
that dircheck.dtx may be used	1994-03-07 ltlength.dtx v1.0a
with initex 1	\@settodim: (DPC) Extra group for
1994-03-02 ltboxes.dtx v0.1e	colour 531
General: Add 2kernel module 879	1994-03-07 ltlists.dtx v1.0a
Remove need for drv file 879	General: Initial version, split from
1994-03-02 ltclass.dtx v0.3a	latex.dtx 861
General: Remove need for driver file 1042	Long lines wrapped to 72 columns 861
1994-03-03 ltboxes.dtx v0.1f	1994-03-07 ltpage.dtx v1.0a
\@irsbox: Replaced a missing \else 895	General: Initial version, split from
1994-03-04 ltfloat.dtx v1.0a	ltherest.dtx 1033
General: Initial version, split from	1994-03-07 ltpictur.dtx v0.1a
latex.dtx 972	General: Initial version, split from
1994-03-04 ltsect.dtx v1.0a	latex.dtx 922
General: Initial version, split from	Long lines wrapped to 72 columns 922
latex.dtx 958	1994-03-07 ltsect.dtx v1.0a
1994-03-04 lttab.dtx v1.0a	\@hangfrom: (DPC)Extra groups for
General: Initial version, split from	colour 965
latex.dtx 897	1994-03-07 lttab.dtx v1.0a
	General: Long lines wrapped to 72
	columns 897

1994-03-08 ltclass.dtx v0.3b	1994-03-13 ltfiles.dtx LaTeX2e
General: Modify driver code into ‘new style’	\@addtofilelist: Macro added . . .
1042	\listfiles: Reset \@addtofilelist at begin document
1994-03-08 ltdirchk.dtx v1.0a	473
General: Reorganize driver module into ‘new style’	1994-03-13 ltffsbas.dtx v2.1g
1	General: add 2kernel module to omit repeated code
1994-03-08 ltplain.dtx v1.0a	533
General: Remove need for a driver file. 14	1994-03-13 ltffsdcl.dtx v2.1c
1994-03-10 ltffsbas.dtx v2.2f	General: add 2kernel module to omit repeated code
\math@egroup: Changed	663
\begin@group/\endgroup to	1994-03-14 ltboxes.dtx v1.0b
\bgroupp/\egroup.	\@isavebox: Use \color@setgroup .
559	\@isavepicbox: Use
1994-03-11 ltffsdcl.dtx v2.1b	\color@setgroup
\DeclareSymbolFontAlphabet@:	\color@begin@group: macro added for color support
Added check against use of	\color@end@group: macro added for color support
alphabet switch outside of math	\lrbox: Use \color@setgroup
mode.	\sbox: Use \color@setgroup
694	1994-03-14 ltfloat.dtx 1.0c
\SetMathAlphabet@: Changed	\@xympar: (DPC) Use
parameter template in temporary	\color@begin@group
macro to catch check add below. 682	986
1994-03-12 ltclass.dtx v0.3c	1994-03-14 ltfloat.dtx v1.0c
General: Change name from docclass to ltclass	\@endfloatbox: (DPC) Use
1042	\color@end@group
\ProvidesFile: Add \wlog	\@footnotetext: (DPC) Use
1053	\color@begin@group, add
\ProvidesPackage: Add \wlog	\endgraf
1051	\@savemarbox: (DPC) Use
use \@gtempa	\color@begin@group
1051	\@xfloat: (DPC) Use
1994-03-12 ltdefs.dtx v1.0b	\color@begin@group
\@reargdef: New defn, in terms of	977
\@yargdef	1994-03-15 ltfiles.dtx LaTeX2e
85	\@missingfileerror: Quit on x or X just like a real error
\@yargdef: Name changed from	470
\XXX@argdef	1994-03-15 ltfontcmd.dtx v3.2a
85	General: Adapted to mass formatting
1994-03-12 ltdirchk.dtx v1.0b	Changed \/ to \@@italiccorr . . .
General: Change name from	Removed \@renewfontswitch
dircheck.dtx	Removed defs of short-forms and all sizes except \normalize
1	747
Minor edits to the typeouts in	747
ltxcheck	747
1	747
1994-03-12 ltfloat.dtx v1.0b	1994-03-15 ltoutput.dtx v1.0l
\@savemarbox: (DPC) Extra group for colour	\@addtocurcol: Changed \addvspace to \vskip
985	\@combinedblfloats: Removed
\@xympar: (DPC) Extra bgroup for colour	boxmaxdepth setting.
986	\@make@normalcolbox: Removed
1994-03-12 ltplain.dtx v1.0b	boxmaxdepth setting.
General: Name changed from lplain.	\@outputbox@append: \maxdepth
The end of an era	changed to \@maxdepth
14	\@topnewpage: Corrected and amended warning message
1994-03-12 ltplain.dtx v1.0e	1179
General: Replaced remaining width, height, depth by L ^A T _E X macro names to save tokens.	1182
14	1168
1994-03-13 ltcntrl.dtx v1.0c	
\@tf@for: (DPC) Add \@tf@r so a single group is correctly treated.	
396	
1994-03-13 ltfilehook.dtx v0.3b	
\unqu@tefilef@und: Use new cmd	
\@addtofilelist	
1110	

Warning added: it should be improved	1169	1994-03-28 ltthm.dtx v1.0a General: Initial version, split from latex.dtx	954
General: Added some warnings when page gets full of top floats.	1152	1994-03-29 ltcounts.dtx v1.0c General: Create file from parts of ltmiscen and ltherest.	521
Driver added and further tidying.	1152	1994-03-29 ltlength.dtx v1.0c General: Create file ltcntlen from parts of ltmiscen and ltherest.	531
Removed duplicated code and corrected docstrip options.	1152	1994-03-29 ltmiscen.dtx v1.0d General: Remove counter macros to ltcntlen	819
Some boxmaxdepth settings removed.	1152	1994-03-29 ltpageno.dtx v1.0c General: Create file ltcntlen from parts of ltmiscen and ltherest.	796
1994-03-16 ltclass.dtx v0.3f General: Add pkgindoc package	1090	1994-03-29 ltxref.dtx v1.0c General: Create file ltcntlen from parts of ltmiscen and ltherest.	797
1994-03-16 ltfiles.dtx LaTeX2e <code>\listfiles</code> : Move this code directly into <code>\document</code>	473	1994-03-31 ltbibl.dtx v1.0a General: Initial version of ltidxbib.dtx, split from ltherest.dtx	998
1994-03-16 ltfiles.dtx v1.0c <code>\document</code> : (DPC) directly add file list settings	455	1994-03-31 ltidxglo.dtx v1.0a General: Initial version of ltidxbib.dtx, split from ltherest.dtx	995
1994-03-16 ltmiscen.dtx v1.0b <code>\verbatim</code> : Remove <code>\global\@inlabelfalse</code> again.	837	1994-04-09 ltcounts.dtx v1.0d <code>\@newctr</code> : <code>\@nocnterr</code> now has counter name argument	523
1994-03-28 lttalloc.dtx v1.0d General: Redefinition of ‘new’ allocations removed.	391	<code>\addtocounter</code> : <code>\@nocnterr</code> now has counter name argument	522
1994-03-28 ltdirchk.dtx v1.0d General: Improve documentation	1	<code>\setcounter</code> : <code>\@nocnterr</code> now has counter name argument	522
1994-03-28 lterror.dtx v1.0d <code>\@invalidchar</code> : (DPC) Comment out (use <code>catcode15</code> instead)	407	<code>\stepcounter</code> : Use <code>\addtocounter</code> to have name checked	523
General: Remove test for <code>\inputlineno</code> undefined.	403	1994-04-09 ltthm.dtx v1.0b <code>\@othm</code> : Use standard counter error message (FMi)	956
1994-03-28 ltfiles.dtx v1.0d <code>\document</code> : (DPC) Use <code>\normalsize</code> not <code>\@normalsize</code>	454	1994-04-11 ltclass.dtx v0.3g <code>\endfilecontents</code> : Add star form, don’t write <code>\endinput</code> at the end of the file.	1074
(DPC) remove <code>\@normalsize check</code>	454	<code>\ProvidesFile</code> : Protect against weird catcodes.	1053
1994-03-28 ltfloat.dtx v1.0b <code>\@caption</code> : Use <code>\normalsize</code> not <code>\@normalsize</code>	975	1994-04-11 ltfssbas.dtx v2.1h General: Added <code>\defaultscipratio</code> and <code>\defaultscriptscipratio</code> . ASAJ.	533
General: Split further from ltherest.dtx	972	<code>\defaultscipratio</code> : Macro added	559
1994-03-28 ltlists.dtx v1.0b General: Improve documentation	860	<code>\defaultscriptscipratio</code> : Macro added	559
1994-03-28 ltmiscen.dtx v1.0c General: Improve Documentation	819	1994-04-12 ltboxes.dtx v1.0c General: Remove <code>\@acci</code> , now defined in lplain.dtx	889
1994-03-28 ltplain.dtx v1.0c <code>\newlanguage</code> : Remove some <code>\outer</code> declarations.	17	Remove <code>\@dischyph</code> , now defined in ltinit.dtx	889
1994-03-28 ltsect.dtx v1.0b General: Split further from ltherest.dtx	958		
1994-03-28 lttab.dtx v1.0b General: Improve documentation	897		

1994-04-12 ltdefs.dtx v1.0g	1994-04-18 ltfstrc.dtx v2.3d
\@dischyph: Define \@dischyph, was previously in ltboxes.dtx	General: Changed to new error/warning scheme
1994-04-12 ltplain.dtx v1.0d	\font@submax: Changed dimen to macro
General: Define \@acci	\fontsubfuzz: Changed dimen to macro
1994-04-12 ltvers.dtx v1.0b	\subst@size: \font@submax and \fontsubfuzz now macros
General: Have version info generated automatically.	1994-04-19 ltpage.dtx v1.0b
1994-04-14 ltfntcmd.dtx v3.2b	General: Improve documentation
General: Macros renamed to non-private forms, JB	1994-04-20 ltfntcmd.dtx v3.3a
\DeclareOldFontCommand: Renamed from \@newfontswitch	General: Documentation up-dated
1994-04-15 ltboxes.dtx v1.0d	New implementation of \nocorr
\@isavebox: Added missing percent character.	\check@nocorr@: Macros added
1994-04-17 ltcunts.dtx v1.0e	\maybe@ic@: \nocorr etc removed from list of tokens to check, leaving only punctuation characters
\@newctr: Use \@nocounterr instead of \@nocnterr	1994-04-20 ltmiscen.dtx v1.0e
\addtocounter: Use \@nocounterr instead of \@nocnterr	\@enddocument@kernel@warnings: Changed logic for producing warning messages
\setcounter: Use \@nocounterr instead of \@nocnterr	1994-04-21 ltboxes.dtx v1.0e
1994-04-17 lterror.dtx v1.0h	\@iiminipage: Extra \bgroup for color
\@nocounterr: New name for error message, old error message (without arg) kept	\@mpfootnotetext: Extra \endgraf for color
1994-04-17 ltthm.dtx v1.0c	\endminipage: Extra \egroup for color
\@othm: Use new std counter error message (FMi)	1994-04-21 ltfinal.dtx v0.1c
\@thm: Use new std counter error message (FMi)	General: Added comments, set the catcodes of 128–255.
1994-04-18 ltfinal.dtx v0.1b	1994-04-22 ltssini.dtx v2.1g
General: Initialise \textheight, \textwidth and page style	\not@math@alphabet: Message changed again
1994-04-18 ltfloat.dtx v1.0d	1994-04-23 ltfinal.dtx v0.1d
\@footnotetext: (DPC) Remove Colour support	General: Check that \font@submax is still zero
\@savemarbox: (DPC) Remove Colour support	1994-04-24 ltoutput.dtx v1.0m
1994-04-18 ltssbas.dtx v2.1i	\@resetfps: Number 2 changed to \tw@
General: Macro \no@alphabet@help removed again	Warning changed
\calculate@math@sizes: Changed message to log only	\@specialoutput: Message changed to give more info and ‘top’ removed
\no@alphabet@error: Use std LaTeX error macro	\@topnewpage: Message changed to give more info
1994-04-18 ltssdcl.dtx ???	Warning message removed as it will be generated later
\DeclareMathAlphabet: Pass correct arg (2 not 3)	General: Changed \@normalsize to \normalsize.
1994-04-18 ltssdcl.dtx v2.1d	Corrected unverbred commands in documentation.
General: Removed surplus \no@alphabet@error (see fam.dtx)	Removed some long lines and other aesthetic changes.

- Warning messages
changed/corrected. 1152
- 1994-04-24 ltpictur.dtx v0.1b
General: Removed surplus spaces after
`\hbox to` in several cases 922
- 1994-04-25 ltclass.dtx v0.3h
General: Removed spurious extra ‘s at
the end of error messages 1042
- 1994-04-25 ltfloat.dtx v1.0e
`\@largefloatcheck`: Changed warning
message to give more info 981
Command added 981
General: Changed warning messages 972
Removed obsolete tracing code . . 972
- 1994-04-27 ltfstrc.dtx v2.3e
General: Corrected item that was
forgotten in last change. 633
- 1994-04-28 lterror.dtx v1.0j
`\@inmatherr`: Macro added 407
- 1994-04-28 lterror.dtx v1.1c
`\@inmatherr`: Replaced `\noexpand`
with `\protect`. 407
- 1994-04-28 ltfsdcl.dtx v2.1e
General: Removed all `\uppercase` in
hex num parsing macros 663
- 1994-04-28 ltlists.dtx v1.0c
`\item`: Replaced `\@ltxnomath` by
`\@inmatherr` 872
- 1994-04-28 ltpictur.dtx v0.1c
`\@multiput`: (DPC) Macro added . . 926
General: bezier curves added 949
`\multiput`: (DPC) Ignore spaces
between `)`(. 925
`\picture`: (DPC) Ignore spaces before
(. 924
- 1994-04-28 ltplain.dtx v1.0g
General: Turn off overfull box tracing
in log 25
- 1994-04-29 ltclass.dtx v1.0a
General: Change version number to 1
(no other change) 1042
- 1994-04-29 ltmiscen.dtx v1.0f
`\@verbatim`: `\leavevmode` added . . 837
Change to `\everypar` added 837
- 1994-04-29 ltoutenc.dtx 1.4a
General: Removed
`\EncodingSpecific`. Renamed all
the commands. Added
`\DeclareTextGlyph` and
`\UndeclareTextCommand`. 479
Removed Rokicki’s OT1 variant
encoding. Moved the driver to the
top. 478
- 1994-04-30 ltfntcmd.dtx v3.3b
General: Documentation up-dated and
tidied 747
Prefix `frag@` changed to `frag` in
`\@protecteddef` 747
Title changed 747
Warning changed to info message in
`\@protecteddef` 747
- 1994-04-30 ltoutput.dtx v1.0n
`\@activechar@info`:
`\@activechar@warning` changed to
`\@activechar@info` 1188
`\@combinedblfloats`: Removed rule in
topnewpage case 1195
`\@emptycol`: Empty column action
added: `\@emptycol` 1167
`\@flsetnum`: Rogue space removed 1231
`\@specialoutput`: Cut-off point
changed to `2\baselineskip` . . . 1172
Empty column action added:
`\@emptycol` 1172
Extra empty column added for
twocolumn case 1172
Extra empty column added for
twocolumn case (wrong, see
below) 1172
`\@topnewpage`: Added setting of
`\col@number` 1167, 1168
Cut-off point changed to
`3\baselineskip` 1169
Empty column action added:
`\@emptycol` 1169
Message changed for Frank 1169
General: `\@activechar@warning`
changed to an info message. . . . 1152
Added `\col@number`. 1152
Documentation tidied. 1152
Empty column action added. 1152
Fixed bug from `\dblfigrule` with
`\@topnewpage`. 1152
Full of floats action improved. . . 1152
`\col@number`: Added `\col@number` 1164
`\onecolumn`: Added setting of
`\col@number` 1166
- 1994-05-01 lterror.dtx v1.0k
`\@latexerr`: (CAR) Added draft
`\@latexinfo`. 403
- 1994-05-01 ltoutenc.dtx 1.4a
General: Added the `\a` command. . . 488
Added the `\SaveAtCatcode` and
`\RestoreAtCatcode` commands. . . 491
Removed the uc/lc table settings,
since the T1 uc/lc table is now the
default. 499

Rewrote for the new syntax.	491, 494	<code>\end@float</code> : (CAR) Added	
1994-05-01 ltoutenc.dtx v1.4a		<code>\@largefloatcheck</code>	979
General: Removed Rokicki's		1994-05-03 ltfsdcl.dtx v2.1f	
encoding.	475	General: Renamed	
Renamed the commands, removed		<code>\@DeclareMathDelimiter</code> to	
the <code>\EncodingSpecific</code> command.		<code>\DeclareMathDelimiter</code>	663
Turned all slots into decimal.		1994-05-03 ltlists.dtx v1.0d	
Added <code>\a</code>	475	<code>\@item</code> : <code>\hskip</code> changed to <code>\kern</code> . .	873
1994-05-02 ltcntrl.dtx v1.0l		<code>\item</code> : Removed superfluous braces	872
<code>\@break@tfor</code> : Macro added (from		1994-05-03 ltmiscen.dtx v1.0h	
ltfiles.dtx)	396	<code>\@centercr</code> : <code>\@badcrerr</code> replaced by	
1994-05-02 ltdefs.dtx v1.1f		<code>\@nolnerr</code>	833
<code>\renewcommand</code> : Removed surplus		1994-05-03 lttab.dtx v1.0d	
<code>\space</code> in error	86	<code>\@endpbox</code> : Use <code>\@finalstrut</code> based	
<code>\renewenvironment</code> : Removed surplus		on depth of <code>\@arstrutbox</code>	920
<code>\space</code> in error	87	1994-05-04 ltclass.dtx v1.0b	
1994-05-02 ltfiles.dtx v1.0f		<code>\NeedsTeXFormat</code> : Changed wording of	
<code>\iffileonpath</code> : <code>\@break@loop</code>		the warning	1062
renamed to <code>\@break@tfor</code>	468	1994-05-04 lterror.dtx v1.0m	
<code>\@obsoletefile</code> : Make		<code>\@badcrerr</code> : Error message removed	406
<code>\@onlypreamble</code>	472	1994-05-05 ltbibl.dtx v1.0c	
1994-05-02 ltfinal.dtx v0.1e		<code>\@citex</code> : Set switch for warning and	
General: Added setting the 'letter'		end of run.	1000
catcodes.	1272	<code>\@nocite</code> : Do not write page number in	
Added setting the 'other'		<code>\@nocite</code> warning message.	1001
catcodes.	1272	Set switch for warning and end of	
Added setting the special		run.	1001
catcodes.	1272	1994-05-05 ltfinal.dtx v0.1g	
Made slot 127 illegal	1272	General: Added empty errhelp.	1260
Set all the catcodes	1260	<code>\errhelp</code> : Set error help empty. . .	1277
1994-05-02 ltfinal.dtx v0.1f		1994-05-05 ltfntcmd.dtx v3.3c	
General: Set the catcode of		<code>\@math@egroup</code> : Corrected	
control-J.	1272	<code>\@fontswitch</code> and added saved	
1994-05-02 ltmiscen.dtx v1.0g		versions	754
General: Changed 91 to 1991 and		General: Corrected <code>\@fontswitch</code> . .	747
moved some bits	819	1994-05-05 ltmiscen.dtx v1.0i	
1994-05-02 ltoutput.dtx v1.0o		General: Removed braces from	
<code>\@resethfps</code> : Code shortened	1230	<code>\ifnextchar</code> and <code>\ifstar</code> arguments .	819
General: Code of <code>\@resethfps</code>		1994-05-07 lttab.dtx v1.0c	
shortened.	1152	<code>\@maxtab</code> : Changed <code>\@firsttab</code> to	
1994-05-03 ltbibl.dtx v1.0b		<code>\chardef</code>	901
<code>\@nocite</code> : Make <code>\@nocite</code> issue a		Changed <code>\@maxtab</code> to <code>\chardef</code> . .	901
warning for an undefined citation		General: Removed definition of <code>\+</code> .	897
key.	1001	Removed surplus braces from	
1994-05-03 ltfinal.dtx v0.1f		<code>\@ifnextchar</code> constructs	897
General: Set the catcode of control-J		1994-05-08 ltfntcmd.dtx v3.3d	
to be 'other', for use in messages.	1260	General: Removed	
1994-05-03 ltfloat.dtx v1.0f		<code>\@undefinedfonterror</code>	747
General: (CAR) Added		<code>\@normalsize</code> : Removed	
<code>\@largefloatcheck</code>	972	<code>\@undefinedfonterror</code>	755
Removed unnecessary braces from		1994-05-09 ltfntcmd.dtx v3.3f	
arguments of <code>\@ifnextchar</code>	972	General: Replaced all <code>\next</code> by	
<code>\end@dblfloat</code> : <code>\@largefloatcheck</code>		<code>\@let@token</code> and undo change	
added	980	3.3e, whatever that was.	747

1994-05-10 ltxdefs.dtx v1.0n			
General: (ASAJ) Added		<code>\@changed@cmd</code> and	
<code>\DeclareProtectedCommand</code>	80	<code>\DeclareProtectedCommand</code>	479
Added <code>\DeclareProtectedCommand</code>	90	Renamed the commands again.	
Removed braces around		Made the encoding part of the	
<code>\@ifundefined</code> argument. ASAJ.	86	command syntax. Added the	
<code>\makeatother</code> : Added <code>\makeatletter</code>		<code>\DeclareTextCommand</code> interface.	
and <code>\makeatother</code> ASAJ.	110	Used	
1994-05-10 lterror.dtx v1.0n		<code>\DeclareProtectedCommand</code>	475
<code>\@latexerr</code> : (ASAJ) Added extra		<code>\DeclareTextAccent</code> : Reimplemented	
blank lines to <code>\@latexerr</code>	403	using <code>\DeclareTextCommand</code>	482
1994-05-10 ltmiscen.dtx v1.0j		1994-05-11 ltspc.dtx v1.0o	
<code>\@sverb</code> : Slight change in error		<code>\hspace</code> : Use	
message text.	840	<code>\DeclareRobustCommand</code> . ASAJ.	448
1994-05-11 ltboxes.dtx v1.0f		1994-05-12 ltboxes.dtx v1.0g	
<code>\@begin@tempboxa</code> : Use new		<code>\@finalstrut</code> : macro added	895
<code>\color@setgroup</code> concept.	880	<code>\fbox</code> : New definition, merged with	
<code>\@iiminipage</code> : Use new		<code>\framebox</code>	885
<code>\color@setgroup</code> concept.	891	<code>\framebox</code> : Merged <code>\fbox</code> and	
<code>\@mpfootnotetext</code> : Use new		<code>\framebox</code>	886
<code>\color@setgroup</code> concept.	892	<code>\normalcolor</code> : macro added for color	
Use new <code>\normalcolor</code> and		support	882
<code>\@finalstrut</code>	892	1994-05-12 ltxdefs.dtx v1.0p	
General: Superfluous braces removed		General: (ASAJ) Fixed a bug with	
from several commands	879	<code>\relax</code> which was using <code>\@gobble</code>	
<code>\color@setgroup</code> : macro added for		before defining it.	80
color support	882	Fixed a bug with <code>\relax</code> which	
<code>\endminipage</code> : Use new		was using <code>\@gobble</code> before defining	
<code>\color@setgroup</code> concept.	892	it.	90
1994-05-11 ltclass.dtx v1.0c		1994-05-12 ltfssbas.dtx v2.1j	
<code>\endfilecontents</code> : Add checks for		General: New <code>baselinesstretch</code> concept	533
form feed and tab	1074	Replaced hand-protected commands	
1994-05-11 ltdirchk.dtx v1.0e		by <code>\DeclareRobustCommand</code> defs	533
General: Add <code>\ProvidesFile</code> as used		<code>\f@linespread</code> : New macro	545
in fd files.	4	<code>\fontencoding</code> : Use	
1994-05-11 lterror.dtx v1.0o		<code>\DeclareRobustCommand</code>	543
<code>\@latexerr</code> : (ASAJ) Removed one of		<code>\fontfamily</code> : Use	
the extra blank lines to		<code>\DeclareRobustCommand</code>	544
<code>\@latexerr</code>	403	<code>\fontseries</code> : Use	
1994-05-11 ltlogos.dtx v1.0o		<code>\DeclareRobustCommand</code>	544
<code>\LaTeX</code> : Use		<code>\fontshape</code> : Use	
<code>\DeclareProtectedCommand</code> .		<code>\DeclareRobustCommand</code>	544
ASAJ.	450	<code>\fontsize</code> : Redefined to use	
<code>\LaTeXe</code> : Use		<code>\set@fontsize</code>	545
<code>\DeclareProtectedCommand</code> .		<code>\linespread</code> : New macro	545
ASAJ.	450	<code>\mathversion</code> : Use	
1994-05-11 ltoutenc.dtx 1.5a		<code>\DeclareRobustCommand</code>	546
General: Made T1 and OT1 generate		1994-05-12 ltfssdcl.dtx v2.1g	
packages rather than def files.		General: Allow <code>\relax</code> as undefined	
Renamed the ‘package’ module to		command	663
‘teststy’.	478	Allow <code>\relax</code> ’ed cmds to be	
1994-05-11 ltoutenc.dtx v1.5a		declared	663
General: Reimplemented		1994-05-12 ltfssini.dtx v2.1i	
<code>\DeclareTextCommand</code> using		General: Moved <code>\fontencoding</code> to	
		fam.dtx	695

Moved <code>\fontfamily</code> to fam.dtx	695	1994-05-13 ltfiles.dtx v1.0g	
Moved <code>\fontseries</code> to fam.dtx	695	<code>\document</code> : Added execution of	
Moved <code>\fontshape</code> to fam.dtx	695	<code>\every@size</code>	454
Moved <code>\fontsize</code> to fam.dtx	695	1994-05-13 ltfinal.dtx v0.1h	
Moved <code>\mathversion</code> to fam.dtx	695	General: Added package <code>otlenc</code> , and	
Moved <code>\selectfont</code> to tracefnt.dtx	695	defined <code>\@accii</code> , <code>\@acciii</code> and	
1994-05-12 ltfsstrc.dtx v2.3f		<code>\@acciii</code>	1260
<code>\selectfont</code> : Use		1994-05-13 ltfinal.dtx v1.0h	
<code>\DeclareRobustCommand</code>	637	General: Added output enc stuff . . .	1276
1994-05-12 ltoutenc.dtx 1.5a		1994-05-13 ltfloat.dtx v1.0g	
General: Removed the		<code>\@footnotetext</code> : (DPC) Add new	
<code>\SaveAtCatcode</code> and		style colour support:	
<code>\RestoreAtCatcode</code> commands.	491	<code>\normalcolor</code>	991
Rewrote for the new syntax.	491, 494	(DPC) Use <code>\@finalstrut</code>	991
1994-05-12 ltoutput.dtx v1.0p		<code>\xfloat</code> : (DPC) Use <code>\normalcolor</code>	977
<code>\writersetup</code> : <code>\normalcoloradded</code>	1188	1994-05-13 ltfntcmd.dtx v3.3g	
General: <code>\normalcoloradded</code> in		General: Replaced <code>\@protecteddef</code> by	
various places (DPC).	1152	<code>\DeclareRobustCommand</code>	747
1994-05-13 ltboxes.dtx v1.0h		1994-05-13 ltfsbas.dtx v2.1k	
<code>\@arrayparboxrestore</code> : New accent		General: Remove File identification	
system, use <code>\let</code> not <code>\def</code>	890	‘typeout’	533
1994-05-13 ltcounts.dtx v1.0f		1994-05-13 ltfsbas.dtx v2.1l	
General: Removed <code>\@Ialph</code>	528	<code>\DeclareFontEncoding</code> : Init encoding	
Removed <code>\@ialph</code>	528	change command	537
1994-05-13 ltdefs.dtx v1.0q		<code>\define@newfont</code> : Use <code>\@input@</code> for fd	
General: (ASAJ) Renamed		files	550
<code>\DeclareProtectedCommand</code> to		1994-05-13 ltssdcl.dtx v2.1h	
<code>\DeclareRobustCommand</code> .		General: Removed file identification	
Removed <code>\@if@short@command</code>	80	typeout	663
(ASAJ) Replaces <code>\space</code> by ‘ ’ in		1994-05-13 ltssini.dtx v2.1j	
<code>\csname</code>	80	General: Removed file identification	
Renamed		typeout	695
<code>\DeclareProtectedCommand</code> to		1994-05-13 ltfsstrc.dtx v2.3g	
<code>\DeclareRobustCommand</code> .		General: Removed typeouts as	
Removed <code>\@if@short@command</code> .		<code>\ProvidesPackage</code> writes to log.	633
Moved to after the definition of		1994-05-13 ltoutenc.dtx v1.5b	
<code>\gobble</code>	90	General: Added <code>\{</code> , <code>\}</code> and <code>\\$</code>	475
1994-05-13 ltdefs.dtx v1.0r		Renamed	
General: (ASAJ) Added logging		<code>\DeclareProtectedCommand</code> to	
message to		<code>\DeclareRobustCommand</code>	475
<code>\DeclareProtectedCommand</code>	80	Replaces <code>\space</code> by ‘ ’ in <code>\csname</code>	475
Added logging message to		1994-05-13 ltpictur.dtx v0.1d	
<code>\DeclareProtectedCommand</code>	90	General: Removed surplus braces from	
1994-05-13 ltdefs.dtx v1.0s		<code>\if..</code> constructions	922
General: (ASAJ) Added		1994-05-13 lttab.dtx v1.0d	
<code>\@backslashchar</code>	80	<code>\@contfield</code> : Colour support	903
(ASAJ) Coded <code>\@ifdefinable</code> more		<code>\@startfield</code> : Colour support	903
efficiently.	80	<code>\@stopfield</code> : Colour support	903
Coded more efficiently, thanks to		<code>\a</code> : moved to ltoutenc	901
FMi.	86	1994-05-14 fontdef.dtx v2.1f	
1994-05-13 ltfiles.dtx LaTeX2e		General: Removed <code>.def</code> files.	725
<code>\listfiles</code> : Stop <code>\listfiles</code> being		1994-05-14 ltfsbas.dtx v2.1m	
run twice	472	<code>\enc@update</code> : Macro added	544

- 1994-05-14 ltffsbas.dtx v2.1n
 General: Set defaults for all `\f@...` 545
`\DeclareErrorFont`: Don't set
`\f@encoding` 553
`\DeclareFontEncoding`: Log if
 encoding is redeclared 537
 Only init enc change cmd when new
 encoding 537
- 1994-05-14 ltffsini.dtx v2.1k
 General: Init error font just before
 checking for fontdef.cfg 720
`\reset@font`: Remove surplus braces 718
- 1994-05-14 ltfstrc.dtx v2.3h
`\selectfont`: Added `\enc@update` . 639
- 1994-05-14 ltoutenc.dtx 1.5d
 General: Moved the driver to the top. 478
- 1994-05-14 ltoutenc.dtx v1.5c
 General: Added the fontenc package 518
 Added the fontenc package. 475
 Fixed a bug which caused an
 infinite loop if `\f@encoding` was
 incorrectly set. 475, 479
 Moved fontsmpl to its own dtx file. 475
- 1994-05-14 ltoutenc.dtx v1.5d
 General: Rewrote
`\DeclareTextCommand` to define its
 argument to use the current
 encoding by default, rather than
 the encoding provided to
`\DeclareTextCommand`. ... 475, 479
 Tidied up the documentation. ... 475
- 1994-05-14 ltoutenc.dtx v1.5e
 General: Replaced `\ENC@cmd` by
`\ENC-cmd`. 475
- 1994-05-15 ltffsbas.dtx v2.1o
 General: encoding cmds changed to
 enc-cmd 533
- 1994-05-16 fontdef.dtx v2.1g
 General: Removed
`\DeclareFontEncoding` for ot1 and
 t1 and input .def files instead .. 725
- 1994-05-16 ltalloc.dtx v1.1a
 General: (ASAJ) Split from ltinit.dtx. 391
- 1994-05-16 ltcntrl.dtx v1.0a
 General: (ASAJ) Split from ltinit.dtx. 393
- 1994-05-16 ltdefns.dtx v1.1a
 General: (ASAJ) Split from ltinit.dtx. 80
- 1994-05-16 lterror.dtx v1.1a
 General: (ASAJ) Completely new
 error interface. 397
 (ASAJ) Split from ltinit.dtx. ... 397
- 1994-05-16 ltfinal.dtx v1.0i
 General: moved output enc stuff to
 lfonts 1276
- 1994-05-16 ltffsbas.dtx v2.1p
`\fontsize`: Pass `\baselinestretch` not
`\f@linespread` 545
`\linespread`: Remove surplus braces 545
- 1994-05-16 ltffsini.dtx v2.1m
`\@acciii`: Define saved versions of
 accents 721
- 1994-05-16 ltlogos.dtx v1.1a
 General: (ASAJ) Split from ltinit.dtx. 450
- 1994-05-16 ltmath.dtx v1.0k
`\ensuremath`: Use
`\DeclareRobustCommand` and add
 extra braces in math mode 856
- 1994-05-16 ltoutenc.dtx 1.5h
 General: `\pounds` was still using u
 rather than ui shape. 491
- 1994-05-16 ltoutenc.dtx v1.5f
 General: enc files now have uc
 encoding name parts (FMi) 475
 Revert code so that the encoding
 given is used in
`\DeclareTextCommand` (FMi) ... 475
- 1994-05-16 ltoutenc.dtx v1.5g
 General: Made fontenc.sty use the new
 mixed-case encoding files. 475
 Removed the lowercasing of the
 filename. 518
- 1994-05-16 ltoutenc.dtx v1.5h
 General: Added `\NG`, `\ng`, `\TH`, `\th`,
`\DH`, `\dh`, `\DJ` and `\dj`. 475
 Added `\r` (ring accent) and `\k`
 (ogonek) accents. 475
 Fixed a bug with `\pounds`. 475
 Removed `\P` from the OT1
 definitions file. 475
- 1994-05-16 ltoutenc.dtx v1.5i
 General: Fixed a bug with `\d`. 475
- 1994-05-16 ltoutput.dtx v1.0q
`\@writsetup`: Changed setting of
 accents (FMi): with the new
 encoding setup they can use `\let`.
 It could also use the new internal
 commands? 1189
 General: Changed setting of accents
 (FMi). 1152
- 1994-05-16 ltpar.dtx v1.1a
 General: (ASAJ) Split from ltinit.dtx. 408
- 1994-05-16 ltplain.dtx v1.0h
 General: Comment out encoding
 specific commands 31
 Remove `\@acci` and friends again . 32
 Remove unnecessary def for `\item` 31
`\loop`: Use Kabelschacht method ... 29
`\m@th`: Remove unnecessary space ... 31

1994-05-16 ltspace.dtx v1.1a		Replaced <code>\defaultencoding</code> with	
General: (ASAJ) Split from ltinit.dtx.	430	<code>\encodingdefault</code> .	475
1994-05-17 ltclass.dtx v1.0e		1994-05-19 ltbibl.dtx v1.1a	
<code>\@use@option</code> : Execute option after		General: Initial version of ltbibl.dtx,	
removing from list, not before	1058	split from ltidxbib.dtx	998
1994-05-17 ltdefs.dtx 1.1b		1994-05-19 ltcounts.dtx v1.1a	
General: (ASAJ) Added the		General: Extracted file from ltcntlen.	521
<code>\@protect@...</code> commands.	90	1994-05-19 ltdefs.dtx v1.1d	
1994-05-17 ltdefs.dtx v1.1b		General: (RmS) Added definitions for	
General: (ASAJ) Added definitions for		<code>\@namedef</code> and <code>\@nameuse</code> again.	80
<code>protect</code> .	80	1994-05-19 ltfinal.dtx v0.1k	
(ASAJ) Removed warnings and		General: Removed <code>\makeat...</code>	1260
logging to lterror.dtx.	80	1994-05-19 ltidxglo.dtx v1.1a	
Added the discussion of protected		General: Initial version of ltidxglo.dtx,	
commands, defined the values that		split from ltidxbib.dtx	995
<code>\protect</code> should have.	90	1994-05-19 ltlength.dtx v1.1a	
1994-05-17 ltdefs.dtx v1.1c		General: Extract file ltlength from	
General: (ASAJ) Redid definitions for		<code>ltcntlen</code> .	531
<code>protect</code> .	80	1994-05-19 ltpageno.dtx v1.1a	
1994-05-17 lterror.dtx v1.1b		General: Extract file ltpageno from	
General: (ASAJ) Moved error stuff		<code>ltcntlen</code> .	796
from ltdefs.dtx.	397	1994-05-19 ltplain.dtx v0.1k ltfinal	
1994-05-17 ltfssini.dtx v2.1n		<code>\showoutput</code> : used <code>\maxdimen</code> not	
<code>\copyright</code> : Really add extra braces	718	<code>99999</code>	33
<code>\nfss@text</code> : Added braces to allow		<code>\showoverfull</code> : used <code>\@ne</code> not 1	33
use in subscripts	718	1994-05-19 ltxref.dtx v1.1a	
1994-05-17 ltmath.dtx v1.0i		General: Extract file ltxref from	
General: Replaced <code>\let</code> by <code>\gdef</code> , for		<code>ltcntlen</code> .	797
indirect definition.	851	1994-05-20 ltdefs.dtx v1.1e	
1994-05-17 ltoutenc.dtx v1.5j		General: Changed command name	
General: Added braces to <code>\pounds</code> so		from <code>\@checkcommand</code> to	
it works as a subscript.	475	<code>\CheckCommand</code> .	80
1994-05-18 ltdefs.dtx 1.1c		<code>\CheckCommand</code> : Changed name from	
General: (ASAJ) Renamed the		<code>\@checkcommand</code> to	
commands, and removed one		<code>\CheckCommand</code> .	88
which is no longer needed.	90	1994-05-20 lterror.dtx v1.1c	
1994-05-18 ltdefs.dtx v1.1c		General: (ASAJ) Added	
General: Redid the discussion and		<code>\@latex@info@no@line</code> .	397
definitions, in line with the		(ASAJ) Added missing full stops.	397
proposed new setting of <code>\protect</code>		(ASAJ) Fixed a bug with	
in the output routine.	90	<code>\@inmatherr</code> .	397
1994-05-18 ltfinal.dtx v0.1j		1994-05-20 ltfinal.dtx v0.1l	
General: Corrected the lccode for		General: Use new font warning	
<code>d-bar</code> .	1260	commands	1267
1994-05-18 ltlogos.dtx v1.1b		1994-05-20 ltfloat.dtx v1.0h	
General: (ASAJ) Added the \TeX		<code>\@endfloatbox</code> : Restore outer value of	
logo.	450	<code>@nobreak</code> switch.	981
(ASAJ) Made the $\LaTeX 2\epsilon$ logo use		<code>\outer@nobreak</code> : Macro added:	
the text font ‘2’ rather than the		default is to do nothing.	981
math font ‘2’.	450	1994-05-20 ltfntcmd.dtx v3.3h	
1994-05-18 ltoutenc.dtx v1.5k		General: Use new error commands	747
General: Made dotted-i produce ‘i’.	475	1994-05-20 ltfssbas.dtx v2.1q	
Removed braces from <code>\pounds</code> and		General: Use new error commands	533
<code>\dollar</code> .	475		

1994-05-20 ltfstrc.dtx v2.3i	1994-05-22 lterror.dtx v1.2a
General: Use new error command	General: (ASAJ) Made
names 633	\GenericError, \GenericWarning
1994-05-20 ltmiscen.dtx v1.0l	and \GenericInfo robust. 397
\@writefile: Added correct setting of	(ASAJ) Replaced \ and tilde by
\protect. 826	\MessageBreak and \space. 397
1994-05-20 ltmiscen.dtx v1.0m	(ASAJ) Replaced
General: Use new warning commands 819	\@generic@message and
1994-05-20 ltoutput.dtx v1.0s	\@generic@error by
\@writsetup: Added setting of	\GenericError, \GenericWarning
\protect during \shipout. 1188	and \GenericInfo. 397
General: Added setting of \protect	(ASAJ) Replaces \string by
during \shipout. 1152	\protect in some messages. 397
1994-05-20 ltpage.dtx v1.0d	1994-05-22 lterror.dtx v1.2d
\markright: Changed setting for	\GenericError: (DPC) Alternative
\protect. 1034	version added for old TeXs 398
1994-05-20 ltsect.dtx v1.0c	(DPC) New version using long
\addcontentsline: Correct setting of	command name. 398
\protect. 967	1994-05-22 ltfloating.dtx v1.0i
\addtocontents: Correct setting of	General: Use new warning commands 972
\protect. 968	1994-05-22 ltoutput.dtx v1.0t
1994-05-21 ltbibl.dtx v1.1b	General: Changed warnings and infos
General: Use new warning commands 998	to new commands. 1152
1994-05-21 lterror.dtx v1.1d	1994-05-22 ltpictur.dtx v0.1e
General: (ASAJ) Made the error	General: Use new warning cmds 922
commands robust. 397	1994-05-23 ltclass.dtx v1.0h
1994-05-21 ltfiles.dtx v1.0h	\NeedsTeXFormat: Don't stop
General: Use new error commands 451	completely when format is wrong 1062
1994-05-21 ltlists.dtx v1.0f	\usepackage: Remove argument if
General: Use new error commands 860	possible 1061
1994-05-21 ltmiscen.dtx v1.0n	1994-05-23 ltdirchk.dtx v1.0f
General: Use new error commands 819	General: Document \@TeXversion 1
1994-05-21 ltsect.dtx v1.0d	1994-05-23 ltfstrc.dtx v2.3j
General: Use new error commands 958	General: Removed def of
1994-05-21 lttab.dtx v1.0f	\@warn@break 652
General: Use new error commands 897	1994-05-23 ltoutput.dtx v1.0u
1994-05-21 ltxref.dtx v1.1b	\@activechar@info: Added
General: Use new warning commands 797	\MessageBreak 1188
\newlabel: Use new warning	\@writsetup: Changed resetting of
commands 800	\protect after shipout to use
1994-05-22 ltclass.dtx v1.0f	\aftergroup 1188
General: Use new warning and error	General: Added \MessageBreak. 1152
commands 1038	Changed resetting of \protect after
1994-05-22 ltdefs.dtx v1.1f	shipout. 1152
General: Use new warning and error	1994-05-24 lterror.dtx v1.2e
cmds 80	\@latex@info@no@line: Macro added 401
1994-05-22 lterror.dtx v1.1e	1994-05-24 lterror.dtx v1.2f
General: (ASAJ) Replaced bgroup by	General: (DPC) wrap long lines 397
begingroup in error messages, to	1994-05-24 ltfntcmd.dtx v3.3i
stop extra mathords creeping into	General: Tidying and typos fixed 747
math mode. 397	1994-05-24 ltmiscen.dtx v1.0q
	\@currrentline: Use \@empty as outer
	default 832

1994-05-25 ltdirchk.dtx v1.0g	<code>\filename@parse</code> : Mac parser had "		
	typo for :	12	
1994-05-25 ltfntcmd.dtx v3.3j	General: Insertion of <code>\aftergroups</code> to implement <code>\nocorr</code> moved to the end of the group	747	
	<code>\check@icr</code> : Macros added	751	
	<code>\check@nocorr@</code> : Insertion of <code>\aftergroups</code> moved and defaults set up for efficiency	751	
	<code>\DeclareTextFontCommand</code> : <code>\expandafter</code> inserted	749	
	Insertion of <code>\aftergroups</code> moved	749	
1994-05-25 ltoutput.dtx v1.0v	General: Extra documentation.	1152	
1994-05-25 ltsect.dtx v1.0e	<code>\dottedtocline</code> : Put braces around argument 4 (the actual toc entry) to avoid font (and possibly other) changes leaking out to the leaders.	970	
1994-05-25 ltthm.dtx v1.0c	General: Modify documentation	954	
1994-05-25 ltvers.dtx v1.0d	General: Remove PRELIMINARY TEST RELEASE from startup banner (spring is here)	37	
1994-05-25 ltxref.dtx v1.1c	General: Modify documentation	797	
1994-05-26 ltfiles.dtx LaTeX2e	<code>\missingfileerror</code> : Modify message format	470	
1994-05-26 ltlogos.dtx v1.1c	General: Remove <code>\SLiTeX</code> logo	450	
1994-05-26 ltmiscen.dtx v1.0r	General: <code>\literal</code> removed	843	
1994-05-26 ltplain.dtx v1.1m	<code>\iterate</code> : (CAR) added <code>\long</code>	29	
	<code>\underbar</code> : (CAR/FMi) changed to use box <code>\tw@</code>	31	
1994-05-26 ltplain.dtx v1.1p	<code>\underbar</code> : (DPC) changed to use <code>\sbox</code>	31	
1994-05-29 ltfssdcl.dtx v2.1j	General: Use new error commands	663	
1994-05-31 ltfinal.dtx v1.0n	General: Renamed <code>lthyphen.*</code> to <code>lthyphen.*</code>	1260	
1994-06-01 ltboxes.dtx v1.0i	<code>\@frameb@x</code> : Macro added.	887	
	<code>\@iframebox</code> : New version, so <code>\width</code> is correct in <code>\framebox</code>	886	
	<code>\fbox</code> : New version, using <code>\@frameb@x</code>	885	
	<code>\framebox</code> : New version, so <code>\width</code> is correct in <code>\framebox</code>	886	
1994-06-01 ltlogos.dtx v1.1d	<code>\LaTeX</code> : Add <code>\m@th</code> to force math size calculations	450	
1994-06-01 ltoutput.dtx v1.0w	General: Tidied up typesetting.	1152	
1994-06-08 ltfinal.dtx v1.0m	General: Add patch file system	1276	
1994-06-09 ltfinal.dtx v1.0n	General: For \TeX 2, do not set codes for higher half of character table.	1265, 1273	
1994-06-09 ltfntcmd.dtx v3.3k	General: Tidying and typos fixed in documentation	747	
1994-06-18 ltfntcmd.dtx v3.3l	General: Added check for empty text <code>\check@nocorr@</code> : Added check for empty text	747 751	
1994-06-22 ltfntcmd.dtx v3.3m	General: Removed space from <code>\nfss@text</code>	747	
	Renamed <code>\check@nocorr</code>	747	
	<code>\check@nocorr@</code> : Renamed <code>\check@nocorr</code> to <code>\text@command</code> to improve <code>\long</code> error message	751	
	<code>\DeclareTextFontCommand</code> : Removed space from <code>\nfss@text</code>	749	
1994-06-22 ltmath.dtx v1.2t classes	<code>\mathindent</code> : Set <code>\mathindent</code> at the end of the class instead of at begin document	857	
1994-07-20 ltlogos.dtx v1.1e	<code>\LaTeX</code> : Save a few tokens	450	
	<code>\LaTeXe</code> : Save a few tokens	450	
1994-07-20 ltpage.dtx v1.0h	<code>\sloppy</code> : Save a few tokens	1037	
1994-09-16 ltfssbas.dtx v2.1s	<code>\nfss@catcodes</code> : Reset [and] as well, just in case	551	
1994-10-07 ltoutenc.dtx v1.5l	General: Moved the ogonek accent.	475	
1994-10-11 ltdirchk.dtx v1.0h	<code>\@TeXversion</code> : Check for TeX3.14	13	
	General: Modify all of <code>ltxcheck</code> again	13	
1994-10-12 ltsect.dtx v1.0f	General: Doc. typos	958	
1994-10-14 fontdef.dtx v2.2a	General: New coding	723	
1994-10-14 ltfssini.dtx v2.2a	General: New coding for <code>cfg</code> files	695	
1994-10-14 ltmiscen.dtx v1.0s	General: Move math to other file	819	

1994-10-14 ltplain.dtx v1.1a	1994-10-25 fontdef.dtx v2.2c
General: Moved code to other files. . . 14	General: Added OMSenc.def 725
1994-10-15 ltffsbas.dtx v2.1t	1994-10-25 ltboxes.dtx v1.0l
$\extract@alph@from@version$: Warn	$\@isavepicbox$: missing percent
if math alpha is used outside	(moved from ltpatch) 884
math 559	1994-10-25 ltdefs.dtx v1.2b
1994-10-18 ltboxes.dtx v1.0j	General: Documentation
$\@frameb@x$: \leavevmode added . . 887	improvements 80
$\@iframebox$: \leavevmode moved to	1994-10-25 ltoutenc.dtx 1.6a
$\@frameb@x$ 886	General: Added \textdollar ,
$\@parboxto$: Macro added to remove	\textlbrace , \textrbrace ,
misuse of $\@empty$ 888	\textsterling , \textunderline . 494
General: stuff from ltpatch done . . . 879	Removed \textlbrace ,
\fbox : \long added 885	\textrbrace , \textunderline to
\mbox : \long added 880	give them their proper names. . . 494
\sbox : \long added 884	1994-10-25 ltoutenc.dtx v1.6a
1994-10-18 ltclass.dtx v1.0j	General: Added
General: Move \listfiles to	\ProvideTextCommand ,
ltfiles.dtx 1038	\UseTextSymbol , \UseTextAccent ,
1994-10-18 ltdefs.dtx v1.2a	$\DeclareTextSymbolDefault$,
$\@star@or@long$: macro added 83	$\DeclareTextAccentDefault$,
General: Add extra test for \endgraf 80	$\DeclareTextCommandDefault$,
Add star-forms for all commands . 80	and
$\renew@environment$: reset end	$\ProvideTextCommandDefault$. . . 475
command 87	Added the \Provide commands,
1994-10-18 ltfiles.dtx v1.0i	and the default definitions. . . . 479
\listfiles : code moved here from	Added the defaults. 488
ltclass 472	Added the files OTlenc.def,
1994-10-18 ltoutenc.dtx v1.5l	Tlenc.def and OMSenc.def. 488
General: Added new definitions of	Added the OMS encoding. 499
\patterns and \hyphenation . . . 487	1994-10-27 ltoutenc.dtx 1.6b
1994-10-18 ltoutenc.dtx v1.5m	General: Added \textasciicircum
General: Added new definitions of	\textasciitilde \textbackslash
\patterns and \hyphenation . . . 475	\textbar \textbraceleft
1994-10-18 ltsect.dtx v1.0g	\textbraceright
$\@dottedtocline$: Added	\textcompwordmark \textemdash
\normalcolor for page number . . 970	\textendash \textexclamdown
General: Added \normalcolor 958	\textgreater \textthyphenchar
1994-10-19 ltffsbas.dtx v2.1t	\textthyphen \textless
\DeclareFontEncoding : Add missing	\textquestiondown
\relax 537	\textquotedblleft
1994-10-23 ltfsstrc.dtx v23.k	\textquotedblright
$\every@math@size$: Renamed to	\textquotedbl \textquoteleft
$\every@math@size$ 642	\textquoteright
1994-10-23 ltmath.dtx v1.0l	\textunderscore
$\@eqnnum$: Added \normalcolor since	\textvisiblespace 494
\eqno introduces a subgroup of the	Added: \textemdash \textendash
displayed math group 853	\textexclamdown
\ensuremath : Remove extra braces:	\textthyphenchar \textthyphen
but see p 168 of Leslie's book . . 856	\textquestiondown
1994-10-24 ltboxes.dtx v1.0k	\textquotedblleft
\fbox : Inner braces added (to fix	\textquotedblright
latex/1061) 885	\textquoteleft
	\textquoteright 491

- 1994-10-27 ltoutenc.dtx v1.5d
 General: Rewrote
 $\backslash\text{DeclareTextSymbol}$ to define its argument to use the current encoding by default, to fit with
 $\backslash\text{DeclareTextCommand}$ 479
- 1994-10-27 ltoutenc.dtx v1.6b
 General: Added $\backslash\text{textbackslash}$. . . 499
 Added more defaults for OT1. . . 488
 Removed the enc.def files 475
 Removed the files OT1enc.def, T1enc.def and OMSenc.def. . . . 488
 Renamed $\backslash\text{textlbrace}$ to $\backslash\text{textbraceleft}$ and $\backslash\text{textrbrace}$ to $\backslash\text{textbraceright}$ 499
- 1994-10-29 ltmath.dtx 1.0m
 General: ASAJ: Added
 $\backslash\text{DeclareMathOperator}$ 844
 ASAJ: Tidied up documentation. 852
- 1994-10-29 ltmath.dtx v1.0m
 General: ASAJ: Added
 $\backslash\text{mathellipsis}$, $\backslash\text{mathdollar}$ and $\backslash\text{mathsterling}$ 851
 ASAJ: Removed $\backslash\text{dag}$, $\backslash\text{ddag}$ 851
 ASAJ: Renamed $\backslash\text{S}$ and $\backslash\text{P}$ to $\backslash\text{mathsection}$ and $\backslash\text{mathparagraph}$ and made them $\backslash\text{mathchardefs}$ 851
- 1994-10-29 ltoutenc.dtx v1.6c
 General: Added commands like $\backslash\text{dots}$ for use in text and math. 488
 Renamed $\backslash\text{P}$, $\backslash\text{S}$, $\backslash\text{dag}$ and $\backslash\text{ddag}$ to $\backslash\text{textparagraph}$, $\backslash\text{textsection}$, $\backslash\text{textdagger}$ and $\backslash\text{textdaggerdbl}$ 475
- 1994-10-30 ltdefs.dtx v1.2c
 $\backslash\text{@onelevel@sanitize}$: Macro added 109
 General: (CAR) $\backslash\text{@onelevel@sanitize}$ added 80
- 1994-10-30 ltdefs.dtx v1.2f
 General: (DPC) $\backslash\text{newwrite}$'s moved to ltfiles 80
- 1994-10-30 ltmath.dtx v1.0n
 General: ASAJ: Moved the new commands to ltoutenc. 851
- 1994-10-30 ltoutenc.dtx v1.6d
 General: Added
 $\backslash\text{DeclareTextCompositeCommand}$. 475
 Added $\backslash\text{textcircled}$. . . 475, 489, 499
 Added $\backslash\text{t}$ 489
 Added math commands. 475
 Added OML encoding. 475, 489
 Added the OML encoding. 500
- Made $\backslash\text{textless}$ and $\backslash\text{textgreater}$ come from OML. 489
 Moved math commands here from ltmath. 491
 Removed $\backslash\text{textregistered}$ 489
 Rewrote $\backslash\text{copyright}$ to use $\backslash\text{textcircled}$ 489
- 1994-10-31 fontdef.dtx v2.2d
 General: Added OMLenc.def 725
- 1994-10-31 fontdef.dtx v2.2e
 General: ... and moved further down 725
- 1994-10-31 ltfloat.dtx v1.1a
 $\backslash\text{@dblfloat}$: Major changes since two-column and one-column cases merged 975
 $\backslash\text{@dblflset}$: Macro added 975
 Major changes to parameter parsing, setting of local variables, etc; two-column and one-column cases merged; space hacks moved 975
 $\backslash\text{@endfloatbox}$: (DPC/CAR) Extra box added to remove colour resetting from vmode 981
 $\backslash\text{@floatboxreset}$: Macro added . . . 979
 $\backslash\text{@footnotetext}$: (DPC/CAR) Move colour setting to output routine . 991
 $\backslash\text{@savemarbox}$: (DPC/CAR) Extra box added for colour 985
 $\backslash\text{@setfps}$: Macro added 976
 $\backslash\text{@xdblfloat}$: Macros removed:
 $\backslash\text{@dbflt}$, $\backslash\text{@xdblfloat}$ 981
 $\backslash\text{@xfloat}$: (DPC/CAR) Extra box added to remove colour resetting from vmode 977
 Major changes, removing setting of local variables, space hacks etc; two-column and one-column cases merged 976
 Reset hook added 977
 $\backslash\text{@xympar}$: (DPC/CAR) Extra box added since needed for floats . . . 986
 $\backslash\text{fps@dbl}$: Macro added 976
- 1994-10-31 ltoutput.dtx v1.1a
 $\backslash\text{@topnewpage}$: (DPC/CAR) Extra box added to remove colour resetting from vmode 1168
 (DPC/CAR) Use
 $\backslash\text{color@begingroup}$ for colour . 1168
 (DPC/CAR) Use $\backslash\text{normalcolor}$ 1168
- 1994-11-02 ltoutenc.dtx v1.6d
 General: Wrapped lines longer than 70 characters. 475

1994-11-03 ltclass.dtx v1.0k	1994-11-04 ltpage.dtx v1.0e
General: Move <code>\@missingfileerror</code>	<code>\markright</code> : Added
to <code>ltfiles</code> 1042	<code>\@unexpandable@protect</code> .
1994-11-03 ltdirchk.dtx v1.0i	ASAJ. 1034
General: Generate an error if latex.ltx	1994-11-04 ltsect.dtx 1.0h
not used with clean initex 1	<code>\sect</code> : (ASAJ) Added
1994-11-03 ltfiles.dtx v1.0j	<code>\protected@edef</code> 962
<code>\@missingfileerror</code> : Move here from	General: (ASAJ) Added
<code>ltclass</code> 470	<code>\protected@xdef</code> to <code>\thanks</code> . . . 958
1994-11-04 ltboxes.dtx v1.0m	1994-11-04 ltsect.dtx v1.0h
<code>\@mpfootnotetext</code> : Added	<code>\addcontentsline</code> : Added
<code>\protected@edef</code> . ASAJ. 892	<code>\protected@write</code> to
1994-11-04 ltdefs.dtx v1.2e	<code>\addcontentsline</code> . ASAJ. 967
General: Added	<code>\addtocontents</code> : Added
<code>\set@display@protect</code> to	<code>\protected@write</code> to
<code>\typeout</code> . ASAJ. 80	<code>\addtocontents</code> . ASAJ. 968
Added commands for setting and	1994-11-04 lttab.dtx v1.0h
restoring <code>\protect</code> . ASAJ. 92	<code>\mkpream</code> : (ASAJ) Added
Rewrote protected short commands	<code>\@unexpandable@protect</code> to
using <code>\x@protect</code> . ASAJ. 90	<code>\mkpream</code> 916
1994-11-04 lterror.dtx v1.2g	<code>\multicolumn</code> : (ASAJ) added
General: Added	<code>\set@typeset@protect</code> 912
<code>\set@display@protect</code> to	1994-11-04 ltxref.dtx v1.1d
<code>\Generic*</code> commands. ASAJ. . . 397	<code>\label</code> : (ASAJ)Added
1994-11-04 ltfiles.dtx v1.0k	<code>\protected@edef</code> 801
<code>\nofiles</code> : Added setting of	(ASAJ)Added <code>\protected@write</code> 801
<code>\protected@write</code> , <code>\makeindex</code>	1994-11-05 ltboxes.dtx v1.0n
and <code>\makeglossary</code> to <code>\nofiles</code> .	<code>\@mpfootnotetext</code> : Color resetting for
ASAJ. 458	footnotes moved to <code>endminipage</code> :
<code>\protected@write</code> : Macro added	as for main page. 892
ASAJ. 458	<code>\color@endbox</code> : macro added for color
1994-11-04 ltfloat.dtx v1.1b	support 882
<code>\@footnotetext</code> : (ASAJ) Added	<code>\color@hbox</code> : macro added for color
<code>\protected@edef</code> 991	support 882
<code>\footnotemark</code> : Added	<code>\endminipage</code> : Color resetting for
<code>\protected@xdef</code> to	footnotes moved to here: as for
<code>\footnotemark</code> 992	main page. 892
1994-11-04 ltidxglo.dtx v1.1b	1994-11-05 ltboxes.dtx v1.0o
<code>\@wrglossary</code> : Added	<code>\@mpfootnotetext</code> : Color groups
<code>\protected@write</code> to	restored here. 892
<code>\@wrglossary</code> 997	1994-11-05 ltfloat.dtx v1.1c
<code>\@wrindex</code> : Added <code>\protected@write</code>	<code>\@dblflset</code> : Add compatibility with
to <code>\@wrindex</code> 996	old version of <code>\@xfloat</code> 975
General: Removed <code>\if@filesw</code> from	<code>\@endfloatbox</code> : Use new <code>\color@hbox</code>
<code>\makeindex</code> 995	concept. 981
<code>\makeglossary</code> : Removed <code>\if@filesw</code>	<code>\@footnotetext</code> : Removed
from <code>\makeglossary</code> 996	<code>\normalcolor</code> (again) 991
1994-11-04 ltmiscen.dtx v1.0t	<code>\@savemarbox</code> : Use new <code>\color@hbox</code>
<code>\@writefile</code> : Removed setting of	concept. 985
<code>\protect</code> . ASAJ. 826	<code>\@setfps</code> : Add compatibility with old
1994-11-04 ltoutenc.dtx v1.6f	version of <code>\@xfloat</code> 976
General: Added <code>_</code> 490	<code>\@xfloat</code> : Add compatibility with old
Added <code>\mathunderscore</code> 491	version of <code>\@xfloat</code> : but the

arguments, provided at exorbitant cost, are now completely ignored	976	1994-11-09 ltfsbas.dtx v2.1v	
Use new <code>\color@hbox</code> concept. . .	977	<code>\@vpt</code> : (DPC) macros added, from	
<code>\@xympar</code> : Use new <code>\color@hbox</code> concept.	986	setsizes.dtx	561
1994-11-05 ltoutenc.dtx v1.6g		(DPC) reduce save stack usage	
General: Added setting of		latex/1742	561
<code>\@typeset@protect</code> to <code>\patterns</code>		1994-11-10 ltbibl.dtx v1.1c	
and <code>\hyphenation</code>	487	General: Fix <code>\nocite{*}</code>	998
1994-11-05 ltoutput.dtx v1.1b		<code>\nocite</code> : Fix <code>\nocite{*}</code>	1001
<code>\@topnewpage</code> : Use new <code>\color@hbox</code> concept.	1168	1994-11-10 ltmath.dtx v1.2v classes	
<code>\@writsetup</code> : Change protect settings for new-style, protect-free aux-files.	1188	<code>eqnarray</code> : Added value of <code>\parskip</code> to <code>\abovedisplayskip</code> to compensate for negative <code>\topsep</code>	859
Use new <code>\color@hbox</code> concept. . .	1188	1994-11-10 ltoutput.dtx v1.1e	
1994-11-05 ltoutput.dtx v1.1c		<code>\@writsetup</code> : Modify <code>\protect</code> setting	1188
<code>\@begindiv</code> : Added macro	1194	1994-11-10 ltplain.dtx v1.1b	
<code>\@beginvibox</code> : Added macro . . .	1164	General: (CAR) added patch to <code>\loop</code> . . .	14
<code>\@writsetup</code> : Add new <code>\AtBeginDvi</code> concept	1188	<code>\iterate</code> : (CAR) added extra <code>\relax</code> . . .	29
<code>\AtBeginDvi</code> : Added macro	1164	1994-11-11 ltspacedtx v1.2a	
1994-11-06 ltfsbas.dtx v2.1u		<code>\:</code> (DPC) Make robust	434
<code>\cf@encoding</code> : New macro	545	1994-11-12 ltfntcmd.dtx v3.3o	
<code>\DeclareFixedFont</code> : Renamed <code>\every@size</code> to <code>\every@math@size</code>	535	<code>\normalsize</code> : Added <code>\MessageBreak</code>	755
1994-11-06 ltfsini.dtx v2.2b		1994-11-12 ltlists.dtx v1.2b ltspaced	
<code>\@setsizes</code> : Use <code>\typeset@protect</code>	717	<code>\endtrivlist</code> : Changed order of tests to make <code>\noitemerror</code> correct: end of an era.	869
1994-11-06 ltfsstrc.dtx v2.3k		1994-11-12 ltmiscen.dtx v1.0u	
<code>\glb@currsize</code> : New implementation	641	<code>center</code> : Changed end macro to <code>\def</code> : safer and consistent	834
<code>\try@simples</code> : New implementation	652	<code>flushleft</code> : Changed end macro to <code>\def</code> : safer and consistent	835
<code>\try@size@substitution</code> : New implementation	652	<code>flushright</code> : Changed end macro to <code>\def</code> : safer and consistent	836
<code>\tryis@simple</code> : New implementation	653	1994-11-12 ltplain.dtx v1.1c	
1994-11-07 fontdef.dtx v2.2f		General: Comment out more encoding specific commands	31
General: (DPC) Add <code>\DeclareMathSizes</code> declarations (DPC) Updated to use <code>\ProvidesFile</code>	725	1994-11-12 ltspacedtx v1.2b	
1994-11-07 ltfiles.dtx v1.0m		<code>\addpenalty</code> : Corrected error message	442
<code>\document</code> : Renamed <code>\every@size</code> to <code>\every@math@size</code>	454	<code>\addvspace</code> : Corrected error message	441
1994-11-07 ltplain.dtx v1.0l		1994-11-13 ltspacedtx v1.2c	
<code>\@unused</code> : move here from ltdefs, remove duplicate <code>\@mainaux</code>	23	<code>\addpenalty</code> : Recorrected error message	442
1994-11-07 preload.dtx v2.1e		<code>\addvspace</code> : Recorrected error message	441
General: (DPC) Updated to use <code>\ProvidesFile</code>	744	1994-11-14 ltoutput.dtx v1.1f	
1994-11-09 ltboxes.dtx v1.0p		<code>\@begindiv</code> : Use normal box register: why a box?	1194
<code>\@finalstrut</code> : Revert <code>\finalstrut</code> to 2.09 equivalent (from ltpatch)	895	<code>\@beginvibox</code> : Use normal box register: why a box?	1164
General: more color changes.	879	<code>\@writsetup</code> : Modify new <code>\AtBeginDvi</code> concept	1188
		General: Removed old definition of <code>\@testfp</code>	1152

1994-11-14 ltspace.dtx v1.2d		1994-11-18 ltfsbas.dtx v2.1x	
\backslash : (DPC) Macro modified	434	General: (DPC) use \backslash reserved@f not	
1994-11-14 lttab.dtx v1.0i		\backslash next	533
\backslash tabularnewline: (DPC) Macro		1994-11-18 ltfsdcl.dtx v2.1m	
added	911	\backslash DeclareMathDelimiter: (DPC)	
1994-11-16 fontdef.dtx v2.2h		\backslash expandafter instead of \backslash next . .	687
General: (DPC) Removed \backslash { and \backslash }	725	1994-11-18 ltfsstrc.dtx v2.3m	
1994-11-17 ltboxes.dtx v1.0q		General: \backslash next to \backslash reserved@f . . .	633
General: \backslash @tempa to \backslash reserved@a . .	879	1994-11-18 ltmath.dtx v1.0p	
1994-11-17 ltclass.dtx v1.0l		\backslash phantom: (DPC) colour support . .	846
General: \backslash @tempa to \backslash reserved@a . .	1038	(DPC) use \backslash expandafter instead of	
1994-11-17 ltcntrl.dtx v1.0b		\backslash next	846
General: \backslash @tempa to \backslash reserved@a . .	393	\backslash prime@s: (DPC) use \backslash @let@token	
1994-11-17 ltdefs.dtx v1.0g		instead of \backslash next and	
General: \backslash @tempa to \backslash reserved@a . . .	80	\backslash expandafter instead of \backslash nxt . .	851
1994-11-17 ltdirchk.dtx v1.0j		\backslash smash: (DPC) colour support	847
General: \backslash @tempa to \backslash reserved@a	1	(DPC) use \backslash expandafter instead of	
1994-11-17 lterror.dtx v1.2h		\backslash next	847
General: \backslash @tempa to \backslash reserved@a . .	397	1994-11-21 ltfloat.dtx v1.1f	
1994-11-17 ltfiles.dtx v1.0n		\backslash @endfloatbox: Added reset of	
General: \backslash @tempa to \backslash reserved@a . .	451	minipage flag	981
1994-11-17 ltfinal.dtx v1.0o		Corrected position of	
General: \backslash @tempa to \backslash reserved@a . .	1260	\backslash outer@nobreak	981
1994-11-17 ltfloat.dtx v1.1e		\backslash @marginparreset: Macro added . .	985
General: \backslash @tempa to \backslash reserved@a . .	972	\backslash @savemarbox: Added \backslash @setminipage	
1994-11-17 ltfntcmd.dtx v3.3p		etc	985
General: \backslash @tempa to \backslash reserved@a . .	747	Added resetting of size and font . .	985
1994-11-17 ltfsbas.dtx v2.1w		Changed to \backslash color@vbox	985
General: \backslash @tempa to \backslash reserved@a . .	533	Use \backslash @setnobreak etc	985
1994-11-17 ltfsdcl.dtx v2.1m		\backslash @setminipage: Macro added	979
General: \backslash @tempa to \backslash reserved@a . .	663	\backslash @setnobreak: Macro added	979
1994-11-17 ltfsstrc.dtx v2.3l		\backslash @xfloat: Added \backslash @setminipage . .	977
General: \backslash @tempa to \backslash reserved@a . .	633	Added resetting of size and font . .	977
1994-11-17 ltmath.dtx v1.0o		Changed to \backslash color@vbox so that	
General: \backslash @tempa to \backslash reserved@a . .	844	large floats overflow at the	
1994-11-17 ltmiscen.dtx v1.0v		bottom	977
General: \backslash @tempa to \backslash reserved@a . .	819	Missing percents reinserted after 4,	
1994-11-17 ltoutenc.dtx v1.6h		8: these are not numbers.	976
General: (DPC) \backslash @tempa to		Use \backslash @setnobreak	977
\backslash reserved@a	475	\backslash @xympar: Changed to \backslash color@vbox	986
1994-11-17 ltoutput.dtx v1.1h		1994-11-21 ltoutput.dtx v1.1i	
General: \backslash @tempa to \backslash reserved@a. . .	1152	\backslash @addtocurcol: Added \backslash if@nobreak	
1994-11-17 ltpictur.dtx v1.0f		test before float box	1204, 1210
General: \backslash @tempa to \backslash reserved@a . .	922	\backslash @specialoutput: Added \backslash if@nobreak	
1994-11-17 ltsect.dtx v1.0i		test	1174
General: \backslash @tempa to \backslash reserved@a . .	958	\backslash @topnewpage: Changed to	
1994-11-17 lttab.dtx v1.0j		\backslash color@vbox	1168
General: \backslash @tempa to \backslash reserved@a . .	897	1994-11-22 ltfsdcl.dtx v2.1o	
1994-11-18 ltboxes.dtx v1.0r		General: wrap long lines	663
\backslash color@vbox: macro added for color		1994-11-22 ltoutenc.dtx v1.6i	
support	882	General: Corrected \backslash dots so that	
1994-11-18 ltfinal.dtx v1.0n		there's no kerning in monowidth	
General: re-allow slots 127–255 . . .	1272	fonts.	475

Corrected typo with <code>\mathunderscore</code>	475	Rewrote <code>\@text@composite</code> so it allows an empty argument, or an argument containing lots of commands.	482
Fixed empty accents. Again. . . .	475		
1994-11-24 ltdefs.dtx v1.2h <code>\@newenv</code> : Added test for <code>\endgraf</code> . .	87	1994-12-01 ltfinal.dtx v1.0p General: Renamed <code>lthyphen.*</code> to <code>hyphen.*</code>	1260
1994-11-25 ltplain.dtx v1.1f General: (DPC) Comment out lots of obsolete code	14	1994-12-01 lthyphen.dtx v1.0g General: Rename <code>lthyphen.ltx/cfg</code> to <code>hyphen.ltx/cfg</code>	1258
1994-11-26 ltfloat.dtx v1.1b <code>\footnote</code> : (ASAJ) Added <code>\protected@xdef</code>	991	1994-12-01 ltplain.dtx v1.1g General: (DPC) More doc changes . .	14
1994-11-28 ltcntrl.dtx v1.0c General: Documentation improvements	393	1994-12-02 fontdef.dtx v2.2i General: Commented out <code>\ldots</code> . ASAJ.	723
1994-11-30 ltfiles.dtx v1.0o <code>\@dofilelist</code> : Macro added	474	1994-12-02 ltffssini.dtx v2.2c <code>\copyright</code> : <code>\copyright</code> is now in <code>ltoutenc</code> . ASAJ	718
<code>\listfiles</code> : Use <code>\@dofilelist</code> . . .	472	1994-12-02 ltlists.dtx v1.0e <code>\@trivlist</code> : RmS: Added check for looping	868
<code>\nofiles</code> : There is no <code>\@gobblethree</code>	458	1994-12-02 ltoutenc.dtx 1.7b General: Redefined <code>\a</code> properly. . . .	488
1994-11-30 ltffssbas.dtx v2.1y <code>\fontshape</code> : Use <code>\@current@cmd</code> in <code>\@enc@update</code> . ASAJ.	544	1994-12-02 ltoutenc.dtx v1.7b General: Fixed a bug with <code>\a</code>	475
1994-11-30 ltmath.dtx 1.0q General: ASAJ: <code>\DeclareMathOperator</code> moved to <code>AMSLATEX</code>	844	1994-12-04 lthyphen.dtx v1.0h General: Documentation edits for <code>/1989</code>	1258
1994-11-30 ltmiscen.dtx v1.0w <code>\@enddocument@kernel@warnings</code> : (DPC) Do warnings even for <code>\nofiles</code>	822	1994-12-05 ltoutenc.dtx v1.7c General: Added braces to <code>\textcircled</code>	475
<code>\enddocument</code> : (DPC) Use <code>\@dofilelist</code>	821	1994-12-06 ltffssbas.dtx v2.1z <code>\DeclareFontEncoding</code> : use <code>\nfss@catcodes</code>	537
1994-11-30 ltoutenc.dtx 1.7a General: Redefined <code>\a</code> for the new scheme.	488	<code>\nfss@catcodes</code> : Added tab char as well	551
1994-11-30 ltoutenc.dtx v1.6g General: Removed new definitions of <code>\patterns</code> and <code>\hyphenation</code> , since encoding-specific commands now expand in the mouth.	487	1994-12-08 ltoutenc.dtx v1.7d General: Added <code>\null</code> and <code>\sh@ft</code> to <code>\b</code> and <code>\d</code>	475
1994-11-30 ltoutenc.dtx v1.7a General: Added new code for encoding-specific commands. These now expand in the mouth, which means that ligaturing and kerning can happen.	475	1994-12-08 lttab.dtx v1.0k <code>\@array</code> : Add <code>\tabularnewline</code> . . .	910
Always load the <code>enc.def</code> file, so that the default encoding for the commands will change.	518	<code>\tabularnewline</code> : (DPC) Made it <code>\relax</code>	911
Redefined <code>\@changed@cmd</code> to expand in the mouth.	479	1994-12-09 ltbibl.dtx v1.1d <code>\bibliographystyle</code> : (DPC) Allow use in preamble.	1001
Removed <code>\@changed@x@mouth</code> since <code>\@changed@x</code> now expands in the mouth.	479	1994-12-10 ltfloat.dtx v1.1g <code>\@dblfloat</code> : Old version reinstated temporarily	975
		<code>\@dblfloatset</code> : Macro removed temporarily	975
		Old version reinstated temporarily	975
		<code>\@setfps</code> : Macro removed temporarily	976

<code>\@xdblfloat</code> : Macros reinserted temporarily	981	1995-04-02 ltfssini.dtx v2.2d	<code>\not@math@alphabet</code> : add <code>\noexpand</code> to second part of message	716
<code>\@xfloat</code> : Old version reinstated temporarily	976	1995-04-21 ltclass.dtx v1.0m	<code>\DeclareOption*</code> : Made long /1498	1055
Sanitization added temporarily . .	976	<code>\endfilecontents</code> : Close input check stream: latex/1487		1074
General: Some temps reinserted temporarily	972	1995-04-21 ltfinal.dtx v1.0q	General: Allow initial patch level 0	1276
<code>\fps@dbl</code> : Macro removed temporarily	976	1995-04-21 ltoutenc.dtx v1.7h	General: Added <code>\null \k</code> latex/1274	475
1994-12-10 ltfntcmd.dtx v3.3q		1995-04-22 ltfiles.dtx v1.0p	<code>\includeonly</code> : Allow blanks in argument	459
<code>\@math@egroup</code> : Don't read arguments	754	1995-04-22 ltmiscen.dtx v1.0x	General: Removed extra def of <code>\@gobble</code>	819
<code>\check@nocorr@</code> : Use <code>\space</code> command for comparison	751	1995-04-23 ltsect.dtx v1.0j	<code>\addcontentsline</code> : Use <code>\contentsline</code> internally.	967
1994-12-10 ltfssdcl.dtx v2.1p		1995-04-24 ltbibl.dtx v1.1e	<code>\@citex</code> : Add <code>\mbox</code> to undefined case: latex/1239.	1000
<code>\document@select@group</code> : Surround with braces (add fourth arg) . . .	669	1995-04-24 ltbibl.dtx v1.1f	<code>\bibcite</code> : Make <code>\@onlypreamble</code> /1388.	999
<code>\select@group</code> : Surround with braces (add fourth arg)	666	1995-04-24 ltcntrl.dtx v1.0d	<code>\@for</code> : Don't expand second argument with <code>\edef</code> : /1317 (DPC)	396
1994-12-10 ltoutenc.dtx v1.7e		1995-04-24 ltoutput.dtx v1.1j	<code>\fl@tracemessage</code> : Do not add to kernel unless 'trace' specified . .	1226
General: Added documentation for the OML encoding.	475	1995-04-24 ltoutput.dtx v1.1l	<code>\@begindivbox</code> : Add <code>\vbox</code> latex/1392	1164
Replaced width with <code>\@width</code> and ditto height in vrules.	475	<code>\@writsetup</code> : Reset <code>\@</code> latex/1451 (DPC)		1189
1994-12-14 ltoutenc.dtx v1.7f		1995-04-24 ltpage.dtx v1.0f	<code>\fussy</code> : reset <code>\emergencystretch</code> latex/1344	1037
General: Added braces to <code>\copyright</code> so it works unbraced in subscripts.	475	1995-04-24 ltplain.dtx v1.1h	<code>\newlanguage</code> : Remove remaining <code>\outer</code> declarations.	17
Added check for math mode in <code>\@changed@cmd</code>	475	1995-04-24 ltxref.dtx v1.1e	<code>\newlabel</code> : Make <code>\@onlypreamble</code> for /1388.	800
Commented out <code>\textasciicircum</code> , <code>\textasciitilde</code> , <code>\textbackslash</code> , <code>\textbar</code> , <code>\textgreater</code> , <code>\textthyphenchar</code> , <code>\textthyphen</code> and <code>\textless</code> to save memory.	475	1995-04-25 ltdefs.dtx v1.2i	<code>\@check@c</code> : Make <code>\long</code> for latex/1346	88
1995-01-12 ltmath.dtx v1.2y classes		<code>\new@environment</code> : Parse arguments slowly but safely /1507		86
<code>\@eqnnum</code> : Added <code>\normalcolor</code> . . .	857	1995-04-25 ltfiles.dtx v1.0q	<code>\document</code> : Removed execution of <code>\every@size</code> latex/1407	454
1995-03-03 ltoutenc.dtx 1.7g				
General: Corrected an error in documentation referring to the tabular rather than the tabbing environment.	488			
1995-04-02 ltfntcmd.dtx v3.3r				
<code>\@math@egroup</code> : Read them again to be able to add <code>\relax</code>	754			
1995-04-02 ltfssdcl.dtx v2.1q				
<code>\document@select@group</code> : fix problem for pr/1275	669			
<code>\select@group</code> : fix problem for pr/1275	666			
<code>\set@mathdelimitter</code> : fix pr/1329 . .	690			

1995-04-25 ltsect.dtx v1.0k	1995-05-07 ltoutput.dtx v1.1m
\@dottedtocline: Added \hbox	General: Use \hb@xt@. 1152
around dots. 970	1995-05-07 ltpictur.dtx v1.0g
1995-04-27 ltboxes.dtx v1.0s	General: Use \hb@xt@ 922
\@frameb@x: Move \leavevmode for	1995-05-07 ltplain.dtx v1.1j
graphics/1512 887	General: Use \hb@xt@ 14
\@ifframebox: Move \leavevmode for	1995-05-07 ltsect.dtx v1.0o
graphics/1512 886	General: Use \hb@xt@ 958
\@iirsbox: Move \leavevmode for	1995-05-07 lttab.dtx v1.0l
graphics/1512 895	General: Use \hb@xt@ 897
\@irsbox: Move \leavevmode for	1995-05-08 ltbibl.dtx v1.1g
graphics/1512 895	\@citex: Use \@firstofone 1000
\fbbox: Move \leavevmode for	\bibitem: Removed unnecessary
graphics/1512 885	braces 999
\raisebox: Move \leavevmode for	\ncite: Use \@firstofone 1001
graphics/1512 894	1995-05-08 ltdefs.dtx v1.2k
1995-04-27 ltfiles.dtx v1.0r	\typein: Use \@firstofone 82
\document: Added \global to support	1995-05-08 ltdefs.dtx v1.2l
groups in hook 455	\typein: Remove unnecessary braces 82
1995-04-27 ltmiscen.dtx v1.0y	Replace \def by \let 82
\enddocument: \@checkend moved	1995-05-08 ltfstrc.dtx v2.3n
after hook 820	\ifnot@nil: Use \@firstofone 646
1995-04-27 ltplain.dtx v1.1i	1995-05-11 fontdef.dtx v2.2j
General: Move \hang and	General: Updates to some plain
\textindent to latex209.def 31	macros 723
1995-04-29 ltcntrl.dtx v1.0e	1995-05-12 ltclass.dtx v1.0n
General: Moved init of \protect to	\DeclareOption*: Use \toks@ to
ltdefs.dtx 396	remove need to double hash
Removed unused defs for	/1557 1055
\@setprotect and	1995-05-12 ltfloat.dtx v1.1h
\@resetprotect 396	\@footnotemark: Add \nobreak to
1995-04-29 ltdefs.dtx v1.2j	allow hyphenation. latex/1605 993
\protect: Init \protect here 92	1995-05-12 ltpictur.dtx v1.0h
1995-04-29 ltpar.dtx v1.1b	\pictur@: Macro added for
General: (TO) Comments clean-up. 408	latex/1355 924
1995-05-02 ltsect.dtx v1.0l	1995-05-12 ltvers.dtx v1.0e
\@dottedtocline: Don't reset to	General: Add autoload docstrip guards 37
\rmfamily 970	Check for format older than 1 year 37
1995-05-03 ltsect.dtx v1.0m	1995-05-13 ltfstrc.dtx v2.3o
General: TO: Promoted	General: Use single hash mark in
documentation to doc.sty	\DeclareOption 634
standard 958	1995-05-16 ltfloat.dtx v1.1i
1995-05-06 ltsect.dtx 1.0n	\@makefnmark: Now use
\@secntformat: Use \quad instead of	\textsuperscript. 990
\hskip 964	\textsuperscript: Command
\@sect: Added \relax after	added./pr1503 990
\@secntformat just in case 962	\thefootnote: Streamlined parts of
1995-05-07 ltboxes.dtx v1.0t	code. 989
General: Use \hb@xt@ 879	1995-05-17 ltboxes.dtx v1.0u
1995-05-07 ltdefs.dtx v1.2k	\@irsbox: Removed surplus braces 895
\hb@xt@: Macro added 81	1995-05-17 ltdefs.dtx v1.0o
1995-05-07 ltmath.dtx v1.0r	\g@addto@macro: Make long for
General: Use \hb@xt@ 844	latex/1522 112

1995-05-17 ltlists.dtx v1.0g		1995-05-24 ltdefs.dtx v1.1l	
\@item: Removed surplus braces . . .	874	\newif: (DPC) New implementation .	87
\@nbitem: Removed surplus braces .	875	1995-05-24 ltdefs.dtx v1.2m	
enumerate: Use \thr@@ and remove		\typein: (DPC) New implementation	82
surplus braces	876	1995-05-24 ltfloat.dtx v1.1l	
itemize: Use \thr@@	876	\@textsuperscript: Command	
1995-05-18 ltfloat.dtx v1.1j		added.	990
\@makefnmark: Added \normalfont.	990	General: Moved definition of \footins	
\thempfootnote: Added \itshape. .	989	and \footnoterule from ltplain.	989
1995-05-19 ltpictur.dtx v1.1a		\textsuperscript: Use	
General: Support autoloading feature	922	\@textsuperscript	990
1995-05-20 ltcounts.dtx v1.1b		1995-05-24 ltfssbas.dtx v3.0a	
\@definecounter: Streamlined code	524	General: (DPC) Make file from	
\@fnsymbol: Allowing both text and		previous file, fam.dtx 1995/05/20	
math	529	v2.2d	533
\fnsymbol: Streamlined code	528	\mathgroup: (DPC) No need to	
1995-05-20 ltcounts.dtx v1.1c		redefine \newfam as not outer . .	533
\@definecounter: And do it right .	524	1995-05-24 ltfsscnp.dtx v3.0a	
1995-05-20 ltfloat.dtx v1.1k		General: (DPC) Make file from	
\@makefnmark: Moved \normalfont		previous file, fam.dtx 1995/05/20	
back and use \@textsuperscript	990	v2.2d	658
Moved \normalfont to		1995-05-24 ltfssdcl.dtx v3.0a	
\textsuperscript	990	General: (DPC) Make file from	
\textsuperscript: Use \normalfont.	990	previous file, latint.dtx 1995/05/21	
1995-05-21 ltfssdcl.dtx v2.1t		v2.1t	663
\DeclareMathRadical: Allow for		1995-05-24 ltfssini.dtx v3.0a	
undefined cs names	691	General: (DPC) Make file from	
1995-05-21 ltlists.dtx v1.0f		previous file, lfonts.dtx 1995/05/23	
General: Moved to doc.sty standard	860	v2.2e	695
1995-05-21 ltmath.dtx v1.0r		\cal: (DPC) Remove definition . . .	721
\@sqrt: Use \sqrtsign	854	\mit: (DPC) Remove definition . . .	721
General: Remove \mathhexbox from		1995-05-24 ltfsstrc.dtx v3.0a	
this file	849	General: (DPC) Make file from	
Update some plain macros	844	previous file, tracefnt 1995/05/16	
\lefteqn: Use \rlap	856	v2.3o	633
\@t: Use \sqrtsign instead of		1995-05-24 ltfsstrc.dtx v3.0b	
\sqrt	846	General: (DPC) Fix \ProvidesFile	
1995-05-21 ltoutenc.dtx v1.7h		usage	633
\@inmathwarn: Added several		1995-05-25 ltclass.dtx v1.0p	
\@onlypreamble	480	\endfilecontents: Delete	
1995-05-21 ltoutenc.dtx v1.7j		\filecontents after preamble .	1074
General: Updated some plain macros	492	1995-05-25 ltfilehook.dtx v1.0t	
1995-05-21 ltplain.dtx v1.1j		\unqu@tefilef@und: (CAR) added	
General: Moved some code to other		\long	1110
files	14	1995-05-25 ltfiles.dtx v1.0s	
1995-05-22 ltplain.dtx v1.1k		\document: Added check for \topskip	
General: Definitions of \footins and		zero	455
\footnoterule moved to ltfloat. .	33	1995-05-25 ltfiles.dtx v1.0t	
1995-05-22 lttab.dtx v1.1a		\iffileonpath: (CAR) added \long	468
General: Support autoloading feature	897	\document: Corrected typo	455
1995-05-23 ltfssini.dtx v2.2e		\IfFileExists@: (CAR) added \long	466
\newfont: Font assignment made local		\nofiles: (CAR) added \long	458
again.	717	\protected@write: (CAR) added	
		\long	458

1995-05-25 lfloat.dtx v1.1m	1995-06-06 ltfinal.dtx v1.0s
\@savemarbox: (CAR) Resettings	General: Made \MakeUppercase and
moved to hook 985	\MakeLowercase brace their
\@xfloat: (CAR) Resettings moved to	argument. 1260
hook 977	1995-06-09 ltoutenc.dtx v1.7l
1995-05-25 ltlists.dtx v1.0i	\DeclareTextComposite: Rewrote
\endtrivlist: Macros moved from	\DeclareTextComposite to define
ltspace.dtx 869	the composite as a no-argument
1995-05-25 ltmath.dtx v1.3c classes	command rather than a
\@eqnnum: replace	two-argument command. 483
\reset@font\rmfamily with	1995-06-11 ltspace.dtx v1.2g
\normalfont (PR 1578) 857	\restorecr: (CAR) \relax added to
1995-05-25 ltspace.dtx v1.2f	stop silent eating of *. 449
\@vbsphack: (CAR) not used so	1995-06-13 ltfinal.dtx v1.0t
‘removed’. 440	General: Add patch level string more
\@vspacer: (CAR) \@restorepar	carefully 1276
added to avoid possible infinite tail	Call \errorstopmode 1279
recursion caused by a typo in the	1995-06-13 ltpictur.dtx v1.1b
argument. 444	General: Use \ProvidesFile in
(CAR) macros modified to be more	autoload 922
efficient 444	1995-06-14 lttab.dtx v1.1b
General: Macros moved to ltlists.dtx	General: Use \ProvidesFile in
430	autoload 897
1995-05-26 ltdefs.dtx v1.2n	1995-06-15 ltfssbas.dtx v3.0c
\@gobblefour: (CAR) Added \longs	General: (DPC) minor documentation
89	changes 533
1995-05-26 ltmath.dtx v1.0s	1995-06-15 ltfsscnp.dtx v3.0b
\@eqnnum: Removed \rmfamily (PR	General: (DPC) minor documentation
1578), replaced \reset@font with	edits 658
\normalfont 853	1995-06-15 ltfssdcl.dtx v3.0b
1995-05-26 ltpage.dtx v1.0g	General: (DPC) minor documentation
\ps@plain: removed \rmfamily (PR	changes 663
1578) 1034	1995-06-19 ltbibl.dtx v1.1h
1995-05-27 ltfssbas.dtx v3.0b	\bibcite: Call \@newl@bel so
\mathgroup: (FMi) But a need to	repeated keys produce better
define \new@mathgroup 533	warning. 999
1995-06-05 fontdef.dtx v2.2k	1995-06-19 ltclass.dtx v1.0q
General: Moved math commands from	\documentclass: Don’t redefine
ltoutenc.dtx. 741	\usepackage in compat mode for
1995-06-05 ltfinal.dtx v1.0r	/1634 1059
General: Added \MakeUppercase and	1995-06-19 ltxref.dtx v1.1e
\MakeLowercase. 1260	\newlabel: Use \@newl@bel to share
1995-06-05 ltoutenc.dtx v1.7k	code with \bibcite 800
\@inmathwarn: Removed	1995-06-28 ltfssini.dtx v3.0b
\protected@cmd and replaced with	General: (DPC) Fix documentation
explicit \noexpand. 480	typos 695
General: Allowed	1995-06-28 ltmath.dtx v1.0t
\ProvideTextCommandDefault	General: minor doc edits 844
after the preamble. 481	1995-07-02 ltplain.dtx v1.1n
Commented out \textless and	General: Removed surplus ‘by’ and ‘=’
\textgreater. 489	in various places 14
Moved math commands to	\offinterlineskip: Replaced 1000 by
fontdef.dtx. 491	\@m 30
Save some tokens in	
\textvisiblespace and	
\textunderscore. 489	

\showoutput: Use \showoverfull to save space	33	General: Improve Documentation . .	819
\tracingall: Use \showoutput to save space	33	\enddocument: Set \@setckpt to \@gobbletwo instead of defining it by hand	820
1995-07-03 ltdefs.dtx v1.2o		Shorten redefinition of \bible and \newlabel	821
\set@typeset@protect: Use \@typeset@protect for init	92	1995-07-14 ltbibl.dtx v1.1i	
1995-07-03 ltfntcmd.dtx v3.3s		\bible: Remove \@onlypreamble so still defined in new \enddocument	999
\tst@ic: Use clean interface for jump	753	1995-07-14 ltxref.dtx v1.1g	
1995-07-05 ltfntcmd.dtx v3.3s		\newlabel: Remove \@onlypreamble so still defined in new \enddocument	800
\tst@ic: Renamed from \test@next	753	1995-07-19 ltfssini.dtx v3.0d	
1995-07-05 ltspace.dtx v1.2h		General: (DPC) TeX2 support	721
\@gneline: Use \break	436	1995-07-20 ltboxes.dtx v1.0v	
\@no@pgbk: Macro replaces \@pgbk and \@nopgbk	434	\@isavebox: Use \sbox	884
\nopagebreak: Reimplemented both using \@no@pgbk	433	\@isavepicbox: Use \sbox	884
1995-07-09 ltcntrl.dtx v1.0f		1995-07-21 ltoutput.dtx v1.1o	
\@iforloop: Reimplemented using Kabelschacht method	396	\@writsetup: Command added . .	1188
\@iwhiledim: Reimplemented using Kabelschacht method	394	New, experimental, versions: need in-lining	1188
\@iwhilenum: Reimplemented using Kabelschacht method	394	1995-08-09 ltmath.dtx v1.0u	
\@iwhilesw: Reimplemented using Kabelschacht method	394	General: Added code for class options leqno and fleqn	857
\@tfor: Reimplemented using Kabelschacht method	396	1995-08-11 ltlength.dtx v1.1b	
1995-07-09 ltlists.dtx v1.0j		General: Doc typos fixed for latex/753	531
enumerate: Use \expandafter	876	1995-08-16 ltcntrl.dtx v1.0g	
itemize: Use \expandafter	877	\@break@tfor: Made long	396
1995-07-12 ltpictur.dtx v1.1d		\@forloop: Made defs long	396
General: allow 2e commands in 209 mode. latex/1737	922	\@fornoop: Made defs long	396
1995-07-13 ltdefs.dtx v1.0p		\@iforloop: Made defs long	396
General: Updates to documentation .	80	\@iwhiledim: Made defs long	394
1995-07-13 ltfiles.dtx v1.0u		Removed \@whilenoop	394
General: Updates to docu	451	\@iwhilenum: Made defs long	394
1995-07-13 ltfssbas.dtx v3.0d		Removed \@whilenoop	394
\@defaultsubs: macro added	556	\@iwhilesw: Removed \@whileswnoop	394
\@defaultsubs: macro added	556	\@tfor: Made defs long	396
General: minor documentation changes	533	1995-08-16 ltfiles.dtx v1.0v	
\wrong@fontshape: Change a macro not a switch to flag default font substitutions	555	\document: set \@maxdepth	455
1995-07-13 ltmiscen.dtx v1.0z		set \do globally	455
\@centercr: Use \nobreak	833	set \topskip globally	455
\@enddocument@kernel@warnings: Use \@defaultsubs instead of switch	822	1995-08-24 ltfssbas.dtx v3.0f	
\@writefile: Added missing percent and use \relax in the THEN case	826	General: Added autoload code	533
\@xobeysp: Use \nobreak	836	1995-08-24 ltfstrc.dtx v3.0c	
		General: Macro \gobble@font@spec removed	647
		\@tryis@simple:	654
		1995-08-25 ltoutput.dtx v1.1p	
		General: Support autoloading feature (FMi).	1152

1995-09-01 lterror.dtx v1.2i	1995-10-10 ltplain.dtx v1.1r
General: Add autoloading support . . . 397	General: Autoloading tracing code 14
1995-09-01 ltplain.dtx v1.1m	1995-10-10 ltthm.dtx v1.0f
\empty: Use \let to save space 27	General: Make \newtheorem ‘only
\I: Use \let to save space 27	preamble’ 954
1995-09-14 ltplain.dtx v1.1o	1995-10-11 ltoutput.dtx v1.1r
General: Moved \multispan to	\clearpage: Added a check so that it
lftab.dtx 14	does not lose the argument of
1995-09-14 lftab.dtx v1.1c	\twocolumn[...] 1165
\cline: (DPC) New implementation 920	1995-10-16 ltbibl.dtx v1.1j
1995-09-15 ltfssini.dtx v3.0e	\cite: (DPC) Make robust 999
General: (DPC) Modify TeX2	1995-10-16 ltboxes.dtx v1.0w
message 721	General: Clarify makebox description 879
1995-09-19 ltmiscen.dtx v1.1a	1995-10-16 ltdefs.dtx v1.2u
\verb: Put \noligs after	\ifstar: (DPC) New
\verbatim@font where it belongs. 842	implementation, for /1910 108
1995-10-01 ltfiles.dtx LaTeX2e	\new@command: (DPC) Use \@testopt
\addtofilelist: Macro added . . . 472	/1911 84
1995-10-02 ltdefs.dtx v1.2q	\new@environment: (DPC) Use
\ifnch: Use \@let@token for	\@testopt /1911 86
internal/924, save \reserved@e . 108	\typein: (DPC) Use \@testopt /1911 82
\ifnextchar: Use \@let@token . . 107	1995-10-16 ltfssini.dtx v3.0f
\protected@testopt: Macro added . 85	\reset@font: Added \relax after
\@testopt: Macro added 84	\usefont, as the latter eats up
\@xargdef: New implementation,	spaces. 718
using \@test@opt 84	1995-10-16 ltmath.dtx v1.0y
1995-10-03 fontdef.dtx v2.2l	\@yeqncr: (DPC) Use \@testopt
General: \@@sqrt from patch file for	/1911 855
/1701 723	\sqrt: (DPC) Make robust /1808 . . 854
1995-10-03 ltdefs.dtx v1.2r	1995-10-16 ltspc.dtx v1.2j
\typein: Add missing \@typein for	\nolinebreak: (DPC) Use \@testopt
/1710 (from patch file) 82	/1911 433
1995-10-03 ltpictur.dtx v1.1e	\nopagebreak: (DPC) Use \@testopt
General: New autoloading code 922	/1911 433
1995-10-04 ltfssbas.dtx v3.0g	1995-10-16 ltthm.dtx v1.0g
General: Modify autoloading code 533	General: Revert to previous
1995-10-04 ltfssstrc.dtx v3.0d	\newtheorem behaviour 954
General: (DPC) Modify autoloading	1995-10-17 ltclass.dtx v1.0r
code 633	\@providesfile: Delay definition of
1995-10-04 lftab.dtx v1.1d	\ProvidesFile till ltfinal 1053
General: Modify autoloading support . 897	\ProcessOptions*: Reset
1995-10-06 ltfiles.dtx v1.0w	\CurrentOption for
\@missingfileerror: Autoloading error 470	graphics/1873 1058
1995-10-09 lterror.dtx v1.2j	1995-10-17 ltdirchk.dtx v1.0l
General: Modify autoloading support . 397	General: Modify initex version of
1995-10-09 ltoutenc.dtx v1.7m	\ProvidesFile 4
\@inmathwarn: Autoloading error 481	1995-10-17 ltfinal.dtx v1.0v
1995-10-10 ltfssbas.dtx v3.0h	\@providesfile: reset macro 1278
\showhyphens: Use \normalfont and	\reserved@b: reset here after the
make colour safe, and	\input above 1277
autoloading 560	1995-10-17 ltplain.dtx v1.1s
1995-10-10 ltfssdcl.dtx v3.0c	\@eject: Move \supereject to compat
\@non@alpherr: (DPC) autoloading error	file 30
message 667	

1995-10-17 lttab.dtx v1.1e	1995-10-24 ltxref.dtx v1.1h
\@cline: (DPC) Use \@multicnt .. 920	\@multiplelabels: Switch for
\@multispan: (DPC) Macro added. 920	multiplelabels removed 801
1995-10-19 ltfinal.dtx v1.0w	\@newl@bel: Switch for multiplelabels
\@filelist: Move after \@reserved@a	replaced by inline code 800
setting 1278	\@refundefined: Switch for
1995-10-20 ltbibl.dtx v1.1k	refundefined replaced 798
\@citex: Removed refundefined flag 1000	\@setref: Switch for refundefined
\@nocite: Removed refundefined flag 1001	renamed 799
1995-10-20 ltclass.dtx v1.0s	\if@multiplelabels: Macro removed 801
\@begindocumenthook: Make setting	1995-10-25 ltalloc.dtx v1.1b
conditional, for autoloading version 1071	General: General doc improvements 391
1995-10-20 ltfssbas.dtx v3.0i	1995-10-25 ltfloat.dtx v1.1n
General: (DPC) Modify autoloading code,	\@endfloatbox: (CAR) macro added:
change \@undefined 533	to unify code for double and single
1995-10-20 ltfsssrc.dtx v3.0e	versions 981
General: (DPC) Modify autoloading	\end@dblfloat: (CAR) unify code for
code 633	double and single versions 980
1995-10-22 ltfssbas.dtx v3.0j	\end@float: (CAR) unify code for
General: (RmS) New size function	double and single versions 979
macro \@genb@sfcnt needs to be	1995-10-25 ltidxglo.dtx v1.1d
disabled at \@document. 533	General: Doc cleanup 995
1995-10-22 ltfsssrc.dtx v3.0f	1995-10-25 ltsect.dtx v1.0q
General: Added 'genb' and 'sgenb' size	\@subparagraphmark: Use \@let not
functions to support new DC font	\@def to save space. 966
naming scheme. 633	1995-10-27 ltpictur.dtx v1.1f
1995-10-23 lttab.dtx v1.1f	General: Move initialization to kernel
\@settab: (CAR)Ensure that	from autoloading file 949
\@hightab increases by at most	1995-10-31 ltboxes.dtx v1.0x
one 904	\@finalstrut: Add \@nobreak in horiz
\@startline: (CAR)Ensure that	mode to allow hyphenation.
\@nxttabmar is never larger than	internal/1931 895
\@hightab 902	1995-11-01 fontdef.dtx v2.2m
\@poptabs: (CAR)Ensure that	General: add \@nfss@catcodes for
\@curtab is never larger than	internal/1932 726
\@hightab 905	1995-11-01 ltdirchk.dtx v1.0n
\@tabbing: (CAR)Make \@hightab	General: Initialise \@addtofilelist to
consistently a local variable 904	\@gobble 4
1995-10-24 ltfiles.dtx v1.1a	1995-11-01 ltfinal.dtx v1.0x
\@document: Removed multiplelabels	General: (DPC) Switch meaning of
switch 454	\@addtofilelist for cfg files .. 1266
Removed refundefined switch ... 454	1995-11-01 ltfssbas.dtx v3.0m
1995-10-24 ltfssbas.dtx v3.0k	\DeclareFontShape@: (DPC) Test for
\@@defaultsubs: macro removed .. 556	\@relax not \@undefined,
\@wrong@fontshape: Make this code	internal/1933 534
inline since it happens only here 555	1995-11-01 ltfssini.dtx v3.0g
1995-10-24 ltmiscen.dtx v1.1b	General: (DPC) Switch meaning of
\@enddocument@kernel@warnings:	\@addtofilelist for cfg files ... 721
Changed logic for producing	1995-11-02 ltfssbas.dtx v3.0n
warning messages and removed	\@wrong@fontshape: (DPC) Remove
switch 822	extra space with \@string for
Use \@refundefined instead of	latex/1676 555
switch 822	

1995-11-02 ltoutenc.dtx v1.7n		1995-11-28 ltfloat.dtx v1.1n	
General: Changed internal name <code>\a</code> to <code>\@tabacckludge</code> to protect against redefinition by malicious users.	488	General: documentation fixes	972
1995-11-07 ltlists.dtx v1.0k		1995-11-28 ltfssbstrc.dtx v3.0g	
<code>\@doendpe</code> : Enclosed <code>\setbox0</code> assignment by a group so that it leaves the contents of box 0 intact.	871	General: documentation fixes	633
1995-11-07 ltoutenc.dtx v1.7o		1995-11-28 ltoutenc.dtx v1.7r	
General: Added <code>\leavevmode</code> at start of <code>\c</code> , otherwise the output routine might be invoked within the macro.	492	General: Added math mode checks to text commands.	479
Changed <code>\char32</code> to <code>\@xxxii</code> (two tokens less).	493	doc fixes	475
Replaced octal number 27 by decimal number 23 to protect against the quote character being active.	493	Renamed <code>\@changed@x@err</code> to <code>\TextSymbolUnavailable</code>	479
Replaced some 0's by <code>\z@</code> (faster).	493	1995-11-29 ltoutenc.dtx v1.7t	
1995-11-10 ltoutput.dtx v1.1s		General: Added <code>\textasciicircum</code> , <code>\textasciitilde</code> , <code>\textbackslash</code> , <code>\textbar</code> , <code>\textgreater</code> and <code>\textless</code>	495
<code>\@shipoutsetup</code> : Command removed	1188	Added <code>\textasciicircum</code> , <code>\textasciitilde</code> , <code>\textregistered</code> and <code>\texttrademark</code>	489
<code>\@writsetup</code> : Command removed In-lined	1188	Added <code>\textbackslash</code> and <code>\textbar</code>	489, 500
1995-11-14 ltclass.dtx v1.0t		Added <code>\textless</code> and <code>\textgreater</code>	489, 500
<code>\@unprocessedoptions</code> : Allow empty option	1073	1995-12-01 ltoutenc.dtx v1.7u	
<code>\@loadwithoptions</code> : macro added	1060	General: Made <code>\SS</code> a Default, rather than having the default point to the OT1 definition.	489
<code>\LoadClassWithOptions</code> : macro added	1060	1995-12-04 ltspace.dtx v1.2k	
<code>\RequirePackageWithOptions</code> : macro added	1061	<code>\nobreakspace</code> : Macro added	446
1995-11-17 ltfssbas.dtx v3.0m		1995-12-04 ltspace.dtx v1.2l	
<code>\@wrong@font@char</code> : (DPC) Macro added. latex/1676	556	<code>\@xobeysp</code> : braces added to definition of tilde	447
<code>\define@newfont</code> : Redefine <code>\typeout</code> latex/1676	549	1995-12-04 preload.dtx v2.4e	
<code>\@wrong@font@shape</code> : Support <code>\@wrong@font@char</code> latex/1676	555	General: Ulrik Vieth. added 12pt OMS and OML preloads /1989	746
1995-11-17 ltoutenc.dtx v1.7p		1995-12-05 ltdefs.dtx 1.2w	
<code>\UseTextSymbol</code> : Support <code>\@wrong@font@char</code> latex/1676	485	<code>\@unexpandable@protect</code> : Removed <code>\unexpandable@noexpand</code> as never used. internal/1733	90
1995-11-18 ltoutenc.dtx v1.7q		1995-12-05 ltfiles.dtx v1.1c	
<code>\UseTextSymbol</code> : Modify message slightly	485	<code>\document</code> : <code>\ignorespaces</code> added for latex/1933	455
1995-11-21 fontdef.dtx v2.2n		1995-12-05 ltfloat.dtx v1.1n	
General: Incorporate changed figures, as in plain.tex	740	<code>\@textsuperscript</code> : Use <code>\ensuremath</code> for latex/1984.	990
1995-11-27 ltfssbas.dtx v3.0n		1995-12-05 ltoutenc.dtx v1.7v	
<code>\nfss@catcodes</code> : Reset hash, for definitions in fd files	551	<code>\@inmathwarn</code> : Changed <code>\TextSymbolUnavailable</code> text	481
		1995-12-06 ltfssbas.dtx v3.0o	
		<code>\nfss@catcodes</code> : Reset hat, for typeouts etc in fd files	551
		1995-12-07 ltbibl.dtx v1.1l	
		<code>\@citex</code> : Restored name of <code>\G@refundefinedtrue</code>	1000

1995-12-07 lfloat.dtx v1.1m	1996-05-17 fontdef.dtx v2.2o
\@textsuperscript: Move \m@th out of the \ensurermath for latex/1984.	General: \@@sqrt removed, at last
990	723, 739
1995-12-07 ltxref.dtx v1.1i	1996-05-17 ltfiles.dtx v1.1f
\@setref: Switch for redefined restored	\nofiles: added \write to \protected@write for latex/2146
799	458
\G@refundefinedtrue: Renamed (back) from \G@refundefined	1996-05-18 ltoutenc.dtx v1.7x
798	General: Produce error if encoding not found. pr/2054
1995-12-11 ltoutenc.dtx v1.7w	518
General: Modified \copyright	1996-05-21 ltoutenc.dtx v1.7y
489	General: Corrected error message (CAR)
1995-12-13 ltdefs.dtx 1.2x	518
\ -: Documentation changed.	1996-05-21 ltsect.dtx v1.0s
110	\@sect: (DPC) Added extra braces for internal/2148
1996-01-10 ltfiles.dtx v1.1d	962
\@iffileonpath: Change argument handling to not require doubled hash. latex/2024	(DPC) Moved brace to allow commands like \MakeUppercase in 6th argument. Changed \par to \endgraf to allow non-long commands. internal/2148
468	962
1996-01-20 ltidxglo.dtx v1.1e	\@ssect: (DPC) Added extra braces for internal/2148
\makeglossary: Make no-op after use pr/2048	965
996	(DPC) Moved brace to allow commands like \MakeUppercase in 4th argument. Changed \par to \endgraf to allow non-long commands. internal/2148
\makeindex: Make no-op after use pr/2048	965
996	1996-05-23 ltoutenc.dtx v1.7z
1996-01-20 ltspace.dtx v1.2m	\@strip@args: \expandafter added to match other changes for latex/2133
\vspacer: Made robust	484
444	\add@accent: macro added. latex/2133
1996-03-25 ltmath.dtx v1.1a	482
\@ensuredmath: Macro added for amslatex/2104	\DeclareTextAccent: Reimplemented using \add@accent to save space latex/2133
856	482
\ensurermath: Reimplement for amslatex/2104	\DeclareTextCompositeCommand: Modified to cope with new \add@accent command: required removal of check for one argument-command
856	483
1996-04-18 ltpage.dtx v1.0i	1996-05-24 ltoutput.dtx v1.1t
General: Improve documentation	\@specialoutput: Check that \@colroom is less than \vsize, indicating that a float has been added
1033	1172
1996-04-22 ltmiscen.dtx v1.1c	Cut-off point changed to 1.5\baselineskip
General: Improve Documentation	1172
819	\@topnewpage: Cut-off point changed to 2.5\baselineskip
1996-04-22 ltspace.dtx v1.2n	1169
General: Documentation Improvements	1996-05-25 ltoutput.dtx v1.1u
430	\@specialoutput: Correct the above check
1996-04-22 lttab.dtx v1.1g	1172
\@tabclassz: (DPC) Extra \hskip keeps tabcolsep in empty columns internal/2122	751
917	
1996-04-23 ltcounts.dtx v1.1d	
General: Documentation improvements	
521	
1996-04-24 ltfiles.dtx v1.1e	
\document: (DPC) Reset \AtBeginDocument eg for latex/1297	
454	
1996-05-08 ltfstrc.dtx v3.0h	
\math@egroup: Use \bgroup instead of \beginngroup to match a kernel change made in 1994!!	
645	
1996-05-09 ltfntcmd.dtx v3.3t	
\check@icr: Default definitions added	
751	

1996-06-03 ltmiscen.dtx v1.1d	1996-07-26 ltfloat.dtx v1.1n
<code>\@verbatim</code> : Exchanged the following two code lines so that <code>\dospecials</code> cannot reset the category code of characters handled by <code>\@noligs</code> 837	<code>\@endfloatbox</code> : remove unnecessary <code>\global before \@minipage...</code> . . . 981
General: Move setting of verbatim font and <code>\@noligs</code> 819	<code>\@savemarbox</code> : remove unnecessary <code>\global before \@minipage...</code> . . . 985
<code>\verb</code> : Put setting of verbatim font after <code>\dospecials</code> so that <code>\dospecials</code> cannot reset the category code of characters handled by <code>\@noligs</code> 842	<code>\@setminipage</code> : remove unnecessary <code>\global before \@minipage...</code> . . . 979
1996-06-10 ltboxes.dtx v1.0y	<code>\@setnobreak</code> : remove unnecessary <code>\global before \@nobreak...</code> . . . 979
<code>\@parboxto</code> : (DPC) Changed <code>\endgraf</code> to <code>\@@par</code> 888	1996-07-26 ltfsbas.dtx v3.0p
1996-06-10 ltsect.dtx v1.0t	<code>\@DeclareMathSizes</code> : use faster <code>\if</code> test 542
<code>\@sect</code> : (DPC) Changed <code>\endgraf</code> to <code>\@@par</code> 962	<code>\nfss@catcodes</code> : omit <code>\relax</code> as not needed 551
<code>\@ssect</code> : (DPC) Changed <code>\endgraf</code> to <code>\@@par</code> 965	1996-07-26 ltfsdcl.dtx v3.0e
1996-06-13 ltdirchk.dtx v1.0r	<code>\@init@restore@version</code> : Removed <code>\ifrestore@version</code> switch and replaced by <code>\@init@restore@version</code> 667
General: documentation improvements mainly from internal/2174 1	1996-07-26 ltfsstrc.dtx v3.0i
1996-06-14 lttab.dtx v1.1h	<code>\@init@restore@glb@settings</code> : macro added replacing <code>\if@inmath</code> switch 644
<code>\@tabclassz</code> : (DPC) Change both <code>\z@skip</code> to 1sp for latex/2160 917	1996-07-26 ltlists.dtx v1.0l
1996-06-22 ltspace.dtx v1.2o	<code>\@item</code> : Remove unnecessary <code>\global</code> before <code>\@minipage...</code> 873
General: Documentation of problems added 430	Remove unnecessary <code>\global before</code> <code>\@nobreak...</code> 874
1996-07-10 ltfinal.dtx v1.0y	1996-07-26 ltmath.dtx v1.1b
<code>\@toks</code> : Free up memory from scratch registers /2213 1277	General: Removed <code>\global before</code> <code>\@ignoretrue</code> in various places. . . 844
1996-07-19 ltoutenc.dtx v1.8a	1996-07-26 ltmiscen.dtx v1.1e
<code>\@strip@args</code> : Use char 0 not @ as carrier for <code>\lowercase</code> /2197 . . . 485	<code>\@ignorefalse</code> : put <code>\global</code> into definition 820
1996-07-26 ltboxes.dtx v1.0z	<code>\@begin</code> : remove <code>\global before</code> <code>\@ignore...</code> 828
<code>\if@minipage</code> : put <code>\global</code> into definition 890	<code>\@end</code> : remove <code>\global before</code> <code>\@ignore...</code> 830
1996-07-26 ltclass.dtx v1.0u	<code>\@ignorespacesafterend</code> : user level macro added 820
<code>\@classoptionslist</code> : made only preamble 1043	1996-07-26 ltoutput.dtx v1.1v
<code>\@unusedoptionlist</code> : made only preamble 1043	<code>\@testfp</code> : remove <code>\global before</code> <code>\@test...</code> 1230
1996-07-26 ltdefns.dtx v1.2y	<code>\@xtryfc</code> : remove <code>\global before</code> <code>\@test...</code> 1198
<code>\@reargdef</code> : third arg picked up by <code>\@yargdef</code> 85	<code>\@ztryfc</code> : remove <code>\global before</code> <code>\@test...</code> 1199
<code>\@renew@command</code> : use <code>\noexpand</code> instead of <code>\string</code> 86	General: put <code>\global</code> into definition remove <code>\global before \@test...</code> . . 1162
use <code>\relax</code> in place of empty arg . 86	<code>\@clearpage</code> : add number of missing percents 1165
<code>\@renew@environment</code> : use <code>\relax</code> in place of empty arg 87	1996-07-26 ltplain.dtx v1.1t
	<code>\sh@ft</code> : replace <code>\dimen\z@</code> by <code>\dimen@</code> 32

1996-07-26 ltsect.dtx v1.0u	tests	1166
\@starttoc: removed \global before	1996-10-04 ltclass.dtx v1.0v	
\@nobreak	\RequirePackageWithOptions: Reset	
\@xsect: Removed \global before	\@unprocessedoptions for	
\@nobreak	/2269	1061
1996-07-26 ltspace.dtx v1.2p	1996-10-05 ltfiles.dtx v1.1h	
\if@nobreak: put \global inside	\@clubpenalty: Added setting its	
definition	value	453
1996-07-27 ltfssbas.dtx v3.0q	1996-10-08 ltfntcmd.dtx v3.3u	
General: \if@inmath switch removed	\DeclareTextFontCommand: Removed	
1996-07-27 ltspace.dtx v1.2q	\check@icr when in vmode since	
General: Further documentation of	it causes various errors (see	
problems	pr/2157)	749
1996-07-27 ltspace.dtx v1.2r	1996-10-21 lttab.dtx v1.1i	
General: Correct documentation of	\@array: Use \set@typeset@protect	910
problems	General: Moved the code associated	
1996-08-02 ltfloat.dtx v1.1o	with \mkpream into the group	
\@xympar: Remove \global before	provided by the box, for	
\@ignore	robustness (latex/2183)	909
1996-08-02 ltsect.dtx v1.0v	\multicolumn: Make \multicolumn	
\@afterheading: Removed \global	long (latex/2180)	912
before \@nobreak	\tabbing: Moved the \indent so that	
1996-08-02 ltspace.dtx v1.2s	the \everypar can remove it when	
\@Esphack: Remove \global before	necessary; this is needed because	
\@ignore	the code for items in lists has	
1996-08-25 ltfssbas.dtx v3.0r	changed (see pr/22111)	904
\nfss@catcodes: Reset the acute,	1996-10-23 ltlists.dtx v1.0m	
grave and double quote chars as	\@item: \@nobreak . . . moved into the	
well	\everypar and not executed	
1996-09-21 ltoutput.dtx v1.1w	unconditionally, see above	874
\@writsetup: Added	\kern . . . changed to \setbox . . .	873
\@parboxrestore and made	Added setting of \@clubpenalty and	
consequent deletions: wait for the	set \@nobreakfalse only when	
howls of protest	necessary	874
1996-09-25 ltdirchk.dtx v1.0t	1996-10-23 ltsect.dtx v1.0x	
General: Move ltxcheck to separate file	\@xsect: Replaced \hskip . . . with	
1996-09-28 ltmiscen.dtx v1.1f	\setbox . . . as used in	
\@xobeysp: Moved to ltspace.dtx . .	\@afterheading	963
1996-09-28 ltspace.dtx v1.2t	1996-10-24 ltboxes.dtx v1.1a	
\@xobeysp: Moved from ltmiscen.dtx	\@arrayparboxrestore: Added local	
and redefined to use	settings of flags: dangerous!	889
\@nobreakspace	\@iiminipage: Use it or lose it	
1996-09-29 ltfiles.dtx v1.1g	(@setminpage): Frank will want to	
\document: Added disabling of	lose it	891
\@nodocument	1996-10-24 ltfloat.dtx v1.1p	
1996-09-29 ltoutput.dtx v1.1x	\@floatboxreset: Added local	
\newpage: Checks for noskipsec and	settings of flags: dangerous!	979
inlabel added	\@marginparreset: Added local	
1996-09-29 ltsect.dtx 1.0w	settings of flags: dangerous!	985
\@noskipsectrue: Added	\@xfloat: Added \@nodocument to	
documentation	trap floats in the preamble	976
1996-09-30 ltoutput.dtx v1.1y	1996-10-24 ltoutput.dtx v1.1z	
\newpage: Checks for noskipsec and	\@addtocurcol: Added \@nobreak, etc	
inlabel removed pending further	as appropriate	1204, 1210

<code>\@specialoutput</code> : Added <code>\nobreak</code> as appropriate	1174	1996-11-04 <code>ltlists.dtx v1.0q</code>	
<code>\@topnewpage</code> : Added <code>\@nodocument</code> to trap <code>\twocolumn</code> in the preamble	1168	<code>\@trivlist</code> : Moved check for missing item: only checked when not inlabel flag is false	868
<code>\newpage</code> : Better checks for noskipsec and inlabel added, plus nobreak	1166	1996-11-05 <code>ltfiles.dtx v1.1i</code>	
1996-10-25 <code>ltlists.dtx v1.0n</code>		<code>\nofiles</code> : Standard <code>\if@nobreak</code> test added	458
<code>\endtrivlist</code> : Change <code>\indent</code> to <code>\leavevmode</code>	869	1996-11-09 <code>ltmath.dtx v1.1c</code>	
Reset flags explicitly	869	<code>\@ensuredmath</code> : Made long, as it was before. /2104	856
1996-10-25 <code>ltoutput.dtx v1.2a</code>		1996-11-18 <code>ltfssbas.dtx v3.0s</code>	
<code>\newpage</code> : Reset all flags explicitly	1166	<code>\define@newfont</code> : (DPC) lowercase fd file names. internal/1044	550
1996-10-26 <code>ltlists.dtx v1.0o</code>		1996-11-18 <code>ltoutenc.dtx v1.8d</code>	
<code>\endtrivlist</code> : Correct typo	869	General: (DPC) lowercase external file names. internal/1044	518
1996-10-27 <code>ltoutenc.dtx v1.8c</code>		1996-11-20 <code>fontdef.dtx v2.2p</code>	
<code>\@strip@args</code> : Removed macro	483	General: lowercase fd and enc.def file names /1044	723
General: Added <code>\r A</code>	493	1996-11-20 <code>ltvers.dtx v1.0f</code>	
Added		General: Check for old format modified /2319	37
<code>\textasteriskcentered</code>	489, 500	1996-11-23 <code>ltoutenc.dtx v1.8e</code>	
Corrected syntax descriptions	476	General: Corrected description	476
Removed <code>\aa</code> and <code>\AA</code>	488, 493, 495	Extended description	477
1996-10-28 <code>ltplain.dtx v1.1u</code>		1996-11-28 <code>ltvers.dtx v1.0g</code>	
General: (CAR) More doc changes	14	General: Check for old format modified /2319	37
<code>\dotfill</code> : Removed math mode	32	1996-12-06 <code>ltdirchk.dtx v1.0u</code>	
1996-10-29 <code>ltplain.dtx v1.1v</code>		<code>\IfFileExists</code> : *** removed from various messages for GNU Make. internal/2338	10
<code>\dotfill</code> : Got arithmetic correct (CAR)	32	1996-12-06 <code>ltfloat.dtx v1.1r</code>	
1996-10-29 <code>ltspace.dtx v1.2u</code>		<code>\@caption</code> : Call <code>\@setminpage</code> if needed. latex/2318	975
<code>\@gnewline</code> : Added macro	436	1996-12-06 <code>ltfssini.dtx v3.0h</code>	
<code>\@no@lnbk</code> : Macro replaces <code>\@lnbk</code> and <code>\@no@lnbk</code>	434	General: (DPC) Remove *** from messages internal/2338	721
<code>\:</code> : Corrected and rationalised code	434	1996-12-17 <code>ltdefns.dtx v1.0w</code>	
<code>\nolinebreak</code> : Reimplemented both using <code>\@no@lnbk</code>	433	<code>\g@addto@macro</code> : Use <code>\begingroup</code> to save making a mathord	112
1996-10-31 <code>ltfinal.dtx v1.0z</code>		1996-12-20 <code>ltsect.dtx v1.0z</code>	
General: Added extra <code>\lcode</code> , hoping it does no harm in T1 (pr/1969)	1266, 1273	<code>\@dottedtocline</code> : Added <code>\nobreak</code> for latex/2343	970
1996-10-31 <code>ltlists.dtx v1.0p</code>		1997-01-08 <code>fontdef.dtx v2.2q</code>	
<code>\@trivlist</code> : Added check for missing item in outer list	868	General: Use <code>\DeclareMathDelimiter</code> to set delimiter codes	733
1996-10-31 <code>ltsect.dtx v1.0y</code>		<code>\mathparagraph</code> : Define using <code>\DeclareMathSymbol</code>	741
General: Corrected and tidied documentation; removed long lines	958	1997-01-08 <code>ltfiles.dtx v1.1j</code>	
1996-11-03 <code>ltplain.dtx v1.1w</code>		<code>\@include</code> : reset <code>\deadcycles</code> latex/2365	462
<code>\dotfill</code> : Saved tokens by using <code>\hb@xt@</code>	32		
1996-11-04 <code>lterror.dtx v1.2m</code>			
<code>\@nodocument</code> : Always define <code>\@nodocument</code> in kernel, so that it can be cleared by <code>\document.</code>	404		

1997-01-08 ltmath.dtx v1.1d	1997-05-29 ltlogos.dtx v1.1f
<code>\root</code> : (DPC) Remove spurious space tokens from plain \TeX definition /2359 846	<code>\LaTeXe</code> : Added <code>\m@th</code> so that the \LaTeX 2 ϵ logo works with non-zero values of <code>\mathsurround</code> 450
1997-02-05 ltdefns.dtx v1.0x	1997-06-16 ltdirchk.dtx v1.0v
<code>\g@addto@macro</code> : missing percent /2402 112	General: documentation improvements mainly from internal/2520 1
1997-02-21 ltlists.dtx v1.0r	1997-06-16 ltfloat.dtx v1.1s
<code>\@item</code> : <code>\ifvoid</code> check added for <code>\noindent</code> . latex/2414 873	General: documentation fixes 972
1997-03-21 ltcounts.dtx v1.1e	1997-06-16 ltfntcmd.dtx v3.3v
<code>\fnsymbol</code> : Use <code>\mathsection</code> and <code>\mathparagraph</code> . latex/2445 . . . 528	General: Fix typo in documentation. 747
1997-04-14 ltfiles.dtx v1.1k	1997-08-05 ltoutenc.dtx v1.9e
<code>\document</code> : Set the document space factor defaults. latex/2404 454	General: Corrected order of arguments in <code>\UseTextSymbol</code> example. . . . 476
<code>\normalsfcodes</code> : Macro added (from patch file) latex/2404 458	1997-08-29 ltoutenc.dtx v1.9f
1997-04-14 ltoutput.dtx v1.2b	General: Added OT4 encoding, provided by Marcin Woliński. . . 475
<code>\@writsetup</code> : Call <code>\normalsfcodes</code> (from patch file) latex/2404 . . . 1190	1997-09-09 ltdefns.dtx v1.2z
Move <code>\label</code> and <code>\index</code> (from patch file) 1190	<code>\provide@command</code> : Use <code>\begingroup</code> to avoid generating math ords if used in math mode. pr/2573 88
1997-04-24 ltbibl.dtx v1.1m	1997-09-15 ltpictur.dtx v1.1g
<code>\@citex</code> : <code>\@empty</code> to avoid primitive error on empty cite keys. latex/2432 1000	<code>\@getcirc</code> : Warn if lines become invisible pr/2524 943
1997-04-30 ltoutenc.dtx v1.9a	<code>\@picture@warn</code> : Macro added pr/2524 943
General: Changed <code>\textsc</code> to <code>\scshape</code> 490	<code>\@sline</code> : Warn if lines become invisible pr/2524 932
Introduced <code>\textcopyright</code> and modified <code>\copyright</code> 489	1997-10-06 ltcounts.dtx v1.1f
Introduced <code>\textcopyright</code> and modify <code>\copyright</code> 490	<code>\@Roman</code> : Change <code>\@Roman</code> to be fully expandable, so that the result is written properly to files. 528
Modified <code>\textunderscore</code> , removing <code>\mathunderscore</code> 489	<code>\@slowromancap</code> : Macro added. . . . 528
Modified <code>\underscore</code> , removing <code>\mathunderscore</code> 490	1997-10-08 ltlogos.dtx v1.1h
1997-04-30 ltoutenc.dtx v1.9b	<code>\LaTeX</code> : Simplify macro (force loading of suitable math fonts once). . . . 450
General: Added <code>\leavevmode</code> to <code>\textunderscore</code> 489	1997-10-10 ltclass.dtx v1.0y
1997-05-04 ltoutenc.dtx v1.9c	<code>\endfilecontents</code> : <code>\@currentenv</code> in banner 1076
General: Added ‘hex index tabs’ . . . 496	<code>\reserved@c</code> not <code>\verbatim@out</code> to save a csname 1075
Added TS1 encoding v2.2.beta . . . 502	Check for text before or after <code>\end</code> environment. latex/2636 1076
1997-05-07 ltoutenc.dtx v1.9d	Use <code>\@gobbletwo</code> 1075
General: Added <code>\leavevmode</code> to <code>\textcompwordmark</code> 489	1997-10-17 ltfntcmd.dtx v3.3w
1997-05-07 ltspace.dtx v1.2v	<code>\check@nocorr@</code> : Check for vertical mode moved here, from <code>\DeclareTextFontCommand</code> (see PR/2646). 751
<code>\newline</code> : Made completely robust. 435	<code>\DeclareTextFontCommand</code> : Reinstalled <code>\check@icr</code> as check is now done in <code>\check@nocorr@</code> (see PR/2646). 749
1997-05-29 ltfstrc.dtx v3.0j	
General: Replaced <code>\</code> by <code>\MessageBreak</code> , as suggested by Donald Arseneau. 635	

1997-10-20 ltfinal.dtx v1.1a	Added section.	518
\@uclclist: Removed \aa and \AA from \@uclclist as these are macros.	Added textcomp.sty.	475
1997-10-21 ltdefs.dtx v1.2z1	As in OT1, Added \leavevmode at start of \c, otherwise the output routine might be invoked within the macro.	494
\renewcommand: Use \begingroup/\endgroup rather than braces for grouping, to avoid generating empty math atom. . . .	Changed to decimal codes in \ooalign.	86
1997-10-21 ltffsbas.dtx v3.0t	Changed to decimal codes.	500
\define@newfont: Move \makeatletter to \nfss@catcodes.	Documentation changes and additions.	475
\nfss@catcodes: Moved \makeatletter from \try@load@font@shape.	Example corrected, braces removed.	550
1997-11-09 ltoutput.dtx v1.2c	Removed default settings, see next section.	502
\@specialoutput: Remove incorrect code: only one \@emptycol is needed here	1997-12-19 ltoutenc.dtx v1.9i General: Documentation corrections. . . .	1172
\@topnewpage: Documentation of vsize check enhanced	1997-12-20 fontdef.dtx v2.2s General: Added documentation	1167
1997-11-13 ltffsdcl.dtx v3.0f	1997-12-31 ltoutenc.dtx v1.9k General: Further correction	677
\DeclareSymbolFont: (DPC) Really update \group@list don't leave new version in \toks@. latex/2661	1998-01-12 ltoutenc.dtx v1.9k General: Added \ProvidesPackage for textcomp.sty	665
\stepcounter: (DPC) Remove as never used. (Re)defined in ltxcounts	Adding missing braces and \ushape.	665
1997-11-19 ltfloat.dtx v1.1t	1998-01-16 ltoutenc.dtx v1.9m General: fixed decimal codes. latex/2734	991
\@footnotetext: Missing percent, again	1998-03-04 ltdefs.dtx v1.2z2 \@xargdef: Unnecessary \expandafter removed: pr/2758	991
1997-11-19 ltoutput.dtx v1.2d	1998-03-05 ltoutenc.dtx v1.9n General: Added masc/fem ords as in pr/2579	1198
\@vtryfc: Reindent code, to be understandable(DPC).	1998-03-20 ltdefs.dtx v1.2z3 \@thirdofthree: Macro added	1198
1997-11-20 ltffsdcl.dtx v3.0g	1998-03-20 ltoutenc.dtx v1.9o General: Documentation added about order of decls	668
\document@select@group: (DPC) inline use of \stepcounter (faster, and saves a cname per math version as no reset list)	Documentation added for pr/2783	668
\select@group: (DPC) inline use of \stepcounter (faster, and saves a cname per math version as no reset list)	\UndeclareTextCommand: Macro added for pr/2783	666
1997-11-23 ltoutenc.dtx v1.9g	1998-03-20 lttextcomp.dtx v1.9o General: Added various \UndeclareTextCommand declarations for pr/2783	784
General: Use \textperthousand, \textpertenthousand and \textfractionsolidus not \textpermill, \textpertenmill and \textfraction. /2673	Load decls after defaults for speed. . . .	784
1997-12-17 ltoutenc.dtx v1.9h	1998-03-21 ltclass.dtx v1.0z General: Added to documentation of filecontents	1038
General: Added \textperthousand and \textpertenthousand	1998-03-21 ltclass.dtx v1.1a \@providesfile: Allow &. Internal/2702	1053
Added code for textcomp.sty. . . .		518

General: Correct to new onlypreamble command list	1090	1998-07-06 lttab.dtx v1.1l General: Small correction to documentation	897
1998-03-25 ltffsbas.dtx v3.0u <code>\showhyphens</code> : Suppress unnecessary error when used in preamble	560	1998-08-17 ltboxes.dtx v1.1e General: (RmS) Minor Documentation fixes.	878
1998-04-11 fontdef.dtx v2.2t General: Added <code>\mathring</code> accent (pr/2785)	739	1998-08-17 ltclass.dtx v1.1c General: (RmS) Minor documentation fixes.	1038
1998-04-15 fontdef.dtx v2.2u General: Use new syntax for <code>\DeclareMathDelimiter</code>	733	1998-08-17 ltdirchk.dtx v1.0w General: (RmS) Documentation improvements.	1
1998-04-15 ltffssdcl.dtx v3.0h <code>\@xxDeclareMathDelimiter</code> : Macro added (pr/2662)	687	1998-08-17 ltftcmd.dtx v3.3x General: (RmS) Minor documentation fixes.	747
1998-04-17 fontdef.dtx v2.2v General: Reinsert symbol defs for < and > chars.	733	1998-08-17 ltffsbas.dtx v3.0v General: (RmS) Documentation fixes.	533
1998-04-18 fontdef.dtx v2.2w General: Reinsert symbol def for / char.	733	1998-08-17 ltffssdcl.dtx v3.0i General: (RmS) Corrected minor glitches in changes entries.	663
1998-05-07 ltclass.dtx v1.1b <code>\@onefilewithoptions@clashchk</code> : Modify help message for latex/2805	1067	1998-08-17 ltffssini.dtx v3.0i General: (RmS) Minor documentation fixes.	695
1998-05-18 lttab.dtx v1.1j <code>\@endpbox</code> : Use <code>\setlength</code> to set <code>\hspace</code> , so that the changes in the calc package apply here.	920	1998-08-17 ltlogos.dtx v1.1i General: (RmS) Minor documentation fixes.	450
<code>\tabular*</code> : Use <code>\setlength</code> , so that calc extensions apply.	909	1998-08-17 ltmath.dtx v1.1c General: (RmS) Minor documentation fixes.	844
1998-05-20 ltfinal.dtx v1.1b General: Set up lccodes before loading hyphenation files: pr/2639	1265	1998-08-17 ltmiscen.dtx v1.1g General: (RmS) Minor documentation fixes.	819
Set up uc/lccodes after loading hyphenation files: pr/2639	1273	1998-08-17 ltspc.dtx v1.2w General: Documentation fixes.	430
1998-05-28 lterror.dtx v1.2n <code>\@notdefinable</code> : Added message re 'end...' pr/1555	404	1998-08-17 preload.dtx v2.1g General: (RmS) Minor documentation fixes.	744
1998-06-04 ltboxes.dtx v1.1c <code>\@rule</code> : Support calc-expressions	894	1998-09-19 ltoutenc.dtx v1.9r <code>\a</code> : Added <code>\string</code> (pr/2878)	488
1998-06-12 ltoutenc.dtx v1.9p General: Corrected 130 and 131, see pr/2834	505	1998-11-13 lttab.dtx v1.1m <code>\@array</code> : Check for hmode to see if something went wrong during parsing (pr/2884)	910
Renamed <code>\textmacron</code> pr/2840	506	1999-01-05 fontdef.dtx v2.2x General: Need special protection for character > in <code>\changes</code> entry.	723
1998-06-12 ltoutenc.dtx v1.9q <code>\add@accent</code> : Explicitly set <code>\spacefactor</code> after <code>\accent</code> (pr/2877)	482	1999-01-06 ltffsbas.dtx v3.0w <code>\DeclareFontEncoding</code> : Added <code>\LastDeclaredEncoding</code> to support cyrillic integration (pr/2988)	537
1998-06-12 lttextcomp.dtx v1.9p General: Renamed <code>\textmacron</code> pr/2840	780	<code>\LastDeclaredEncoding</code> : Added <code>\LastDeclaredEncoding</code> to	
1998-06-18 lttab.dtx v1.1k General: Small addition to documentation	897		

- support cyrillic integration
(pr/2988) 537
- 1999-01-06 ltoutenc.dtx v1.9r
 $\@strip@args$: New impl for
 latex/2930 484
 General: Minor documentation fix. 505
- 1999-01-07 ltdefs.dtx v1.3a
 $\@ifnextchar$: made long 107
 $\@newenvb$: made long and brace
 optional arg. latex/2896 87
 $\@testopt$: made long and brace
 optional arg. latex/2896 84
- 1999-01-07 ltdefs.dtx v1.3b
 $\@ifnextchar$: extra $\@long$.
 latex/2902 107
- 1999-01-07 ltoutenc.dtx v1.9r
 General: Hackery to allow using
 fontenc several times 520
 Hackery to temp support cyrillic
 uc/lc 518
- 1999-01-13 ltoutenc.dtx v1.9s
 $\@strip@args$: Simplified solution for
 latex/2930 484
- 1999-01-18 ltdefs.dtx v1.3c
 $\@yargd@f$: New implementation DPC
 /2942 85
- 1999-02-09 ltdefs.dtx v1.3d
 $\@yargd@f$: catch bad argument forms
 by re-inserting #3 85
- 1999-02-12 lttextcomp.dtx v3.0j
 $\@legacyoldstylenums$: Use
 $\@rmdefault$ instead of $\@cmm$
 (pr/2954) 756
- 1999-02-24 ltoutenc.dtx v1.9t
 General: Corrected hackery cyrillic
 uc/lc list 518
- 1999-03-01 ltdefs.dtx v1.3e
 $\@ifnextchar$: remove extra $\@long$.
 internal/2967 107
- 1999-04-15 ltpictur.dtx v1.1h
 $\@getlarrow$: Replaced octal number,
 CAR 934
 $\@upvector$: Replaced octal number,
 CAR 935
 General: Replaced octal number,
 CAR 934, 935
 Replaced octal numbers, CAR 922
- 1999-04-19 ltfloat.dtx v1.1u
 $\@caption$: Made caption an error
 outside a float: latex/2815 975
- 1999-04-27 ltboxes.dtx v1.1f
 $\@parboxto$: (CAR) Changed $\@empty$
 to $\@relax$ as flag for natural
 width: pr/2975 888
- 1999-04-29 ltdefs.dtx v1.3f
 $\@yargd@f$: Full expansion and
 conversion needed for digit in new
 version, see pr/3013 85
 New macro added 85
- 1999-06-10 ltoutenc.dtx v1.9u
 General: Ensure that we also forget
 old options (pr/2888) 520
- 1999-06-12 ltoutenc.dtx v1.9v
 General: Extend $\@uclclist$ only
 once 520
- 1999-10-09 ltmath.dtx v1.1e
 $\@active@math@prime$: Macro added,
 see PR 3104. 851
 $\@prime@s$: Introduce
 $\@active@math@prime$ 851
- 1999-10-09 ltoutput.dtx 1.2f
 $\@activechar@info$: Reset definition
 of active prime character (used in
 math mode) 1188
- 1999-10-28 ltoutenc.dtx v1.9w
 $\@add@accent$: Give
 $\@accent@spacefactor$ a default
 definition (pr/3084) 482
- 1999-12-08 ltoutenc.dtx v1.9x
 General: Changed $\@CYRRHOOK$ and
 $\@cyrrhook$ to $\@CYRRHK$ and $\@cyrrhk$
 as name changed in the cyrillic
 bundle for naming consistency
 with other “hook” glyphs. 518
- 2000-01-07 ltmiscen.dtx v1.1h
 $\@verbatim$: Disable hyphenation even
 if the font allows it. 837
- 2000-01-15 ltpictur.dtx v1.1i
 $\@upvector$: Removed space at
 end-of-line, CAR 935
- 2000-01-30 ltfntcmd.dtx v3.3y
 $\@DeclareTextFontCommand$: Use
 $\@hmode@bgroup$ now (pr/3160) 749
- 2000-01-30 ltoutenc.dtx v1.9y
 General: Use $\@hmode@bgroup$ where
 applicable
 (pr/3160) 492–494, 500–502, 505
 $\@add@accent$: Use $\@hmode@bgroup$
 where applicable (pr/3160) 482
 $\@hmode@bgroup$: Macro added 482
- 2000-01-30 ltoutenc.dtx v1.9z
 $\@use@text@encoding$: Macro
 reimplemented (pr/3160) 485
 $\@add@accent$: Macro reimplemented
 (pr/3160) 482
 $\@hmode@start@before@group$: Macro
 added (pr/3160) 486

2000-05-19 ltmiscen.dtx v1.1i	2001-02-16 ltxref.dtx v1.1k
\enddocument: Reset \AtEndDocument for latex/3060 820	\@newl@bel: Added an extra grouplevel (PR3250), jlb 800
2000-05-26 ltpage.dtx v1.0j	2001-05-25 ltclass.dtx v1.1d
\@markright: Reimplementation to fix expansion error (pr/3203). 1036	\@providesfile: Explicitly set catcode of \endlinechar to 10 (pr/3334) 1053
\leftmark: Use \@empty instead of brace group (pr/3203). 1036	2001-05-25 ltdirchk.dtx v1.0x
\markright: Reimplementation to fix expansion error (pr/3203). 1034	General: Explicitly set catcode of \endlinechar to 10 (pr/3334) 4
\rightmark: Use \@empty instead of brace group (pr/3203). 1036	2001-05-28 ltoutenc.dtx v1.93
2000-06-02 ltpage.dtx v1.0k	General: Added composites for compatibility with T1, pr/3295 493
\@markright: Small adjustment to give slightly less expansion, CAR 1036	Changed the effect of \.i, pr/3295 496
\markright: Small adjustment to give slightly less expansion, CAR 1034	2001-06-02 fontdef.dtx v2.2y
Tidied 1.0j reimplementation, CAR 1034	General: Provide default cfg files (pr/3264) 742
2000-07-11 ltmiscen.dtx v1.1j	2001-06-04 fontdef.dtx v2.2z
\enddocument@kernel@warnings: Fix typo in warning 822	General: Guard against math active equal and pipe sign in \models (pr/3333) 738
2000-07-12 ltoutput.dtx 1.2g	Guard against math active equal sign in \Relbar (pr/3333) 738
General: Ensure that rule is in \normalcolor 1237	2001-06-04 ltclass.dtx v1.1e
2000-07-19 ltoutput.dtx v1.2h	\@providesfile: But only if it is a char (pr/3334) 1053
\@writsetup: Reset and restore \@if@newlist for internal/3231 1189	2001-06-04 ltdirchk.dtx v1.0y
2000-08-23 ltfinal.dtx v1.1c	General: But only if it is a char (pr/3334) 4
General: Fix typo in warning 1267	2001-06-04 ltpictur.dtx v1.1j
2000-08-30 ltoutenc.dtx v1.91	\@sline: Don't warn for exactly zero pr/3318 932
\@use@text@encoding: Rearranged but no change to final code, CAR (pr/3160) 485	2001-06-04 ltvers.dtx v1.0i
\add@accent: Rearranged but no change to final code, CAR (pr/3160) 482	General: Check for old format disabled 37
2000-09-01 ltfinal.dtx v1.1d	2001-06-05 ltoutenc.dtx v1.94
\@errhelp: Set error help empty at very end (pr/449 done correctly). 1277	General: Text composite Commands need kludges for ‘, – see tlb1903.lvt 493
2000-09-24 ltfloat.dtx v1.2b	2001-08-26 ltclass.dtx v1.1f
\end@dblfloat: FMI: use output routine to defer float 980	\@providesfile: Readded setting of space char (pr/3353) 1053
2000-09-24 ltoutput.dtx v1.2b	2002-02-24 ltplain.dtx v1.1x
\@docclearpage: FMI: ensure \docclearpage is called again until all floats are output. 1176	\loggingall: Macro added 33
2000-09-24 ltoutput.dtx v1.2n	\loggingoutput: Macro added 33
\@addtocurcol: FMI: test for wide float was in wrong place 1203	\showoutput: Use newly added \loggingoutput 33
2001-01-07 ltoutput.dtx v1.2j	\tracingall: Use newly added \loggingoutput 33
\@writsetup: And do it in the right macro (pr/3286) 1189	2002-06-16 ltoutenc.dtx v1.95
	General: Added \textbardbl (pr/3400) 500
	Added default for \textbardbl (pr/3400) 489

2002-06-17 ltoutenc.dtx v1.95	2004-01-03 ltoutenc.dtx v1.99b
General: Corrected <code>\c</code> for T1	General: Added <code>\textogonekcentered</code>
(pr/3442) 494	(pr/3532) 494
Definition of <code>\textexclamdown</code>	Added composites for <code>\k</code> (pr/3532) 499
changed (pr/3368) 492	Use <code>\ooalign</code> for <code>\k</code> (pr/3532) . . 494
Definition of <code>\textquestiondown</code>	2004-01-04 ltbibl.dtx v1.1p
changed (pr/3368) 492	<code>\nocite</code> : Changed error message . 1001
2002-06-18 ltoutenc.dtx v1.95	2004-01-04 ltoutenc.dtx v1.99c
General: Changed def for	General: More adjustments for ogonek
<code>\textregistered</code> to avoid small	(pr/3532) 494
caps (pr/3420) 490	2004-01-23 ltdefs.dtx v1.1g
2002-10-01 ltfloat.dtx v1.1v	<code>\@newenva</code> : Use kernel version of
<code>\thempfootnote</code> : Use braces around	<code>\@ifnextchar</code> (pr/3501) 87
<code>\itshape</code> to keep font change local	<code>\@testopt</code> : Use kernel version of
(pr/3460). 989	<code>\@ifnextchar</code> (pr/3501) 84
2002-10-02 ltfssbas.dtx v3.0x	<code>\@xargdef</code> : Use kernel version of
<code>\DeclareFontSubstitution</code> : Adding	<code>\@ifnextchar</code> (pr/3501) 84
<code>\LastDeclaredEncoding</code>	<code>\@xdblarg</code> : Use kernel version of
introduced a bug as on some	<code>\@ifnextchar</code> (pr/3501) 108
occasions that macro name was	2004-01-23 ltdefs.dtx v1.3g
stored in the internal lists instead	<code>\kernel@ifnextchar</code> : Added macro
of the actual encoding. (pr/3459) 537	(pr/3501) 108
2002-10-28 ltlists.dtx v1.0s	2004-01-28 ltclass.dtx v1.1g
<code>\endtrivlist</code> : Check for math mode	<code>\@providesfile</code> : Use kernel version of
(pr/3437) 869	<code>\@ifnextchar</code> (pr/3501) 1053
2002-10-28 ltoutenc.dtx v1.96	2004-01-28 ltvers.dtx v1.0k
General: coding change, to follow bug	General: Check for old format made 5
fix by DEK in plain.tex	years (pr/3601) 37
(pr/3469) 493, 502	2004-02-02 fontdef.dtx v2.3
2002-12-13 ltbibl.dtx v1.1n	General: Many things from here on
<code>\@citex</code> : Added <code>\leavevmode</code> in case	made robust 737
citation is at start of paragraph	2004-02-02 ltoutenc.dtx v1.99
(pr/3486) 1000	General: Added <code>\textbigcircle</code> . . 500
2003-01-01 ltfntcmd.dtx v3.3z	2004-02-04 fontdef.dtx v2.3a
General: Code checked and	General: Added bigtriangle synonyms
documentation extended by Chris 749	for stmaryrd 735
2003-05-18 ltbibl.dtx v1.1o	2004-02-04 ltspc.dtx v1.3
<code>\nocite</code> : Check if we are after	<code>\nobreakdashes</code> : Macro added 446
<code>\document</code> 1001	2004-02-06 ltoutenc.dtx v1.99d
2003-08-27 ltpictur.dtx v1.1k	<code>\@inmathwarn</code> : New command added
<code>\@bezier</code> : added missing displacement	to fix severe bug: pr/3563 480
pr/3566 951	2004-02-07 ltoutput.dtx v1.2l
<code>\@sline</code> : check for <code>\@linechar</code> being	<code>\@docclearpage</code> : Empty kludgeins box
empty pr/3570 932	if necessary, pr/3528 1176
2003-10-13 ltfinal.dtx v1.1e	2004-02-13 ltoutenc.dtx v1.99e
General: Added extra <code>\lccode</code> for <code>\-</code>	General: Documentation fixes: typos 475
and <code>\textcompwordmark</code> 1266	2004-02-15 ltbibl.dtx v1.1q
2003-12-16 ltoutput.dtx v1.2k	<code>\@cite@ofmt</code> : Added hook with
<code>\@makecol</code> : Ensure that <code>\@elt</code> has a	default value <code>\hbox</code> 1002
defined state (pr/3586) 1178	<code>\@citex</code> : Changed to use a hook with
2003-12-30 ltpictur.dtx v1.1j	default value <code>\hbox</code> 1000
<code>\@getcirc</code> : issue warning if circle size	2004-02-15 ltspc.dtx v1.3a
can't be met pr/3473 943	<code>\nobreakdashes</code> : Added spacefactor
	setting 446

2005-07-27 ltfssdcl.dtx v3.0j		<code>\maybe@ic</code> : Use switch <code>\ifmaybe@ic</code> instead of <code>\if@tempswa</code>	752
<code>\DeclareMathAlphabet</code> : (MH) Make document commands robust . . .	680	<code>\t@st@ic</code> : Use switch <code>\ifmaybe@ic</code> instead of <code>\if@tempswa</code>	753
<code>\DeclareSymbolFontAlphabet</code> : (MH) Make document commands robust	692	2010-08-17 ltmiscen.dtx v1.1k	
<code>\new@mathalphabet</code> : (MH) Make document commands robust . . .	681	<code>\enddocument</code> : Use braces around <code>\input</code> arg (pr/4124)	821
<code>\non@alpherr</code> : (MH) Change because command is now properly robust	667	2010-08-17 ltmiscen.dtx v1.1l	
<code>\SetMathAlphabet</code> : (MH) Make document commands robust . . .	681	<code>\enddocument</code> : Change of plan: use <code>\@@input</code> instead (pr/4124)	821
2005-09-27 ltoutenc.dtx v1.99g		2011-05-08 ltfssdcl.dtx v3.0n	
General: Replace <code>\sh@ft</code> by <code>\ltx@sh@ft</code>	492, 494, 501	<code>\in@</code> : Simplified thanks to Bruno. . .	663
2005-09-27 ltplain.dtx v1.1y		2011-08-19 ltclass.dtx v1.1i	
<code>\ltx@sh@ft</code> : New macro	32	<code>\ifclasswith</code> : Re-jig definition after more stringent <code>\in@</code> test.	1049
<code>\sh@ft</code> : Macro no longer used but left for compatibility	32	2011-09-03 ltfssdcl.dtx v3.0o	
2005-11-08 ltoutenc.dtx v1.99h		<code>\new@mathversion</code> : (Will) Remove <code>\global</code> before <code>\newcount</code> (unnecessary and caused etex bug).	675
General: Added <code>\ij</code> and <code>\IJ</code> from babel. (pr/3771)	489, 493, 495	2012-01-20 ltplain.dtx v2.0b	
2005-11-10 ltmath.dtx v1.1g		<code>\loggingall</code> : etex tracing if available	33
<code>\l</code> : (MH) Fixed potential problem in <code>\l</code> (pr/3399).	852	2013-07-07 ltclass.dtx v1.1i	
General: (MH) Minor documentation fixes.	844	General: Correctly describe how the date in <code>\@ifpackagelater</code> is used	1041
2006-05-18 ltboxes.dtx v1.1g		2014-04-24 ltoutput.dtx v1.2n	
<code>\@parboxto</code> : Ensure <code>\@parboxto</code> holds the value of <code>\@tempdimb</code> not the register itself (pr/3867)	888	<code>\fl@tracemessage</code> : Renamed internal trace commands; provide as package	1226
2006-09-13 ltoutput.dtx v1.1m		2014-04-27 ltfloat.dtx v1.2b	
General: Ensure that rule is in <code>\normalcolor</code>	1237	<code>\end@dblfloat</code> : Inline the code to allow some coexistence with packages that hook into <code>\end@float</code> and do not know about the algorithm change	980
2007-08-05 ltclass.dtx v1.1h		2014-06-10 ltfloat.dtx v1.2b	
<code>\@fileswithoptions</code> : Prevent loss of brackets PR/3965	1063	<code>\end@dblfloat</code> : missing <code>\fi</code> added .	980
2007-08-06 ltcntrl.dtx v1.0h		<code>\newmarks</code> : macro added	1260
<code>\@fornoop</code> : Really make defs long . .	396	<code>\newXeTeXintercharclass</code> : macro added	1261
2007-08-31 ltfssdcl.dtx v3.0l		2014-12-30 ltfloat.dtx v1.2a	
<code>\SetSymbolFont@</code> : Font warning changed to info for encoding change (pr/3975)	679	<code>\@textsubscript</code> : Command added (latexrelease)	991
2009-09-24 ltvers.dtx v1.0l		<code>\textsubscript</code> : Command added (latexrelease)	990
General: Stop checking for old format	37	2014-12-30 ltfssbas.dtx v3.0y	
2009-10-20 ltfssdcl.dtx v3.0m		<code>\mathgroup</code> : move allocation to ltplain.	533
<code>\in@</code> : More robust thanks to Heiko.	663	2014-12-30 ltoutput.dtx v1.2m	
2009-10-28 lttextcomp.dtx v1.99k		General: Command updated (latexrelease)	1237
General: Added Latin Modern and TeX Gyre subsets	786		
2009-11-04 lttextcomp.dtx v1.99l			
General: Added more Latin Modern and TeX Gyre subsets	786		
2009-12-14 ltfntcmd.dtx v3.4a			
<code>\ifmaybe@ic</code> : Macro added	752		

2014-12-30 ltplain.dtx v2.0a	
\@alloc: macro added	19
\@alloc@chardef: macro added . . .	18
\@alloc@top: macro added	18
\@ch@ck: macro added	19
\extrafloats: macro added	20
\newlanguage: New engine-specific allocation scheme (latexrelease) . .	17
2014-12-30 ltspace.dtx v1.3b	
\@: \@ discards spaces when moving (pr3039)(latexrelease)	447
2015-01-03 ltdefs.dtx v1.4a	
\typein: use modified definition in luatex	82
2015-01-03 ltdirchk.dtx v1.1	
General: Enable extra primitives when LuaTeX is used	3
2015-01-03 ltfinal.dtx v2.0a	
General: Skip resetting codes with Unicode engines	1273
Unicode data loading added . . .	1264
2015-01-07 ltvers.dtx v1.0n	
\@check@IncludeInRelease: macro added	39
2015-01-08 ltboxes.dtx v1.1h	
\framebox: Make Robust (latexrelease)	886
\makebox: Make Robust (latexrelease)	879
\parbox: Make Robust (latexrelease)	888
\raisebox: Make Robust (latexrelease)	894
\rule: Make Robust (latexrelease) .	893
\savebox: Make Robust (latexrelease)	883
2015-01-08 ltdefs.dtx v1.4a	
\MakeRobust: Added macro	93
2015-01-08 ltlength.dtx v1.1c	
\setlength: to ensure first length argument is terminated. (latexrelease)	531
2015-01-08 ltmath.dtx v1.1h	
\): Make Robust (latexrelease)	852
\]: Make Robust (latexrelease)	852
2015-01-09 ltffsini.dtx v3.1a	
\em: Allow \emph to produce small caps (latexrelease)	715
\eminershape: macro added (latexrelease)	715
2015-01-09 ltspace.dtx v1.1h	
\addpenalty: Donald Arseneau's fix from PR/377703 (latexrelease) . .	442
2015-01-10 ltcounts.dtx v1.1h	
\@fnsymbol: Unse \TextOrMath (latexrelease)	529
\@stpelt: Reset all within counters in one go (latexrelease)	523
2015-01-11 ltcounts.dtx v1.1h	
\TextOrMath: Add command to solve robustness issues (pr/3752) (latexrelease)	529
2015-01-11 ltfloat.dtx v1.2b	
\@dblfloatplacement: float order in 2-column (latexrelease)	982
\@xfloat: Check for valid option (latexrelease)	976
\end@dblfloat: float order in 2-column (latexrelease)	980
2015-01-11 ltffsbas.dtx v3.0y	
\@DeclareMathSizes: Allow arbitrary units (latexrelease)	542
2015-01-11 ltspace.dtx v1.3d	
\@Esphack: Allow hyphenation (Donald Arseneau pr/3498) (latexrelease)	439
\@esphack: Allow hyphenation (Donald Arseneau pr/3498) (latexrelease)	438
2015-01-14 ltoutput.dtx v1.2n	
\@addtocurcol: float order in 2-column (latexrelease)	1202
\@addtodblcol: float order in 2-column (latexrelease)	1217
\@addtonextcol: float order in 2-column (latexrelease)	1212
\@docclearpage: Empty kludgeins box if necessary, pr/3528	1175
float order in 2-column (latexrelease)	1175
\@startdblcolumn: float order in 2-column (latexrelease)	1196
\@xtryfc: float order in 2-column (latexrelease)	1198
\@ztryfc: float order in 2-column (latexrelease)	1199
2015-01-14 ltspace.dtx v1.3e	
\addpenalty: Avoid adding redundant skips (DPC)	442
2015-01-17 ltvers.dtx v1.0m	
\@check@IncludeInRelease: modified with \@currname	39
2015-01-19 ltvers.dtx v1.0o	
\@check@IncludeInRelease: Optional argument	39
2015-01-20 ltoutput.dtx v1.2m	
\fl@tracemessage: Reset \IncludeInRelease flags	1227

2015-01-22 ltvers.dtx v1.0p	2015-02-21 ltplain.dtx v2.0e
General: Preserve any <code>\everyjob</code>	General: Removed autoload code . . . 14
material inserted by a loader (<code>.ini</code>	2015-02-21 lttab.dtx v1.1n
file) 38	General: Removed autoload code . . . 897
2015-01-23 ltfinal.dtx v2.0b	2015-02-21 ltvers.dtx v1.0r
<code>\newmarks</code> : use reserved count 256 1260	General: Removed autoload code . . . 37
<code>\newXeTeXintercharclass</code> : use	2015-02-21 ltvers.dtx v1.0w
reserved count 257 1261	<code>\@check@IncludeInRelease</code> : set
2015-01-23 ltplain.dtx v2.0c	<code>\@currname</code> empty here (in case
<code>\extrafloats</code> : reserve counts 256–265 20	<code>\IncludeInRelease</code> input early) . . . 39
2015-01-24 ltfinal.dtx v2.0c	2015-02-22 ltfsscnp.dtx v3.0e
General: Skip T1-code entirely with	General: Moved all code into
Unicode engines 1264	<code>latexrelease</code> - obsolete commands
2015-02-03 ltfinal.dtx v2.0d	are no longer automatically part of
General: Set <code>\lccode</code> for - with	the kernel 658
Unicode engines 1265	2015-03-02 ltplain.dtx v2.0f
2015-02-16 ltoutenc.dtx v1.99m	<code>\e@mathgroup@top</code> macro added 19
General: Added <code>\textcommabelow</code>	<code>\newlanguage</code> : allow 255 math groups
<code>latex/4414</code> 490	in Unicode engines 17
2015-02-16 ltoutenc.dtx v1.99n	2015-03-10 ltplain.dtx v2.0g
General: Added <code>\textcommaabove</code> 491	<code>\hideoutput</code> : macro added 36
Added composites for <code>ç</code> 499	<code>\loggingall</code> : Reorganize to be less
Added composites for <code>\c</code> 494	noisy 33
2015-02-16 ltttextcomp.dtx v1.99m	<code>\tracingnone</code> : macro added 34
General: Added <code>lmtt</code> (Heiko Oberdiek)	2015-03-12 ltoutput.dtx v1.2m
<code>latex/4415</code> 786	General: initialise <code>\@dbldeferlist</code>
2015-02-19 ltvers.dtx v1.0q	again 1163
<code>\@check@IncludeInRelease</code> : Swap	2015-03-18 ltfssdcl.dtx v3.0q
argument order 39	<code>\DeclareSymbolFont</code> : Restrict Symbol
2015-02-20 ltplain.dtx v2.0d	fonts to 0-15 677
<code>\loggingall</code> : Spell commands	<code>\document@select@group</code> : Introduce
correctly :-)) 33	<code>\e@mathgroup@top</code> 668
2015-02-21 ltdefs.dtx v1.4b	<code>\select@group</code> : Introduce
General: Removed autoload support 80	<code>\e@mathgroup@top</code> 666
2015-02-21 ltterror.dtx v1.2o	2015-03-26 ltfinal.dtx v2.0d
General: Removed autoload support 397	General: Use renamed
2015-02-21 ltfiles.dtx v1.1m	<code>unicode-letters.def</code> 1264
General: Removed autoload support 451	2015-04-07 ltfssbas.dtx v3.1a
2015-02-21 ltfssbas.dtx v3.0z	<code>\wrong@fontshape</code> : Try loading fd file
General: Removed autoload code 533	if family has changed 554
2015-02-21 ltfsscnp.dtx v3.0d	2015-04-28 ltfinal.dtx v2.0f
General: Removed autoload code 658	<code>\newXeTeXintercharclass</code> : define
2015-02-21 ltfssdcl.dtx v3.0p	<code>\xe@alloc@intercharclass</code> for
General: Removed autoload code 663	compatibility with older xelatex
2015-02-21 ltfsstrc.dtx v3.0k	initialization 1261
General: Removed autoload code 633	2015-05-10 ltlists.dtx v1.0t
2015-02-21 ltoutenc.dtx v1.99m	<code>\@doendpe</code> : Explicitly reset
General: Removed autoload code 475	<code>\clubpenalty</code> before clearing
2015-02-21 ltoutput.dtx v1.2n	<code>\everypar</code> ; see also pr/0462 and
General: Removed autoload code 1152	pr/4065 871
<code>\f@depth</code> : macro added (latexrelease) 1175	2015-06-19 ltfinal.dtx v2.0g
2015-02-21 ltpictur.dtx v1.1k	<code>\e@alloc@intercharclass@top</code> : Use
General: Removed autoload code 922	-1 for first range to get contiguous
	allocation 1261

\newmarks: Use -1 for first range to get contiguous allocation	1260	module_warning: Function added . . .	56
2015-06-19 ltplain.dtx v2.0h		modules: Function modified	54
General: delete spurious old definition of \newtoks	23	create_callback: Function added . .	67
\@alloc: extra braces in case arguments not single token	19	provides_module: Function added . .	54
\newlanguage: Use -1 for first range to get contiguous allocation	17	luatexbase: Table added	54
2015-06-23 ltfinal.dtx v2.0h		2015-10-02 ltdirchk.dtx v1.2a	
General: set \patch@level in ltvers rather than in ltfinal/ltpatch . .	1276	General: Allow backing out of unprefixed names	3
2015-06-23 ltvers.dtx v1.0t		2015-10-02 ltluatex.dtx v1.0b	
General: set \patch@level in ltvers rather than in ltfinal/ltpatch	37	General: Fix backing out of T _E X code	53
2015-08-06 ltplain.dtx v2.0i		2015-10-02 ltluatex.dtx v1.0c	
\extrafloats: Add \string in case argument is not an unexpandable primitive	20	General: Allow backing out of Lua code	53
2015-08-23 ltdirchk.dtx v1.2		2015-10-02 ltluatex.dtx v1.0e	
General: Do not use luatex prefix	3	uninstall: Function added	72
2015-08-23 ltvers.dtx v1.0v		2015-10-03 ltluatex.dtx v1.0f	
General: Allow negative patchlevel for pre-release	38	provides_module: use luatexbase_log	54
2015-08-30 ltplain.dtx v2.1a		2015-10-27 ltplain.dtx v2.1b	
\newinsert: new \newinsert implementation	22	\extrafloats: Use global assignment when switching to extended range	20
2015-09-205 ltoutput.dtx v1.3a		2015-11-07 ltspace.dtx v1.3f	
General: extended \@freelist	1162	\@esphack: Only space if there is no space at the end of the hlist latex/4443	438
2015-09-24 ltluatex.dtx v1.0a		2015-11-14 ltluatex.dtx v1.0g	
call_callback: Function added	67	General: Track LuaT _E X changes for (new)token.create	56
callback_register: Function modified	64	2015-11-18 ltplain.dtx v2.2a	
callback_descriptions: Function added	71	\newlanguage: Extended stream allocation in luatex (0.85)	17
\catcodetable@atletter: Macro added	49	2015-11-19 ltplain.dtx v2.2b	
\catcodetable@initex: Macro added	49	\newlanguage: Only extend allocation of write streams (see luatex list) . .	17
\catcodetable@latex: Macro added .	49	2015-11-27 ltluatex.dtx v1.0h	
\catcodetable@string: Macro added	49	callback_descriptions: Match test in in-callback latex/4445	71
add_to_callback: Function added . .	68	in_callback: Guard against undefined list latex/4445	71
remove_from_callback: Function added	70	2015-11-29 ltluatex.dtx v1.0i	
new_attribute: Function added	57	General: Declare this as local before used in the module error definitions (PHG)	54
disable_callback: Function added . .	71	call_callback: Check name is not nil in error message (PHG)	67
in_callback: Function added	71	create_callback: Check name is not nil in error message (PHG)	67
\newattribute: Macro added	49	2015-12-02 ltluatex.dtx v1.0j	
\newcatcodetable: Macro added	49	General: Adjust hashtokens to store the result of tex.hashtokens()), not the function (PHG)	56
\newluabytecode: Macro added	52	Assorted typos fixed (PHG)	46
\newluachunkname: Macro added	52	Declaration/use of first_head fixed (PHG)	55
\newluafunction: Macro added	51		
\newwhatsit: Macro added	51		
module_error: Function added	56		
module_info: Function added	56		

Remove nonlocal iteration variables (PHG)	46	<code>\DeclareMathAccent</code> : Check for <code>mathaccent</code> not <code>\mathaccent</code> . . .	683
Remove unreachable code after calls to <code>error()</code> (PHG)	46	<code>\DeclareMathRadical</code> : Check for <code>radical</code> not <code>\radical</code>	691
2015-12-02 <code>ltxuatex.dtx</code> v1.0k		<code>\DeclareMathSymbol</code> : Check for <code>mathchar</code> not <code>\mathchar</code>	686
General: resolve name and <code>i.description</code> (PHG)	65	2016-03-13 <code>ltxuatex.dtx</code> v1.0n	
<code>call_callback</code> : Give more specific error messages (PHG)	67	General: <code>contribute_filter</code> added . . .	63
<code>add_to_callback</code> : Give more specific error messages (PHG)	68	<code>insert_local_par</code> added	63
<code>remove_from_callback</code> : adjust initialization of <code>cb local</code> (PHG) . .	70	2016-03-29 <code>ltpictur.dtx</code> v1.1l	
Give more specific error messages (PHG)	70	<code>\@oval</code> : add setting of line tests	944, 945
<code>create_callback</code> : Give more specific error messages (PHG)	67	initialise tests	944
2015-12-10 <code>ltxfinal.dtx</code> v2.0i		<code>\@ovhorz</code> : use glue not leaders if horizontal line not required	947
General: Use new common Unicode data loaders	1264	<code>\@ovvert</code> : use glue not leaders if vertical line not required	946
2015-12-18 <code>ltxuatex.dtx</code> v1.0l		<code>\if@ovhline</code> : macro added (<code>latex/4452</code>)	944
General: Load Unicode data from source	49	<code>\if@ovvline</code> : macro added (<code>latex/4452</code>)	944
2016-01-04 <code>ltxfinal.dtx</code> v2.0j		2016-04-22 <code>ltxfinal.dtx</code> v2.0q	
General: Do not set up inter character classes for XeTeX	1264	<code>\e@alloc@intercharclass@top</code> : XeTeX 0.99996 has 4096 char classes not 256	1261
<code>\e@alloc@intercharclass@top</code> : Start allocation at one not three	1261	2016-06-19 <code>ltxoutenc.dtx</code> v1.99m	
2016-01-05 <code>ltxfinal.dtx</code> v2.0k		General: OT1 definition (was duplicate T1 definition)	494
<code>\e@alloc@intercharclass@top</code> : Remove duplicated code	1261	2016-06-20 <code>ltxclass.dtx</code> v1.1j	
2016-01-05 <code>ltxfinal.dtx</code> v2.0l		General: don't declare as <code>\@onlypreamble</code>	1048
General: Correct <code>latexrelease</code> guards	1264	2016-07-29 <code>ltxplain.dtx</code> v2.2c	
Ensure old definitions for inter-character class toks are available using <code>latexrelease</code>	1264	<code>\extrafloats</code> : use <code>\global \chardef</code>	20
Missing brace	1264	<code>\newinsert</code> : fix for <code>tlb-newinsert-001</code>	22
2016-01-05 <code>ltxfinal.dtx</code> v2.0m		2016-10-02 <code>ltxclass.dtx</code> v1.2a	
General: Undefine XeTeX classes when using patching an older kernel . .	1264	<code>\@ifclasswith</code> : Ignore spaces while checking for option clash	1049
2016-01-05 <code>ltxfinal.dtx</code> v2.0p		<code>\ExecuteOptions</code> : Ignore spaces in argument	1059
General: Only apply XeTeX change if XeTeX is in use	1264	2016-10-15 <code>ltxdirchk.dtx</code> v1.2b	
2016-02-11 <code>ltxuatex.dtx</code> v1.0m		General: Require eTeX	3
General: <code>pdf_stream_filter_callback</code> removed	64	2016-10-15 <code>ltxerror.dtx</code> v1.2p	
<code>process_rule</code> , <code>[hv]pack_quality</code> <code>append_to_vlist_filter</code> added . .	63	General: Require eTeX	397
<code>read_cidmap_file</code> added	62	2016-10-15 <code>ltxfinal.dtx</code> v2.0r	
<code>show_warning_message</code> added . .	63	General: Require eTeX	1260
<code>token_filter</code> removed	63	2016-10-15 <code>ltxfinal.dtx</code> v2.0s	
2016-02-18 <code>ltxssdcl.dtx</code> v3.0r		General: Tidy up status of char 127	1260
<code>\DeclareMathDelimiter</code> : Check for <code>delimiter</code> not <code>\delimiter</code>	688	2016-10-15 <code>ltxfssini.dtx</code> v3.1b	
		General: Require eTeX	695
		2016-10-15 <code>ltxplain.dtx</code> v2.2d	
		General: Require eTeX	14
		2016-10-16 <code>ltxplain.dtx</code> v2.3a	
		<code>\newlanguage</code> : Allow languages up to 16383 in <code>luatex</code>	17

2016-10-19 ltcounts.dtx v1.1j		declare composites with empty base for hat and tilde, use same slots for <code>\textasciicircum ans</code>	
<code>\TextOrMath</code> : Test directly for <code>\protected</code>	529	<code>\textasciitilde</code>	507
2016-11-06 ltplain.dtx v2.3b		declare straight quotes using new <code>\remove@tlig</code> command	507
General: Drop <code>\outer</code> entirely	14		
2016-11-09 ltclass.dtx v2.1b		2017-02-22 ltoutenc.dtx v2.0g	
<code>\@fileswithoptions</code> : Improve <code>\ifx</code> tests PR/4497	1063	General: Fix typo introduced at 2.0f	507
2016-11-17 ltuatex.dtx v1.0p		2017-02-24 ltoutenc.dtx v2.0h	
General: <code>call_</code> edit added	63	General: introduce <code>\DeclareUnicodeAccent</code>	507
2016-12-03 fontdef.dtx v3.0a		<code>\DeclareTextCompositeCommand</code> : add check whether the accent command is defined for this encoding	483
General: (DPC) Default to TU encoding for Unicode TeX engines	725	2017-03-08 ltclass.dtx v1.2c	
<code>\shapedefault</code> : (DPC) Default to TU encoding for Unicode TeX engines	729	General: add <code>\@parse@version@dash</code> to support yyyy-mm-dd as well as yyyy/mm/dd	1048
2016-12-04 ltoutenc.dtx v2.0a		2017-03-09 ltfinal.dtx v2.0t	
General: Added TU encoding	507	<code>\l@nohyphenation</code> : ensure <code>\l@nohyphenation</code> is defined.	1267
2017-01-01 ltoutput.dtx v1.3b		2017-03-09 ltmiscen.dtx v1.1m	
General: make <code>fpmin</code> negative so ignored even if float height is negative	1236	<code>\verbatim</code> : Use <code>\language</code> not <code>\hyphenchar</code>	837
2017-01-10 ltfssbas.dtx v3.2a		<code>\verb</code> : Use <code>\language</code> to stop hyphenation	842
<code>\showhyphens</code> : Add version of <code>\showhyphens</code> that works with XeTeX.	560	2017-03-10 ltfiles.dtx v1.1n	
2017-01-23 ltoutenc.dtx v2.0b		<code>\document</code> : Save language default	454
General: Added TU specific commands in ASCII range pr/4500	507	2017-03-10 ltoutput.dtx v1.3c	
2017-01-24 ltoutenc.dtx v2.0c		<code>\@writsetup</code> : Reset <code>\language</code>	1189
General: Declare TU composites for <code>i</code> and <code>j</code>	507	2017-03-13 ltdefs.dtx v1.5a	
Make <code>\textasteriskcentered</code> U+2217 not U+204E	507	<code>\-</code> : Define <code>\-</code> in terms of <code>\hyphenchar</code>	110
TeX ligature syntax for xetex and luatex reversed	507	2017-03-27 ltdefs.dtx v1.5b	
2017-01-24 ltoutenc.dtx v2.0d		<code>\@dischyph</code> : Define <code>\@dischyph</code> after <code>\-</code>	110
General: Declare macron composites for <code>YyGg</code>	507	2017-03-28 ltuatex.dtx v1.1e	
2017-02-12 ltoutenc.dtx v2.0e		General: <code>glyph_</code> <code>stream_</code> provider added	63
General: Declare fallback code for <code>\textasteriskcentered</code>	507	2017-03-29 ltboxes.dtx v1.3a	
2017-02-18 ltuatex.dtx v1.1c		<code>\@arrayparboxrestore</code> : Reset <code>\lineskiplimit</code>	890
<code>new_attribute</code> : Parameterize count used in tracking	57	2017-04-05 ltoutenc.dtx v2.0i	
<code>new_bytecode</code> : Parameterize count used in tracking	58	<code>\DeclareTextCompositeCommand</code> : Declare accent command if not already declared when declaring a composite.	483
<code>new_chunkname</code> : Parameterize count used in tracking	58	2017-04-10 ltplain.dtx v2.3c	
<code>new_whatst</code> : Parameterize count used in tracking	58	<code>\newlanguage</code> : Correction to code to skip <code>write18</code> in luatex	17
2017-02-19 ltoutenc.dtx v2.0f		2017-04-11 ltoutput.dtx v2.4a	
General: add <code>\@empty</code> to guard against 3rd argument being empty	493	<code>\newpage</code> : account for the depth of the last row of the page	1166
		2017-12-17 ltoutput.dtx v1.4b	
		<code>\@addtonextcol</code> : fix doc guards	1212

2018-01-06 ltxdefs.dtx 1.5c	2018-05-29 ltclass.dtx v1.2j
\@ifundefined: Avoid defining undefined commands to \relax	\endfilecontents: use \curname not \@undefined
106	1077
2018-02-18 ltclass.dtx v1.2d	2018-08-11 ltoutenc.dtx v2.0j
\@ifl@ter: Added 0 up front to make bad data come out as 0.	General: Provide \guillemetleft and \guillemetright
1048	495, 501, 510, 511
General: Introduce rollback concept	2018-08-18 ltuatex.dtx v1.1h
1082	General: append_ to_ vlist_ filter is exclusive
2018-03-08 ltcounts.dtx v1.1k	63
\@ifbothcounters: Interface added	2018-08-24 ltfinal.dtx v2.1f
525	\document@default@language: Add to latexrelease (github/68)
\@removefromreset: Interface added	1267
525	2018-09-02 ltsect.dtx v1.1b
\counterwithin: Interface added . .	\@dottedtocline: Prevent protrusion (https://tex.stackexchange.com/q/172785/10109)
526	970
2018-03-24 ltclass.dtx v1.2e	2018-09-24 fontdef.dtx v3.0b
\pkgcls@use@this@release: Use full file name for old release	General: Start LR-mode if necessary (git/49)
1088	742
2018-03-25 ltfinal.dtx v2.1a	2018-09-24 ltmath.dtx v1.2b
General: default to UTF-8	\smash: Start LR-mode if necessary (git/49)
1268	848
\UseRawInputEncoding: Macro added	\phantom: Start LR-mode if necessary (git/49)
1269	847
2018-03-27 ltclass.dtx v1.2f	2018-09-24 ltspc.dtx v1.3h
\endfilecontents: Use full file name for old release	\enspace: Start LR-mode if necessary (git/49)
1076	449
2018-04-06 ltfinal.dtx v2.1b	\leavevmode@ifvmode: Macro added (git/49)
\UseRawInputEncoding: Undo changes to \DeclareFontEncoding@ and definition of \DeclareUnicodeCharacter	449
1269	2018-09-26 ltxdefs.dtx v1.5e
2018-04-07 ltfinal.dtx v2.1c	\renew@command: Always explicitly generate a space after the curname and not rely on \noexpand to save tokens (git/41)
\UseRawInputEncoding: Undefine \inputencodingname	86
1269	2018-09-26 ltmiscen.dtx v1.1n
2018-04-08 ltclass.dtx v1.2g	\@writefile: Sometimes mask the endline char when writing to files (github/73)
\@ifl@ter: Strip leading spaces from dates.	826
1048	\add@percent@to@temptokena: Sometimes mask the endline char when writing to files (github/73)
2018-04-08 ltclass.dtx v1.2h	826
\@onefilewithoptions: Pass expanded date	\protected@file@percent: Sometimes mask the endline char when writing to files (github/73)
1084	825
2018-04-08 ltfinal.dtx v2.1d	2018-09-26 ltsect.dtx v1.1c
General: Delay full UTF-8 handling to \everyjob	\addcontentsline: Sometimes mask the endline char when writing to files (github/73)
1270	967
2018-04-11 ltcounts.dtx v1.1l	2018-10-10 ltspc.dtx v1.3i
\counterwithin: Correct default (issue/38)	\@esphack: Don't introduce breakpoints if @nobreak is true and after sections
527	438
2018-05-02 ltuatex.dtx v1.1g	
General: find_ sfd_ file removed . . .	
62	
finish_ synctex_ callback added . .	
63	
glyph_ not_ found added	
63	
read_ sfd_ file removed	
62	
2018-05-08 ltclass.dtx v1.2i	
\pkgcls@parse@date@arg: Make suspicious rollback a warning not error: github issue 43	
1085	
2018-05-11 ltfinal.dtx v2.18	
General: Make invalid UTF-8 also safe, for legacy filesystem encodings . .	
1270	

2018-10-11 ltmiscen.dtx v1.1o	font_descriptor_objnum_ - provider added	63
\@@sverb: Provide visible space in \verb* also for XeTeX and LuaTeX (github/69)	make_extensible added	63
\@setupverbvisiblespace: Provide visible space in \verb* also for XeTeX and LuaTeX (github/69)	new_graf added	63
\@verbvisiblespacebox: Provide visible space in \verb* also for XeTeX and LuaTeX (github/69)	page_objnum_provider added	63
\asciispace: Provide visible space in \verb* also for XeTeX and LuaTeX (github/69)	process_pdf_image_content added	63
verbatim*: Provide visible space in \verb* also for XeTeX and LuaTeX (github/69)	wrapup_run added	63
\verbvisiblespace: Provide \verbvisiblespace such that it is usable in normal text (github/70)	2019-07-01 ltclass.dtx v1.3a	838
Provide visible space in \verb* also for XeTeX and LuaTeX (github/69)	\endfilecontents: Support UTF8 and spaces in filecontents environment file name	838
2018-10-21 ltuatex.dtx v1.1i	2019-07-01 ltfiles.dtx v1.2a	
new_luafunction: Function added	\IfFileExists: Support UTF-8	465
2018-11-09 ltbibl.dtx LaTeX2e	\includeonly: Support UTF-8	459
\bibliography: Zap spaces in the argument as BibTeX doesn't support them (github/88)	\set@curr@file: Support UTF-8	464
2018-11-18 ltoutenc.dtx v2.0k	2019-07-09 ltfssbas.dtx v3.2c	
General: Provide \Hwithstroke and \hwithstroke	\DeclareErrorFont: Don't set any \font... macros	553
2018-11-19 ltoutenc.dtx v2.0k	2019-07-09 ltfssini.dtx v3.1c	
General: Added \Hwithstroke and \hwithstroke	General: Explicitly set some defaults	720
2018-11-28 ltoutput.dtx v1.4d	2019-08-22 ltxref.dtx v1.11	
\@combinedblfloats: Unbox \@outputbox to preserve boxing level (github/94)	\labelformat: Commanded moved from varioref.sty	803
2018-12-30 lttab.dtx v1.1p	\Ref: Commanded moved from varioref.sty	803
\@tabclassz: Add extra \hskip to guard against an \unskip at the start of a c-column cell (gh/102)	\refstepcounter: Allow \p@... to have an argument	802
2019-02-07 ltfilehook.dtx v1.1o	2019-08-27 fontdef.dtx v3.0c	
\unqu@tefilef@und: Expand \@filef@und before executing second argument (github/109)	General: Various commands made robust throughout the file	735
2019-02-07 ltfiles.dtx v1.1o	2019-08-27 ltboxes.dtx v1.3b	
\@swaptwoargs: Helper macro added	General: Various commands made robust	878
2019-02-07 ltfssbas.dtx v3.2b	2019-08-27 ltclass.dtx v1.3b	
\define@newfont: Changed wording of warning (github/107)	\endfilecontents: Make various commands robust	1074
2019-06-18 ltuatex.dtx v1.1j	2019-08-27 ltdefs.dtx v1.5f	
General: finish_synctex_callback renamed finish_synctex	General: Make various commands robust	111
	\MakeRobust: Make the assignments global as we may need to apply them inside a group	93
	2019-08-27 ltfilehook.dtx v1.2b	
	\unqu@tefilef@und: Make command robust	1110
	2019-08-27 ltfiles.dtx v1.2b	
	\IfFileExists: Make command robust	465
	2019-08-27 ltfssbas.dtx v3.2d	
	General: Make various commands robust	533

2019-08-27 ltfssdcl.dtx v3.0s	command already robust	686
\DeclareMathAccent : Make math accents robust		683
$\set@mathdelimiter$: Make math delimiters robust		690
2019-08-27 ltfssini.dtx v3.1d	General: Make various commands robust	695
2019-08-27 ltidxglo.dtx v1.1f	General: Make \index and \glossary robust	995
2019-08-27 ltlength.dtx v1.1d	General: Make various command robust	531
2019-08-27 ltlogos.dtx v1.1j	\TeX : Make \TeX command robust	450
2019-08-27 ltmath.dtx v1.2c	General: Make various commands robust	844
2019-08-27 ltmiscen.dtx v1.1p	General: Make various commands robust	819
\begin : Make command robust		828
\end : Make command robust		830
2019-08-27 ltoutput.dtx v1.4e	$\@begindivibox$: Make \AtBeginDvi robust	1164
2019-08-27 ltpage.dtx v1.0l	General: Make various commands robust	1033
2019-08-27 ltpictur.dtx v1.1m	General: Make various commands robust	922
Remove several unnecessary \gdef definitions		922
2019-08-27 ltsect.dtx v1.1d	General: Make various commands robust	958
2019-08-27 ltspace.dtx v1.3j	General: Make various commands robust	430
2019-08-27 lttab.dtx v1.1q	General: Make various commands robust	897
Remove several unnecessary \gdef definitions		897
2019-08-30 lterror.dtx v1.2q	$\conditionally@traceoff$: Macro added	407
$\conditionally@traceon$: Macro added		407
2019-09-09 ltfssdcl.dtx v3.0s	\DeclareMathSymbol : Allow definition if the math symbol was a	
2019-09-11 ltclass.dtx v1.3c	\endfilecontents : Support optional argument for filecontents	1074
2019-09-14 ltfinal.dtx v2.1h	$\@uclclist$: Expand UTF8 chars when case changing (github/177)	1274
2019-09-16 ltxref.dtx v1.1m	General: Correctly revert the $\p@$... change	803
2019-09-21 fontdef.dtx v3.0d	General: Distangle alias (gh/184)	736, 740
2019-10-02 ltxexpl.dtx v0.0	General: Initial version	73
2019-10-02 ltfinal.dtx v2.2	General: Load ltxexpl	1277
2019-10-02 ltluatex.dtx v1.1k	General: linebreak_ filter is exclusive	63
mlist_ to_ hlist is exclusive		63
process_ rule is exclusive		63
2019-10-07 lttab.dtx v1.1q	\extracolsep : This needs to expand	909
2019-10-11 ltfiles.dtx v1.2c	$\set@curr@file$: Remove one brace group	464
2019-10-11 ltfssstrc.dtx v3.0l	$\@font@aliasinfo$: Added 'alias' size function	656
2019-10-18 ltclass.dtx v1.3d	$\load@onefilewithoptions$: Initialize $\dots-h@k$ only when loading the package or class (gh/198)	1065
2019-10-22 ltluatex.dtx v1.1j	General: page_ objnum_ provider and process_ pdf_ image_ content classified data	63
2019-10-25 ltmiscen.dtx v1.1q	$\add@percent@to@emptokena$: Allow unbalanced conditionals in #1 (gh/202)	826
2019-10-26 ltfiles.dtx v1.2d	$\@iffileonpath$: quote on openin	468
\IfFileExists : don't quote name		465
$\IfFileExists@$: quote on openin		466
$\set@curr@file$: remove quotes		464
2019-11-01 ltdirchk.dtx v1.3a	$\filename@parse$: take last . not first	12
2019-11-02 ltmiscen.dtx v1.1s	$\@centercr$: Make $\@centercr$ robust (gh/203)	833
2019-11-02 ltspace.dtx v1.3k	$\@normalcr$: Make also $\@normalcr$ robust	434

2019-11-09 ltfiles.dtx v1.2e	2020-01-22 lttextcomp.dtx v1.0b
\set@curr@file: expand and \string	\tc@subst: The overall default is
before removing quotes 464	\textcompstbsubstdefault not
2019-11-10 ltmiscen.dtx v1.1r	\substdefault 757
\add@percent@to@temptokena: fix to	2020-01-25 fontdef.dtx v3.0f
special comment catcodes	General: Load t1enc.def last (gh/255) 725
(gh/202) 826	2020-01-25 ltoutenc.dtx v2.0m
2019-11-11 ltfiles.dtx v1.2f	General: Load each encoding file only
\iffileonpath: make \@filef@und	once (gh/255) 519
match quoting used on \openin . 468	2020-01-27 ltclass.dtx v1.3g
2019-11-22 ltoutenc.dtx v2.0l	\endfilecontents: Fix typo in error
General: Avoid spurious if fontenc	message 1076
selects LY1 as default encoding	2020-01-28 ltclass.dtx v1.3h
(gh/199) 520	\endfilecontents: Allow spaces in
2019-11-29 ltclass.dtx v1.3e	option string and display only
\@pr@videpackage: Protect package	unknown options not the whole
info text (gh/52) 1052	option list (gh/256) 1074
2019-12-17 fontdef.dtx v3.0e	2020-01-31 ltvers.dtx v1.1e
\mddefault: Set \bfdefault to “b” 728	General: Allow for upcoming format as
\shapedefault: Set \shapedefault	pre-release 0 38
explicitly to “n” 729	2020-02-02 ltluatex.dtx v1.1l
\updefault: Set \updefault to “up” 728	General: Add reverselist callback type 66
2019-12-17 ltfntcmd.dtx v3.4c	glyph_ info added 63
\textssc: Macro added 750	page_ order_ index added 63
2019-12-17 ltfssbas.dtx v3.2e	post_ linebreak_ filter is
\usefont: Don’t call \fontseries or	reverselist 63
\fontshape 544	create_callback: Provide proper
2019-12-17 ltfssini.dtx v3.1e	fallbacks for user-defined callbacks
General: Provide custom series	without user-provided default
settings a la mweights 696	handler 67
\DeclareEmphSequence: Provide \emph	2020-02-05 ltfssini.dtx v3.1g
sequences 714	\DeclareFontSeriesDefault:
2019-12-18 ltoutenc.dtx v2.0m	Clarified error text 697
General: Don’t fake	Corrected misspelled csname
\textcompwordmark; take default	(gh/264) 697
from T1 instead 489	2020-02-05 lttextcomp.dtx v2.0n
\add@accent: Avoid code that breaks	General: Changed the package default
\accent 482	to info (gh/262) 775
2019-12-21 fontdef.dtx v3.0e	Ensure we are on a new format
General: Distangle alias (gh/184) 734–737	(gh/260) 775
2020-01-05 ltclass.dtx v1.3f	2020-02-07 ltfssini.dtx v3.1h
\endfilecontents: Support more	\symbol: XeTeX-specific version to
write streams in LuaTeX gh/238 1074	avoid bug in maths mode. 717
2020-01-11 ltfssini.dtx v3.1f	2020-02-10 ltfssaxes.dtx v1.0c
\rmfamily: Streamlined	\fontseries: Switch
implementation with hook 709	\if@forced@series added 618
\ttfamily: Streamlined	\fontseriesforce: Switch
implementation with hook 709	\if@forced@series added 618
2020-01-20 ltfssdcl.dtx v3.0t	\if@forced@series: Switch
\set@mathdelimiter: fix for gh/251 690	\if@forced@series added 618
2020-01-20 ltoutenc.dtx v2.0n	2020-02-10 ltfssini.dtx v3.1h
General: fix for gh/251 490	\@defaultfamilyhook: Add
	\@defaultfamilyhook to
	\normalfont (gh/269) 710

<code>\reset@font</code> : Add	<code>\update@series@target@value</code> : Drop surplus “m” from <code>\reserved@</code>
<code>\@defaultfamilyhook</code> to	<code>(gh/291)</code> 701, 702
<code>\normalfont</code> (gh/269) 718	2020-02-27 ltdefns.dtx v1.5g
2020-02-10 lttextcomp.dtx v1.0c	<code>\@gobblethree</code> : Macro added 89
General: Use <code>\@tabacckludge</code> for	2020-02-27 ltfssaxes.dtx v1.0d
tabbing where necessary (gh/271) 761	<code>\series@maybe@drop@one@m</code> : Drop
2020-02-11 fontdef.dtx v3.0g	“m” in certain values from a fixed
General: Provide value for	list (gh/293) 621
<code>\@fontenc@load@list</code> (gh/273) . 725	<code>\set@target@series</code> : Drop “m” only
2020-02-11 ltfssini.dtx v3.1h	in a specific set of values (gh/293) 621
General: Provide default value for	2020-02-27 ltfssbas.dtx v3.2g
<code>\@fontenc@load@list</code> (gh/273) . 720	<code>\DeclareFontShape@</code> : Only “m” if the
2020-02-11 ltoutenc.dtx v2.0o	series value is a member of a fixed
General: Update	list and issue warning if doing it
<code>\@fontenc@load@list</code> with option	(gh/293) 534
list (gh/273) 520	2020-03-02 ltexpl.dtx v1.0a
2020-02-14 ltpictur.dtx v1.1n	General: Don’t load expl3 if already in
<code>\linethickness</code> : Suppress spaces	the format (gh/295) 73
following the declaration (gh/274) 927	2020-03-05 ltexpl.dtx v1.1
2020-02-18 ltfssini.dtx v3.1i	General: Load xparse.ltx if
<code>\bfseries</code> : Make the <code>\ifx</code> selection	<code>\NewDocumentCommand</code> is not
outside of <code>\fontseries</code> argument	defined by expl3.ltx 73
so that it is not done several times 704	2020-03-06 ltboxes.dtx v1.3c
<code>\mdseries</code> : Make the <code>\ifx</code> selection	<code>\clap</code> : Macro <code>\clap</code> added 896
outside of <code>\fontseries</code> argument	2020-03-07 ltluatex.dtx v1.1m
so that it is not done several times 705	<code>remove_from_callback</code> : Do not call
<code>\prepare@family@series@update</code> : No	callback.register for user-defined
series auto-update when forced	callbacks 70
(gh/277) 700	2020-03-07 ltmath.dtx v1.2e
Recognize current family if it is not	<code>\negthickspace</code> : Add <code>amsmath</code>
a “meta” family and auto-update	math/text spacing commands to
series using <code>\bfdefault</code> (gh/277) 700	the kernel (gh/303) 850
2020-02-18 ltmath.dtx v1.2d	2020-03-07 ltspace.dtx v1.3l
<code>\mathindent</code> : Make <code>\mathindent</code> a	General: Moved <code>\thinspace</code> ,
skip register to match <code>amsmath</code>	<code>\negthinspace</code> and <code>\,</code> to
(gh/252) 857	ltmath.dtx (gh/303) 430
equation: Separate formula and eqn	2020-03-19 fontdef.dtx v3.0h
number by at least a space in fleqn	General: Support legacy use of
option 858	<code>\bfdefault</code> and <code>\mddefault</code>
2020-02-20 ltclass.dtx v1.3j	(gh/306) 728
<code>\endfilecontents</code> : Fix missing quotes	2020-03-19 ltfssdcl.dtx v3.0u
around file name (gh/284) 1076	<code>\document@select@group</code> : fix for
2020-02-24 ltfssbas.dtx v3.2f	(gnats/3357) 669
<code>\DeclareFontShape@</code> : Drop surplus	2020-03-19 ltfssini.dtx v3.1k
“m” in series when defining	<code>\DeclareFontSeriesDefault</code> : Support
fontshape (gh/289) 534	legacy use of <code>\bfdefault</code> and
2020-02-25 ltfssini.dtx v3.1j	<code>\mddefault</code> (gh/306) 697
<code>\bf@default</code> : Drop surplus “m” from	<code>\maybe@update@bfseries@defaults</code> :
<code>\bfdef@ult</code> and <code>\mddef@ult</code>	Support legacy use of <code>\bfdefault</code>
(gh/291) 707	and <code>\mddefault</code> (gh/306) 704
<code>\prepare@family@series@update</code> :	<code>\mdseries</code> : Support legacy use of
Drop surplus “m” from	<code>\bfdefault</code> and <code>\mddefault</code>
<code>\target@series@value</code> (gh/291) 701	(gh/306) 705

2020-04-06 ltfssini.dtx v3.1m		<code>\@yargarraycr</code> : Support calc syntax (gh/152) 911
<code>\bf@default</code> : Hook added (gh/306) 707		
<code>\maybe@update@bfseries@defaults</code> : Hook added (gh/306) 704		2020-04-22 ltmiscen.dtx v1.1u
<code>\maybe@update@mdseries@defaults</code> : Hook added (gh/306) 705		<code>\@@sverb</code> : Drop spaces before <code>\verb</code> delimiter (gh/327) 840
2020-04-07 ltclass.dtx v1.3k		2020-04-22 ltoutenc.dtx v2.0p
<code>\IfFormatAtLeastTF</code> : Macro added; also in rollback (gh/168) 1047		General: y unicode value in tuenc.def 475
<code>\load@onefile@withoptions</code> : Use different method to ignore unprocessed options (gh/22) 1069		2020-04-29 lttextcomp.dtx v1.0d
<code>\ProcessOptions*</code> : Use different method to ignore unprocessed options (gh/22) 1058		General: Make all capital accents text commands for hyperref (gh/332) 761
<code>\RequirePackageWithOptions</code> : Use different method to ignore unprocessed options (gh/22) 1061		2020-05-02 ltfiles.dtx v1.2g
2020-04-09 ltfloat.dtx v1.2d		<code>\@include</code> : Support spaces in filenames by enclosing the names of .aux-files in quotes (gh/217) 461
<code>\@textsubscript</code> : Set non-zero baseline (gh/249) 991		<code>\includeonly</code> : Get rid of leading and trailing spaces from the filename (gh/217) 459
<code>\textsubscript</code> : Set non-zero baseline (gh/249) 990		Improved support for spaces in filenames (gh/217) 459
2020-04-13 ltfssdcl.dtx v3.0v		Pass the filename to <code>\@include</code> by value instead of by reference (gh/217) 459
<code>\process@table</code> : Small update for speed. 674		2020-05-05 ltxref.dtx v1.1n
2020-04-13 ltfssini.dtx v3.1n		<code>\refstepcounter</code> : record the counter name in <code>\@currentcounter</code> 801
<code>\init@series@setup</code> : Handling <code>\seriesdefault</code> changes (gh/315) 703		2020-05-06 ltspace.dtx v1.3n
<code>\seriesdefault@kernel</code> : Handling <code>\seriesdefault</code> changes (gh/315) 721		General: Made softhyphen active in TU engines 449
2020-04-21 ltmath.dtx v1.2f		2020-05-09 ltdefs.dtx v1.5j
<code>\@yeqncr</code> : Support calc syntax (gh/152) 855		<code>\@if@DeclareRobustCommand</code> : Added <code>\DeclareCommandCopy</code> (gh/239) 101
2020-04-21 ltmiscen.dtx v1.1t		<code>\DeclareCommandCopy</code> : Added <code>\DeclareCommandCopy</code> (gh/239) 98
<code>\@icentercr</code> : Support calc syntax (gh/152) 834		2020-05-11 ltdefs.dtx v1.5j
2020-04-21 ltpictur.dtx v1.1o		<code>\@dischyp</code> : Do not overwrite <code>\-</code> under LuaTeX 110
<code>\@istackcr</code> : Support calc syntax (gh/152) 927		2020-05-15 ltdefs.dtx v1.5g
2020-04-21 ltspace.dtx v1.3m		<code>\@typeout</code> : Allow <code>\par</code> in the argument (gh/335) 80
<code>\@hspace</code> : Support calc syntax (gh/152) 448		2020-05-19 ltfssaxes.dtx v1.0e
<code>\@newline</code> : Support calc syntax (gh/152) 435		<code>\series@maybe@drop@one@m</code> : Need to use <code>\edef</code> (gh/336) 622
<code>\@vspace@calcify</code> : Support calc syntax (gh/152) 435		2020-05-19 ltfssini.dtx v3.2a
<code>\@vspacer</code> : Support calc syntax (gh/152) 444		<code>\IfFontSeriesContextTF</code> : Macros added (gh/335) 712
<code>\@advspace</code> : Support calc syntax (gh/152) 441		2020-05-31 ltmiscen.dtx v1.1u
2020-04-21 lttab.dtx v1.1r		<code>\centering</code> : Added <code>\finalhyphendemerits</code> setting (gh/247) 834
<code>\@itabcr</code> : Support calc syntax (gh/152) 903		<code>\raggedleft</code> : Added <code>\finalhyphendemerits</code> setting (gh/247) 835

<ul style="list-style-type: none"> <code>\raggedright</code>: Added <ul style="list-style-type: none"> <code>\finalhyphendemerits</code> setting (gh/247) 835 	<ul style="list-style-type: none"> 2020-08-02 ltluatex.dtx v1.1q <ul style="list-style-type: none"> <code>\newattribute</code>: Move reset to 0 inside conditional 49 <code>\newluabytecode</code>: Move reset to 0 inside conditional 52 <code>\newluachunkname</code>: Move reset to 0 inside conditional 52 <code>\newluafunction</code>: Move reset to 0 inside conditional 51 <code>\newwhatsit</code>: Move reset to 0 inside conditional 51
<ul style="list-style-type: none"> 2020-06-04 ltexpl.dtx v1.2c <ul style="list-style-type: none"> General: Define a local version of some \LaTeX basic macros to support package loading 73 	<ul style="list-style-type: none"> 2020-08-08 ltclass.dtx v1.3m <ul style="list-style-type: none"> <code>\endfilecontents</code>: define <code>\qcurr@file</code> directly as the quotes have already been removed (gh/220) 1075
<ul style="list-style-type: none"> 2020-06-04 ltfinal.dtx v2.2a <ul style="list-style-type: none"> General: Load ltexpl in ltdefns . . . 1277 	<ul style="list-style-type: none"> 2020-08-10 ltluatex.dtx v1.1r <ul style="list-style-type: none"> General: Load ltluatex Lua module during format building 52
<ul style="list-style-type: none"> 2020-06-05 ltclass.dtx v1.3l <ul style="list-style-type: none"> <code>\currnamestack</code>: Added <ul style="list-style-type: none"> <code>\@expl@pop@filename@@</code> 1045 Added <code>\@expl@push@filename@@</code> and <code>\@expl@push@filename@aux@@</code> . . 1044 	<ul style="list-style-type: none"> 2020-08-15 ltpictur.dtx v1.2a <ul style="list-style-type: none"> <code>\defaultunitsset</code>: Macro added . . 924
<ul style="list-style-type: none"> 2020-06-05 ltfiles.dtx v1.2h <ul style="list-style-type: none"> <code>\document</code>: Added hook to load l3backend code 453 	<ul style="list-style-type: none"> 2020-08-19 ltdefns.dtx v1.5k <ul style="list-style-type: none"> <code>\@carcube</code>: Made <code>\long</code> for <code>\NewCommandCopy</code> 83 <code>\robust@command@act</code>: Made <code>\robust@command@act</code> (was <code>\declare@command@copy</code>) more generic 96 <code>\ShowCommand</code>: Added <code>\ShowCommand</code> (gh/373) 100
<ul style="list-style-type: none"> 2020-06-10 ltluatex.dtx v1.1n <ul style="list-style-type: none"> General: Define <ul style="list-style-type: none"> <code>\@gobble/\@firstofone</code> even for \LaTeX to allow early loading. . . . 47 	<ul style="list-style-type: none"> 2020-08-19 ltexpl.dtx v1.2e <ul style="list-style-type: none"> General: Add <ul style="list-style-type: none"> <code>\@expl@cs@<thing>@spec@@N</code> for <code>\ShowCommand</code> (gh/373) 77 Add <code>\@expl@cs@to@str@@N</code> and <code>\@expl@str@if@eq@@nnTF</code> for <code>\NewCommandCopy</code> (gh/239) 76
<ul style="list-style-type: none"> 2020-07-04 ltoutenc.dtx v2.0q <ul style="list-style-type: none"> General: Implement <code>\remove@tlig</code> in Lua\TeX without font reloading . . 508 	<ul style="list-style-type: none"> 2020-08-20 ltplain.dtx v2.3d <ul style="list-style-type: none"> <code>\alloc@</code>: Define <code>\alloc@</code> in terms of <code>\e@alloc</code> 21
<ul style="list-style-type: none"> 2020-07-08 ltexpl.dtx v1.2d <ul style="list-style-type: none"> General: Add a last-minute hook for <code>expl3</code> 74 	<ul style="list-style-type: none"> 2020-08-21 ltclass.dtx v1.3o <ul style="list-style-type: none"> General: Integration of new hook management interface 1038
<ul style="list-style-type: none"> 2020-07-08 ltfinal.dtx v2.2b <ul style="list-style-type: none"> General: Add a last-minute hook for <code>expl3</code> 1266 	<ul style="list-style-type: none"> 2020-08-21 ltdefns.dtx v1.5l <ul style="list-style-type: none"> General: Integration of new hook management interface 80
<ul style="list-style-type: none"> 2020-07-27 ltmath.dtx v1.2g <ul style="list-style-type: none"> <code>\cases</code>: Don't make the command <code>\long</code> (gh/354) 848 <code>\matrix</code>: Don't make the command <code>\long</code> (gh/354) 848 <code>\pmatrix</code>: Don't make the command <code>\long</code> (gh/354) 848 	<ul style="list-style-type: none"> 2020-08-21 ltdefns.dtx v1.5m <ul style="list-style-type: none"> <code>\MakeRobust</code>: Make <code>\MakeRobust</code> produce the same command structure as <code>\DeclareRobustCommand</code> 93
<ul style="list-style-type: none"> 2020-07-27 ltexpl.dtx v1.2d <ul style="list-style-type: none"> General: Add a last-minute hook for <code>expl3</code> 1266 	<ul style="list-style-type: none"> 2020-08-21 ltexpl.dtx v1.2d <ul style="list-style-type: none"> General: Dropped unused command . . 73
<ul style="list-style-type: none"> 2020-07-27 ltoutenc.dtx v2.0r <ul style="list-style-type: none"> <code>\@use@text@encoding</code>: Don't make the command <code>\long</code> (gh/354) . . . 485 	
<ul style="list-style-type: none"> 2020-07-27 ltpage.dtx v1.0m <ul style="list-style-type: none"> <code>\markright</code>: Don't make the command <code>\long</code> (gh/354) 1034 	
<ul style="list-style-type: none"> 2020-07-27 ltpictur.dtx v1.1p <ul style="list-style-type: none"> <code>\linethickness</code>: Don't make the command <code>\long</code> (gh/354) 927 	
<ul style="list-style-type: none"> 2020-07-27 ltsect.dtx v1.1e <ul style="list-style-type: none"> <code>\author</code>: Don't make the command <code>\long</code> (gh/354) 958 <code>\date</code>: Don't make the command <code>\long</code> (gh/354) 958 	
<ul style="list-style-type: none"> 2020-08-01 ltluatex.dtx v1.1p <ul style="list-style-type: none"> General: <code>new_ graf</code> is <code>exclusive</code> . . . 63 	

2020-08-21 ltfiles.dtx v1.2i	2020-09-30 ltffssini.dtx v3.2d
General: Integration of new hook management interface 451	\maybe@update@bfseries@defaults: \bfdefault@previous not \bfseries@previous (gh/395) .. 704
2020-08-21 ltfinal.dtx v2.2i	\mdseries:\mddefault@previous not \mdseries@previous (gh/395) .. 705
General: Integration of new hook management interface 1260	2020-10-01 ltclass.dtx v1.3r
2020-08-21 ltffssaxes.dtx v1.0g	\@pr@videpackage: Allow for package substitution 1052
General: Integration of new hook management interface 631	2020-10-01 ltsect.dtx v1.1e
2020-08-21 ltffssini.dtx v3.2b	\addcontentsline: add a fourth argument for better hyperref compatibility 967
\bf@def@ult: Integration of new hook management interface 707	2020-10-04 ltfiles.dtx v1.2j
mdseries/defaults: Integration of new hook management interface 710	\@include: Quotes around the aux file name removed, they are not needed and upset BibTeX (gh/400) 461
\maybe@update@bfseries@defaults: Integration of new hook management interface 704	2020-10-04 lthooks.dtx v1.0d
\maybe@update@mdseries@defaults: Integration of new hook management interface 705	General: Definition \AddToHookNext was supposed to be for \AddToHook vice versa (gh/401) 306
\reset@font: Integration of new hook management interface 718	2020-10-08 ltclass.dtx v1.3s
\rmfamily: Integration of new hook management interface 709	\@currnamestack: Added missing 2020/02/02 \IncludeInRelease 1044
\ttfamily: Integration of new hook management interface 709	2020-10-11 ltclass.dtx v1.3t
2020-08-21 ltmiscen.dtx v1.1v	\load@onefilewithoptions: Restore \@currpkg@reqd after finished loading a package file (gh/408). 1067
General: Integration of new hook management interface 819	2020-10-18 ltclass.dtx v1.3t
2020-08-21 ltoutput.dtx v1.4f	\PassOptionsToClass: Drop path from \input@path (gh/414). .. 1054
\@begindvibox: Integration of new hook management interface . . . 1164	2020-10-23 ltmiscen.dtx v1.1w
2020-08-23 ltxref.dtx v1.1o	\enddocument: Make enddocument/afteraux one-time 821
\refstepcounter: add default definition of \@currentcounter . 801	2020-10-26 ltmiscen.dtx v1.1x
2020-09-06 ltclass.dtx v1.3q	\@kernel@before@enddocument: \enddocument should always start out in vmode (gh/385) 824
\load@onefilewithoptions: Save \@currpkg@reqd so that we don't lose track of package substitutions. 1066	2020-11-09 ltclass.dtx v1.3u
2020-09-06 ltdefs.dtx v1.5n	\pkgcls@rollbackdate@error: Change help text because the package may have existed then — there is just no rollback data (gh/423). 1088
\char@if@alph: Macro added 109	2020-11-09 ltmath.dtx v1.2h
2020-09-06 ltexpl.dtx v1.2f	\@kernel@generate@nnegmedspace and \negthickspace: \negmedspace and \negthickspace have been only in amsmath, so we need to undefine for rollback (gh/423) 851
General: Add \@expl@str@map@function@CNN and \@expl@char@generate@nnegmedspace and for \string@makeletter (gh/386) 77	2020-11-20 ltclass.dtx v1.3u
2020-09-09 ltshipout.dtx v1.0b	\@currpath: Macro added 1043
__shipout_picture_overlay:n: Prevent overfull box warnings (gh/387) 1144	\@kernel@currpathstack: Macro added 1046
2020-09-26 ltfinal.dtx v2.2j	
General: Load first aid file if existing 1278	

<code>\load@onefile@withoptions</code> : Copy option list to the requested package.	1068	<code>\fontseries</code> : Distangle series and shape update (gh/444)	618
<code>\PassOptionsToClass</code> : Copy option list to the requested package. . .	1054	<code>\fontshape</code> : Distangle series and shape update (gh/444)	629
<code>\ProvidesPackage</code> : Use string comparison instead of <code>\ifx</code> . . .	1051	<code>\fontshapeforce</code> : Distangle series and shape update (gh/444)	629
2020-11-20 <code>ltxcmd.dtx</code> v1.0a		2020-12-04 <code>ltxssini.dtx</code> v3.2f	
General: Initial version derived from <code>xparse.dtx</code>	113	General: Adjust start values for series and shape (gh/444)	720
2020-11-20 <code>ltxfilehook.dtx</code> v1.0d		2020-12-10 <code>ltxbibl.dtx</code> v1.1s	
<code>\unqu@tefilef@und</code> : Move loading to <code>\@input@file@exists@with@hooks</code> and expand <code>\@filef@und</code> to avoid getting the wrong file name in the case of a substitution.	1109	<code>\nocite</code> : Delay any <code>\nocite</code> in the preamble instead of raising an error	1001
2020-11-23 <code>ltxshipout.dtx</code> v1.0d		2020-12-10 <code>ltxssbas.dtx</code> v3.2h	
<code>__shipout_execute_cont::</code> Check for both kernel and user hook (gh/431)	1134	<code>\usefont</code> : Drop “m” if the series value is a member of a fixed list and issue warning if doing it (gh/453)	544
<code>__shipout_execute_main_cont:Nmn</code> : Check for both kernel and user hook (gh/431)	1136	2020-12-14 <code>ltxclass.dtx</code> v1.3v	
2020-11-24 <code>ltxexpl.dtx</code> v1.2g		<code>\@currnamestack</code> : Removed <code>\@expl@@@hook@curr@name@push@@@</code>	1044
General: Support for roll forward (gh/434)	75–77	2020-12-18 <code>ltxexpl.dtx</code> v1.2h	
2020-11-24 <code>ltxfilehook.dtx</code> v1.0d		<code>\@kernel@after@enddocument@afterlastpage</code> : Define kernel <code>\enddocument</code> hooks early	73
General: Support for roll forward (gh/434)	1107	2020-12-22 <code>ltxssaxes.dtx</code> v1.0h	
2020-11-24 <code>ltxhooks.dtx</code> v1.0f		<code>\delayed@merge@font@series</code> : Distangle series and shape update (gh/444)	620, 621
<code>__hook_end_document_label_check::</code> Support for roll forward (gh/434)	242	<code>\delayed@merge@font@shape</code> : Distangle series and shape update (gh/444)	630
2020-11-24 <code>ltxshipout.dtx</code> v1.0d		2020-12-22 <code>ltxsstrc.dtx</code> v3.0n	
General: Support for roll forward (gh/434)	1148	<code>\selectfont</code> : Execute delayed series and shape updates (gh/444)	637
<code>\AtBeginDvi</code> : Support for roll forward (gh/434)	1147	2021-01-07 <code>ltxfilehook.dtx</code> v1.0e	
2020-11-25 <code>ltxdefns.dtx</code> v1.5o		General: Added rollback for this case to avoid spurious errors (part of gh/463)	1121
<code>\@carcube</code> : Added missing <code>latexrelease</code> entry	83	<code>\unqu@tefilef@und</code> : Restore <code>\CurrentFile(Path)(Used)</code> after the input (gh/464)	1110
2020-12-02 <code>ltxluatex.dtx</code> v1.1s		2021-01-07 <code>ltxhooks.dtx</code> v1.0h	
General: Fix return value of list callbacks	65	<code>__hook_strip_double_slash:w</code> : Assume hook name has at least three nonempty parts (gh/464)	254
2020-12-03 <code>ltxsstrc.dtx</code> v3.0m		<code>__hook_tl_set:cn</code> : Manually define some <code>l3tl</code> commands to work around <code>expl3</code> changes	229
<code>\selectfont</code> : Install a hook in <code>\selectfont</code> (gh/444)	638	2021-01-08 <code>ltxshipout.dtx</code> v1.0f	
2020-12-04 <code>ltxfilehook.dtx</code> v1.0d		<code>__shipout_execute_cont::</code> Added another kernel hook for more flexibility (cf.	
<code>\undeclare@file@substitution</code> : Don’t drop file substitution commands on rollback	1112		
2020-12-04 <code>ltxssaxes.dtx</code> v1.0h			
General: Reorganized the rollback data	563		

https://github.com/pgf-tikz/pgf/issues/4960	2021-01-10 ltfloat.dtx v1.2e	
.....	1134	<code>\@footnotetext</code> : Explicitly run <code>\par</code> at the end of footnote text in preparation for paragraph hooks
2021-01-10 ltshipout.dtx v1.0g		991
<code>\@kernel@after@shipout@background</code> : Internal hook		2021-02-15 ltfsdcl.dtx v3.0w
<code>\@kernel@after@shipout@background</code> added	1138	<code>\document@select@group</code> : fix for (gh/501)
<code>\RawShipout</code> : Macro added	1137	2021-02-16 ltfloat.dtx v1.2f
2021-01-19 ltshipout.dtx v1.0h		<code>\footref</code> : <code>\footref</code> added
<code>__shipout_run_firstpage_hook::</code> : Handling of firstpage hook altered	1138	2021-02-17 ltoutenc.dtx v2.0t
2021-01-21 ltclass.dtx v1.3w		General: Adjust values for <code>\textasteriskcentered</code> To match TS1 definition (gh/502)
<code>\@kernel@currpathstack</code> : Add empty entry for latexrelease	1046	Special definition for <code>\textasteriskcentered</code> when missing in TS1 (gh/502)
2021-01-21 ltexpl.dtx v1.3a		504
General: Move xparse rollback code to <code>ltxcmd.dtx</code>	76	2021-02-18 ltclass.dtx v1.3x
2021-01-21 ltfinal.dtx v2.2l		<code>\@fileswithoptions</code> : save raw class option list (gh/85)
General: Load glyphtounicode.tex for pdfTeX	1268	<code>\@remove@eq@value</code> : macro added (gh/85)
2021-01-22 ltshipout.dtx v1.0i		<code>\@use@option</code> : value from unused option list (gh/85)
<code>__shipout_finalize_box::</code> : Add <code>pre_shipout_filter</code> Lua callback	1133	<code>\OptionNotUsed</code> : value from unused option list (gh/85)
2021-01-24 ltexpl.dtx v1.3a		<code>\PassOptionsToClass</code> : save raw option lists (gh/85)
General: Define <code>expl3</code> hooks conditionally	73	2021-02-19 ltoutenc.dtx v2.0u
2021-01-31 ltfilehook.dtx v1.0f		General: Add <code>\textnonbreakinghyphen</code> , <code>\textfiguredash</code> and <code>\texthorizontalbar</code> (gh/404)
<code>\@curr@file@reqd</code> : set <code>\protect</code> to <code>\string</code> gh/481	1113	2021-02-25 ltfinal.dtx v2.2m
2021-02-03 ltfloat.dtx v1.2e		General: Improve speed of ToUnicode everyjob loading code
<code>\@savemarbox</code> : Explicitly end with <code>\par</code> (gh/489)	985	2021-03-03 ltclass.dtx v1.3y
2021-02-04 ltboxes.dtx v1.4b		<code>\endfilecontents</code> : Fix overwrite check for files with UTF-8 (gh/415)
<code>\color@endbox</code> : Always add the color groups (gh/488)	882	2021-03-05 ltclass.dtx v1.3z
2021-02-08 ltfilehook.dtx v1.0g		<code>\ProcessOptions*</code> : modify so braces to not give errors (gh/513)
<code>\unqu@tefile@und</code> : Undo the internal for robust <code>\InputIfFileExists</code> in rollback (gh/494)	1111	2021-03-12 ltfiles.dtx v1.2k
2021-02-08 ltmiscen.dtx v1.1y		<code>\IfFileExists@</code> : Allow unbalanced conditionals (gh/530)
<code>\begin</code> : Undo the internals for robust <code>\begin</code> and <code>\end</code> in rollback (gh/494)	829	2021-03-18 ltxcmd.dtx v1.0b
<code>\end\verbvisibleospace</code> : Undo the internals for robust <code>\begin</code> and <code>\end</code> in rollback (gh/494)	832	General: Use <code>\NewModuleRelease</code> .
2021-02-10 ltboxes.dtx v1.4b		113
<code>\@mpfootnotetext</code> : Explicitly run <code>\par</code> in support for paragraph tagging	892	2021-03-18 ltfilehook.dtx v1.0h
		<code>__filehook_file_pop_assign:n</code> : Define <code>\g__filehook_input_file_seq</code> to avoid losing data when rolling back.
		1107

2021-03-18 ltfsaxes.dtx v1.0i	2021-04-20 ltexpl.dtx v1.3c
General: Fix rollforward definition.	\@kernel@after@enddocument@afterlastpage:
2021-03-18 ltfsini.dtx v3.2g	Don't empty kernel hooks on
General: Add legacy hook definitions	rollback
for rollback.	2021-04-20 ltfilehook.dtx v1.0i
2021-03-18 lthooks.dtx v1.0i	\@curr@file@reqd: Make expand to
_hook_end_document_label_check::	a string (tracks change in
Only add top-level if not already	l3kernel)
there.	2021-04-26 ltfsbas.dtx v3.2i
Remove the (empty) "top-level"	\usefont: Unconditionally switch to
from \@currnamestack.	the requested font face (gh/444)
General: Use \NewModuleRelease.	2021-04-26 ltfsini.dtx v3.2h
2021-03-18 ltvers.dtx v1.1f	\reset@font: Unconditionally switch
\@check@IncludeInRelease: Add	to the requested font face
support for usage in	(gh/444)
\NewModuleRelease	2021-04-26 ltfsstrc.dtx v3.0o
\new@moduledate: Added	\selectfont: Unset the forced series
\NewModuleRelease.	boolean when reaching
2021-03-19 lttextcomp.dtx v1.0e	\selectfont (gh/444)
General: Use \NewModuleRelease	2021-04-29 lthooks.dtx v1.0m
2021-03-26 ltplain.dtx v2.3e	\ActivateGenericHook: Add
\@unused: Allocate \@inputcheck and	\ProvideHook etc.
\@unused early so that they are	2021-04-29 ltoutenc.dtx v2.0v
before expl3 allocates more	General: Add composites for
streams (gh/538)	\ae/\AE/\ae/\AE (gh/552)
2021-03-27 ltclass.dtx v1.4a	2021-05-18 ltclass.dtx v1.4b
\@currnamestack: Do not completely	\@raw@classoptionslist: Initialise to
roll back if expl3 is loaded.	\relax to match
2021-04-16 ltvers.dtx v1.1g	\@classoptionslist
\new@moduledate: \NewModuleRelease	2021-05-24 ltcmd.dtx v1.0e
with the same arguments as	General: Use \msg_... instead of
\IncludeInRelease.	_kernel_msg_...
2021-04-17 ltfiles.dtx v1.2m	2021-05-24 ltcmdhooks.dtx v1.0b
\@kernel@after@begindocument:	General: Use \msg_... instead of
Move	_kernel_msg_...
\@kernel@before@begindocument	2021-05-24 ltfilehook.dtx v1.0k
and	General: Use \msg_... instead of
\@kernel@after@begindocument	_kernel_msg_...
init earlier so that other modules	2021-05-24 lthooks.dtx v1.0n
can write to the hooks	General: Use \msg_... instead of
2021-04-18 ltluatex.dtx v1.1t	_kernel_msg_...
General: input_level_string added	2021-05-24 ltpara.dtx v1.0g
2021-04-18 ltplain.dtx v2.3f	General: Use \msg_... instead of
\loggingall: 3	_kernel_msg_...
Drop pre- ϵ -TeX support	2021-05-26 ltdefns.dtx v1.5p
\tracingnone: 3	\MakeRobust: Normalize error message
Drop pre- ϵ -TeX support	in \MakeRobust
2021-04-19 ltcmd.dtx v1.0d	2021-06-03 ltclass.dtx v1.4c
_cmd_cmd_type_cases:NnnnnF:	\@kernel@currpathstack: Take care
Renamed	of \@kernel@currpathstack when
_cmd_cmd_if_xparse:NTF to	rolling back/forward.
_kernel_cmd_if_xparse:NTF for	2021-06-04 ltcmd.dtx v1.0f
cross-module usage	General: Normalize various error
	messages

2021-06-05 ltmiscen.dtx v1.1z		2021-07-22 lthooks.dtx v1.0o	
\@sverb: Normalize error message .	840	_hook_gremove_code:nn: Do not	
2021-06-06 ltclass.dtx v1.4c		queue removals (gh/625)	256
\loadwithoptions: handle raw		_hook_prop_gput_labeled_do:Nnnn:	
options for gh/580	1060	Do not queue removals (gh/625)	244
\load@onefile@withoptions: Copy		2021-07-23 ltclass.dtx v1.4e	
raw options for gh/580	1068	\load@onefile@withoptions: Make	
\PassOptionsToClass: apply		class/name/after a one-time	
\expandafter to raw options for		hook	1069
gh/580	1054	Make class/name/before a one-time	
2021-06-09 ltclass.dtx v1.4b		hook	1068
\endfilecontents: Use		Make package/name/after a	
\@latex@note@no@line to display		one-time hook	1069
the information	1075	Make package/name/before a	
2021-06-09 lterror.dtx v1.2r		one-time hook	1068
\@latex@note@no@line: Macros		2021-07-23 ltfiles.dtx v1.2n	
added	402	\@include: Make include/name/after	
2021-06-09 ltfssbas.dtx v3.2j		a one-time hook	461
\DeclareFontShape@: Improve		Make include/name/before a	
information message	534	one-time hook	461
2021-07-08 ltcounts.dtx v1.1m		Make include/name/end a one-time	
\counterwithin: New implementation		hook	461
for \counterwithout and		2021-07-27 lthooks.dtx v1.0o	
\counterwithin	526	\ClearHookNext: Macro added	303
2021-07-11 lterror.dtx v1.2s		_hook_gclear_next_code:n: Macro	
\PackageNoteNoLine: Provide		made public	288
\ClassNote and \PackageNote .	401	2021-07-28 ltsect.dtx v1.1f	
2021-07-12 ltclass.dtx v1.4d		\contentsline: Pick up four	
\@fileswithoptions: add		arguments (gh/633)	969
\unexpanded	1063	2021-07-30 ltcmd.dtx v1.0d	
2021-07-16 ltplain.dtx v2.3g		_cmd_cmd_type_cases:NnnnnnF:	
General: Use 2 as default value for		Added	
\tracinglostchars	24	_cmd_cmd_type_cases:NnnnnF	
2021-07-19 ltclass.dtx v1.4e		for \NewCommandCopy and	
\@classoptionslist: Drop		\ShowCommand support	186
\@onlypreamble	1043	2021-07-31 ltoutput.dtx v1.4e	
\@ifclasslater: Drop		\fl@tracemessage: Enable display	
\@onlypreamble	1047	when doing \tracefloatvals .	1226
\@ifclassloaded: Drop		2021-07-31 ltoutput.dtx v1.4g	
\@onlypreamble	1047	\ShowFloat: Macro added	1224
\@ifclasswith: Drop		2021-08-02 lthooks.dtx v1.0o	
\@onlypreamble	1049	\ActivateGenericHook: Change	
\@ifl@ter: Drop \@onlypreamble	1048	name	302
\@pkgextension: Drop		\DisableGenericHook: Change name	302
\@onlypreamble	1043	2021-08-07 ltcmd.dtx v1.0g	
\@optionlist: Drop \@onlypreamble	1047	_cmd_add_grabber:N: Replicate	
\@unusedoptionlist: Drop		argument processors for all	
\@onlypreamble	1043	embellishments (gh/639)	140
\IfFormatAtLeastTF: Drop		_cmd_add_type_E:w: Replicate	
\@onlypreamble	1047	argument processors for all	
2021-07-20 ltcmdhooks.dtx v1.0c		embellishments (gh/639)	138
_hook_patch_DeclareRobustCommand:Nnn:		2021-08-08 ltfinal.dtx v2.2p	
Use \robust@command@chk@safe		\IfPDFManagementActiveTF: Default	
before \@if@newcommand.	319	definition added (gh/640)	1278

2021-08-10 ltvers.dtx v1.1h	2021-08-30 ltcmd.dtx v1.0h
\@check@IncludeInRelease: Add	General: Added support for
error to aid debugging 39	\NewCommandCopy 145
2021-08-11 ltluatex.dtx v1.1u	Added support for \ShowCommand 151
General: Define missing local function 54	2021-09-03 ltoutput.dtx v1.4h
2021-08-20 lterror.dtx v1.2t	\ShowFloat: Renamed, original name
\@badend: Improve \@badend error	never distributed 1224
message (gh/587) 405	2021-09-06 ltfinal.dtx v2.2q
2021-08-20 lthooks.dtx v1.0p	\@uclclist: Correctly upper and
General: Added deprecation warnings	lowercase \ij and \IJ (gh/658) 1276
for old generic hook commands	2021-09-06 lthooks.dtx v1.0r
(gh/638) 305	_hook_if_execute_immediately:n:
Documentation updates for generic	Macro added (gh/606) 294
hook commands (gh/638) 200	_hook_prop_gput_labeled_do:Nnnn:
Renames of generic hook commands	Use dedicated conditional
(gh/638) 238	(gh/606) 244
Section on generic hooks added	_hook_use_once_clear:n: Clean up
(gh/638) 217	after \UseOneTimeHook (gh/606) 293
2021-08-25 ltclass.dtx v1.4f	_hook_use_once:nnw: Clean up after
General: Standardise generic hook	\UseOneTimeHook (gh/606) 292
names (gh/648) 1038	2021-09-10 ltfssini.dtx v3.2i
\load@onefile@withoptions: Declare	\bfseries: Do delayed changes to
non-generic package and class	\bfdefault in a separate macro
hooks 1069	for better reuse (gh/663) 704
2021-08-25 ltcmdhooks.dtx v1.0d	\DeclareFontSeriesDefault: Do
_hook_try_put_cmd_hook:w:	delayed changes to \bfdefault or
Simplify generic hook detection . 315	\mddefault first (gh/663) 697
2021-08-25 ltfilehook.dtx v1.0l	\maybe@update@bfseries@defaults:
\unqu@tefilef@und: Declare	Do delayed changes to \bfdefault
non-generic file hooks 1110	in a separate macro for better
2021-08-25 ltfiles.dtx v1.2o	reuse (gh/663) 704
\@include: Declare non-generic	\maybe@update@mdseries@defaults:
include hooks 462	Do delayed changes to \mddefault
Standardise generic hook names	in a separate macro for better
(gh/648) 461	reuse (gh/663) 705
2021-08-25 lthooks.dtx v1.0p	\mdseries: Do delayed changes to
_hook_try_declaring_generic_next_hook:nn:	\mddefault in a separate macro
Standardize generic hook names	for better reuse (gh/663) 705
(gh/648) 248	2021-09-12 ltfntcmd.dtx v3.5a
2021-08-27 ltcmd.dtx v1.0h	\check@nocorr@: use \unexpanded to
\NewDocumentEnvironment: Check for	make # safe 751
end-of-environment command . . 194	2021-09-12 ltoutenc.dtx v2.0w
2021-08-27 ltfilehook.dtx v1.0l	General: Move zero skip between i and
_filehook_file_pop_assign:nnnn:	j for hyphenation (gh/658) 493
Internal message name changes 1107	2021-09-18 ltpara.dtx v1.0i
_filehook_file_subst_cycle_error:cN:	\para_end:: Use skip rather than kern
Use \msg_... not	as guard. 424
_kernel_msg_... 1118	2021-09-26 ltfssdcl.dtx v3.0x
2021-08-27 lthooks.dtx v1.0q	\c@localmathalphabets: Counter
General: Internal message name	added for (gh/676) 668
changes 298	\document@select@group: Test if we
2021-08-27 ltpara.dtx v1.0i	should freeze the version (gh/676) 668
General: Internal message name	\freeze@math@version: Macro added
changes 426	for (gh/676) 671

2021-09-28 ltcmdhooks.dtx v1.0e	vpack_ quality is exclusive	63
<code>__hook_make_prefixes:w</code> : Make patching of commands a global operation (gh/674)		324
<code>__hook_patch_retokenize:Nnnn</code> : Make patching of commands a global operation (gh/674)		332
2021-09-28 lthooks.dtx v1.0s		
<code>__hook_if_usable_use:n</code> : Correct usage of older		
<code>__hook_if_file_hook:wTF</code> (gh/675)		292
<code>__hook_try_declaring_generic_hook_split:nNnn</code> : Correct usage of older		
<code>__hook_if_file_hook:wTF</code> (gh/675)		249
2021-10-14 ltboxes.dtx v1.4c		
<code>\@mpfootnotetext</code> : Explicitly set		
<code>\@currentcounter</code> (gh/687) . . .		892
2021-10-14 ltfiles.dtx v1.2p		
<code>\@include</code> : Warn about use in preamble		461
2021-10-14 ltfloat.dtx v1.2g		
<code>\@footnotetext</code> : Explicitly set		
<code>\@currentcounter</code> (gh/687) . . .		991
2021-10-14 ltmath.dtx v1.2j		
<code>\@eqnset</code> : Explicitly set		
<code>\@currentcounter</code> (gh/687) . . .		854
<code>eqnarray</code> : Explicitly set		
<code>\@currentcounter</code> (gh/687) . . .		858
2021-10-15 ltfssdcl.dtx v3.0y		
<code>\DeclareMathVersion</code> : Initialize variable for freezing math version (gh/676)		675
2021-10-15 ltluatex.dtx v1.1v		
General: provide <code>_charproc_data</code> added		63
2021-10-16 ltoutenc.dtx v2.0x		
<code>\add@accent</code> : Dont set		
<code>\spacefactor in math mode</code> gh/643		482
2021-10-19 ltpara.dtx v1.0k		
General: Remove content from		
<code>\tex_everypar:D</code> on rollback . . .		426
2021-10-20 ltcmdhooks.dtx v1.0f		
<code>__hook_make_prefixes:w</code> : Correct patching by expansion+redefinition when the macro contains a parameter token (gh/697)		320
2021-11-17 ltluatex.dtx v1.1w		
General: <code>hpack_ quality is exclusive</code>		63
Never pass on <code>true</code> return values for list callbacks		65
2021-11-30 ltclass.dtx v1.5a		
<code>\@onefilewithoptions@clashchk</code> : Separated out from		
<code>\@onefilewithoptions</code>		1067
2021-11-30 ltexpl.dtx v1.3d		
<code>\skipeval</code> : Moved over from xfp package (gh/711)		77
2021-12-02 ltcmd.dtx v1.0i		
<code>__cmd_run_code::</code> : Correct defaults for optional arguments in		
end-of-environment code (gh/712)		123
<code>__cmd_start_aux:ccnnnn</code> : Correct defaults for optional arguments in		
end-of-environment code (gh/712)		123
2021-12-07 ltexpl.dtx v1.3d		
<code>\skipeval</code> : Added <code>\dimeval</code> and <code>\skipeval</code> (gh/711)		78
2021-12-11 ltdirchk.dtx v1.3a		
General: Add comment lines into <code>latex.ltx</code> to indicate temp definitions that are later overwritten (gh/725)		2
2021-12-12 ltoutenc.dtx v2.0y		
General: <code>\DeclareUnicodeAccent</code> now makes the encoding argument implicit (gh/253)		509
Added <code>\DeclareUnicodeCommand</code> and <code>\DeclareUnicodeSymbol</code> (gh/253)		510
2021-12-27 ltluatex.dtx v1.1x		
<code>\newprotectedluaCmd</code> : Macros added		51
2021-12-28 ltexpl.dtx v1.3d		
<code>\ExpandArgs</code> : Added document level names for <code>\exp_ args:Nc</code> and the like (gh/735)		78
2021-13-31 ltcmd.dtx v1.0j		
<code>__cmd_show:x</code> : Make <code>\ShowCommand</code> stop for interaction		152, 153
2022-01-06 ltexpl.dtx v1.3e		
<code>\ExpandArgs</code> : Adjust document level names for <code>\exp_ args:Nc</code> and the like (gh/735)		78
2022-01-06 ltshipout.dtx v1.0l		
<code>\AtBeginShipoutNext</code> : Correctly simulate <code>\AtBeginShipout</code> and <code>\AtBeginShipoutNext</code> without extending the syntax		1150
<code>\AtEndDvi</code> : Correctly simulate <code>\AtEndDvi</code> without extending the syntax		1149
2022-01-15 ltkeys.dtx v1.0b		
<code>__keys_options_aux:n</code> : Clear option list in end-of-package hook . . .		1094

2022-01-25 ltplain.dtx v2.3h		2022-03-10 ltfilehook.dtx v1.0m	
\obeyspaces: Provide redirection to support special use cases (gh/367)	28	\@curr@file@reqd: Add	
2022-02-05 ltkeys.dtx v1.0b		\set@curr@file@nosearch for graphicx	1113
.usage: Create properties in ltkeys	1092	2022-03-18 ltclass.dtx v1.5b	
2022-02-07 ltkeys.dtx v1.0c		\load@onefilewithoptions: Switch to \ProcessKeyOptions	1066
.usage: Correct .code property	1092	2022-03-18 ltcmd.dtx v1.0l	
2022-02-15 ltkeys.dtx v1.0c		__cmd_cmd_type_cases:NnnnnF: Fix __cmd_cmd_type_cases:NnnnnF prematurely expanding macros (gh/795)	186
\DeclareKeys: Expand module argument	1097	2022-03-18 ltkeys.dtx v1.0f	
\DeclareUnknownKeyHandler: Added \DeclareUnknownKeysHandler	1097	__keys_options:n: Simplify to always cover global options	1094
\ProcessKeyOptions: Expand module argument	1097	__keys_options_global:n: Simplify to always cover global options	1094
\SetKeys: Expand module argument	1099	\ProcessKeyOptions: Remove \ProcessKeyPackageOptions	1097
2022-02-16 ltkeys.dtx v1.0d		2022-04-01 ltfiles.dtx v1.2q	
\DeclareKeys: Allow for active characters in module argument	1097	\@include: Process some hooks is an include file is bypassed	462
\DeclareUnknownKeyHandler: Allow for active characters in module argument	1097	2022-04-01 lthooks.dtx v1.0t	
Rename		\c__hook_generic_include/./end_tl: Support generic include/.../excluded hooks	255
\DeclareUnknownKeysHandler to \DeclareUnknownKeyHandler	1097	2022-04-03 ltfinal.dtx v2.2s	
\ProcessKeyOptions: Allow for active characters in module argument	1097	General: Integration of new mark management interface	1260
\SetKeys: Allow for active characters in module argument	1099	2022-04-03 ltoutput.dtx v1.ih	
2022-02-19 ltcmd.dtx v1.0k		\@opcol: Interface with new mark mechanism	1177
\IfBlankTF: Added \IfBlankTF and friends	197	2022-04-04 ltpage.dtx v1.0n	
2022-02-20 ltfinal.dtx v2.2r		\markright: Interface with new mark mechanism	1034
\@uclclist: Use		2022-04-08 ltmath.dtx v1.2k	
\@expl@text@uppercase@n, removing local redefinition of \UTF@two@octets@noexpand	1274	\openup: Make \protected (gh/123)	849
use \text_lowercase:n	1274	2022-04-12 ltxref.dtx LaTeX2e	
2022-02-21 ltkeys.dtx v1.0e		\pageref: Macro reimplemented with a starred version	799
__keys_options_expand_module:Nn: Faster approach to module expansion	1097	\ref: Macro reimplemented with a starred version	799
2022-02-28 ltcmd.dtx v1.0k		2022-04-12 ltxref.dtx v1.1p	
General: Move latexrelease redefinitions from ltcmd.dtx	198	General: Add starred variants for the ref commands	797
2022-02-28 ltexpl.dtx v1.3f		\Ref: Macro reimplemented with a starred version	803
General: Move latexrelease redefinitions from ltcmd.dtx	76	2022-04-21 ltfinal.dtx v2.2t	
2022-02-28 ltvers.dtx v1.1i		\@uclclist: Support \noexpand in argument of	
\new@moduledate: Detect a missing \IncludeInRelease{0000/00/00}	40	\@expl@text@uppercase@n	1274
2022-03-10 ltbibl.dtx v1.1t		2022-05-06 ltmarks.dtx v1.0c	
\cite: Ensure that an empty argument generates a warning (gh/790)	999	__mark_new_class:n: Wrong command made \onlypreamble	1013

2022-05-08 ltmath.dtx v1.2l	2022-06-20 ltclass.dtx v1.5c
General: Use consistent math styles	<code>\load@onefilewithoptions</code> : Pass raw
under LuaTeX 856	options to <code>\ProcessKeyOptions</code> 1066
2022-05-08 ltshipout.dtx v1.0m	2022-06-20 ltkeys.dtx v1.0h
<code>\@kernel@after@enddocument@afterlastpage</code> :	<code>__keys_options_class:n</code> : Use raw
Handle case where	options data 1095
shipout/lastpage is run too early	<code>__keys_options_class:nmn</code> : New
(gh/813) 1146	function 1095
<code>__shipout_execute_main_cont:Nnnn</code> :	<code>__keys_options_global:n</code> : Use raw
Handle case where	options data 1094
shipout/lastpage is run too early	<code>__keys_options_local::</code> : Use raw
(gh/813) 1136	options data 1096
2022-05-13 lthooks.dtx v1.0u	<code>__keys_options_package:n</code> : Use raw
<code>__hook_use_once_clear:n</code> : Check if	options data 1096
prop exists to avoid l3debug error 293	2022-06-22 ltkeys.dtx v1.0i
2022-05-17 lthooks.dtx v1.0u	<code>.usage</code> : Add <code>.notif</code> property 1092
<code>__hook_initialize_hook_code:n</code> :	2022-06-30 ltfinal.dtx v2.2u
Refuse sorting one-time hooks	<code>\@uclclist</code> : Add
(gh/818). 270	<code>\AddToNoCaseChangeList</code> 1275
2022-05-17 ltmeta.dtx v1.0b	2022-06-30 ltfinal.dtx v2.2v
General: Default definition for targets	<code>\@uclclist</code> : Just use
added 428	<code>\text_lowercase:n</code> without
2022-05-27 ltfiles.dtx v1.2r	<code>\protectd@edf gh/881x</code> 1274
<code>\listfiles</code> : Try saved version string,	2022-07-04 ltfinal.dtx v2.2w
if ver@. is <code>\relax</code> (gh/825) . . . 472	<code>\@uclclist</code> : Introduced <code>\CaseSwitch</code> ,
2022-05-27 ltoutenc.dtx v2.0z	<code>\DeclareCaseChangeEquivalent</code>
General: Save the version string	and <code>\MakeTitlecase</code> to support
(gh/825) 520	hooking into case changing
2022-06-01 ltmarks.dtx v1.0d	gh/889 1274
<code>__mark_prepare_and_extract:nn</code> :	2022-07-04 ltfssbas.dtx v3.2k
Extend the logic for detecting the	<code>\frozen@everydisplay</code> : Ignore spaces
marks in the box (gh/836) 1016	if necessary (gh/886) 548
General: Marks are kernel errors . . 1025	<code>\frozen@everymath</code> : Ignore spaces if
2022-06-02 ltfinal.dtx v2.2u	necessary (gh/886) 548
<code>\@uclclist</code> : Add <code>\NoCaseChange</code> . 1275	2022-07-04 ltfssdcl.dtx v3.0z
2022-06-15 lthooks.dtx v1.0v	<code>\freeze@math@version</code> : Ignore spaces
<code>__hook_activate_generic:n</code> : Ensure	if necessary (gh/886) 672
that a newly activated generic	2022-07-05 ltkeys.dtx v1.0i
hook gets its execution code set . 239	<code>__keys_options_aux:n</code> : Support
2022-06-16 ltkeys.dtx v1.0g	<code>\CurrentOption</code> 1094
<code>__keys_options_class:n</code> : Better	<code>__keys_options_class:nmn</code> : Correct
handling of option removal . . . 1095	naming of raw class options
<code>__keys_options_package:n</code> : Better	storage 1095
handling of option removal . . . 1096	2022-07-23 ltkeys.dtx v1.0j
2022-06-19 ltkeys.dtx v1.0h	<code>__keys_options_end::</code> : Output
<code>__keys_options_class:n</code> : Further	‘public’ package name in
work on handling of option	messages 1094
removal 1095	<code>__keys_options_loaded:nn</code> : Output
<code>__keys_options_package:n</code> : Further	‘public’ package name in
work on handling of option	messages 1098
removal 1096	2022-08-07 ltfssbas.dtx v1.0g
<code>__keys_options_package:nmn</code> : New	<code>\DeclareEncodingSubset</code> : Make
function 1096	global declaration (gh/905) 539

2022-08-10 ltcmd.dtx v1.1a	
<code>__cmd_arg_to_keyvalue:nn</code> : New internal arg-to-keyval processor	182
<code>__cmd_grab_D_long_obey_spaces_no_strip:w</code> : Add support for skipping brace stripping	161
<code>__cmd_grab_R_aux:NNnN</code> : Track changes in D-type implementation	168
<code>__cmd_grab_b_end:Nw</code> : Track changes in D-type implementation	157
General: New switch	115
2022-08-13 ltuatex.dtx v1.1y	
General: shared_callbacks added	64
<code>add_to_callback</code> : Adapted code for shared_callbacks	68
<code>remove_from_callback</code> : Adapted code for shared_callbacks	70
<code>mlist_to_hlist</code> : Use shared_callback system for pre/post/mlist_to_hlist	72
2022-08-18 ltfilehook.dtx v1.0n	
<code>\@disable@packageload@do</code> : Inhibit checking the loaded version when package is load-disabled, and write to the .log (gh/888)	1119
2022-08-20 ltoutput.dtx v1.4j	
<code>\stockheight</code> : Register added	1164
<code>\stockwidth</code> : Register added	1164
2022-08-21 ltkeys.dtx v1.0k	
<code>__keys_options_loaded:nn</code> : Correct error message	1098
2022-09-03 ltmath.dtx v1.2m	
<code>\smash</code> : Guard against reboxing in fractions (gh/517)	848
2022-09-07 ltboxes.dtx v1.4d	
<code>\@iiiminipage</code> : Check for nested minipages and warn (gh/168)	891
<code>\if@in@minipage@env</code> : Check for nested minipages and warn (gh/168)	891
2022-09-17 ltfssdcl.dtx v3.1a	
<code>\DeclareMathVersion</code> : New logic for freezing math versions (gh/921)	675
<code>\freeze@math@version</code> : New logic for freezing math versions (gh/921)	671
2022-09-20 ltfssdcl.dtx v3.1b	
<code>\DeclareSymbolFont</code> : Drop any surplus 'm' in series argument (gh/918)	676
<code>\SetSymbolFont</code> : Drop any surplus 'm' in series argument (gh/918)	678
2022-10-03 ltuatex.dtx v1.2a	
General: Add rules for callback ordering	60
<code>add_to_callback</code> : Add rules for callback ordering	68
<code>declare_callback_rule</code> : Add function	69
<code>remove_from_callback</code> : Add rules for callback ordering	70
2022-10-10 ltclass.dtx v1.5d	
<code>\@unprocessedoptions</code> : Use <code>\protected@edef</code>	1073
<code>\load@onefilewithoptions</code> : Use <code>\protected@edef</code>	1066
<code>\PassOptionsToClass</code> : Use <code>\protected@xdef</code>	1054
<code>\ProcessOptions*</code> : Use <code>\protected@edef</code>	1056
2022-10-20 ltclass.dtx v1.5e	
<code>\load@onefilewithoptions</code> : Define key option handler in ltkeys	1066
2022-10-20 ltkeys.dtx v1.0l	
<code>__keys_options_aux:n</code> : Define key option handler in ltkeys	1094
<code>__keys_options_loaded:nn</code> : Correct an argument for a message	1098
2022-10-22 ltclass.dtx v1.5e	
<code>\@fileswithoptions</code> : Use <code>\protected@xdef</code>	1063
<code>\@use@option</code> : Use <code>\detokenize</code>	1058
<code>\ProcessOptions*</code> : Use <code>\detokenize</code>	1056, 1057
2022-10-22 ltdefs.dtx v1.5r	
<code>\@expandtwoargs</code> : Use <code>\protected@edef</code>	89
2022-10-22 ltkeys.dtx v1.0l	
<code>__keys_options_class:nnn</code> : Correct handling of unused option list	1095
2022-10-26 ltfinal.dtx v2.2x	
<code>\@uclclist</code> : Auto-detect babel locale	1274
Introduce optional argument for case-changing commands	1274
Make case changing commands language-aware	1274
2022-11-06 ltmiscen.dtx v1.2a	
<code>\enddocument</code> : Repeat release info at the end (gh/944)	821
2022-11-06 ltvers.dtx v1.1j	
General: Repeat release info at the end (gh/944)	38
2022-11-08 ltshipout.dtx v1.0n	
<code>__shipout_execute_main_cont:Nnnn</code> : Add shipout hook (gh/920)	1136
<code>shipout/lastpage</code> : Add shipout hook (gh/920)	1138

2022-11-16 ltclass.dtx v1.5f	2023-03-28 ltclass.dtx v1.5g
\endfilecontents: Do not show	\IfFormatAtLeastTF: Added
"current dir" in message	\IfFileAtLeastTF (gh/1015) .. 1047
(gh/917) 1076	2023-03-28 ltfinal.dtx v2.2z
Introduce key 'nowarn' on	\@uclclist: Use groups for gh/1021 1275
filecontents (gh/958) 1074–1076	2023-04-01 ltfssbas.dtx v3.2l
2022-11-24 ltdefs.dtx v1.5s	\math@version: Reset frozen
\DeclareEnvironmentCopy: Add	mathversion gh/1028 546
\NewEnvironmentCopy,	2023-04-02 ltfilehook.dtx v1.0o
\RenewEnvironmentCopy, and	\@set@curr@file@aux: Make
\DeclareEnvironmentCopy	\@set@curr@file@aux \long
(gh/963) 100	gh/942 1115
\ShowEnvironment: Added	2023-04-06 ltcmdhooks.dtx v1.0g
\ShowEnvironment 105	__hook_double_hashes_space:w:
2022-11-28 ltspace.dtx v1.3o	Add case for \c__hook_hashes_tl
\@hspace: Support calc syntax	(hook-args) 326
correctly (gh/967) 448	__hook_make_prefixes:w: Rename to
\@vspace@calcif: Support calc	\c__hook_hashes_tl (hook-args) 321
syntax without a group (gh/967) 435	\c__hook_hashes_tl: Rename to
2022-11-29 ltcmd.dtx v1.1b	\c__hook_hashes_tl and add
__cmd_show:x: Add	\c__hook_hash_tl (hook-args) . 315
\@showenvironmentlisthook ... 154	2023-04-06 lthooks.dtx v1.1a
2022-11-30 ltcmd.dtx v1.1b	__hook_activate_generic:n:
__cmd_show:x: Don't stop for the	Changes to add hook arguments
\begin part of an	(hook-args). 239
environment 152, 153	__hook_braced_real_loop:w: Macro
2022-11-30 ltfinal.dtx v2.2y	added (hook-args). 264
\@uclclist: Set \oe/\OE equal to act	__hook_chk_args_allowed:nn:
as a marker for babel 1275	Macro added (hook-args). 247
2023-01-05 ltfiles.dtx v1.2s	__hook_clear_next:n: Changes to
\document: \do now with default	add hook arguments (hook-args). 288
definition in the kernel (gh/975) 455	__hook_code_gset_aux:nnn: Macro
2023-01-19 ltluatex.dtx v1.2b	added (hook-args). 259
General: Remove unused local variable	__hook_code_gset_auxi:eeen:
tex_setattribute 54	Macro added (hook-args). 257
2023-01-30 ltpara.dtx v1.0l	__hook_cs_end:w: Macro added
\g__para_standard_everypar_tl:	(hook-args). 263
Backout \parskip at top of	__hook_cs_if_empty:c: Macro added
minipage (gh/989) 419	(hook-args). 262
2023-03-12 ltcmd.dtx v1.1c	__hook_gput_next_do:nn: Changes
__cmd_copy_expandable:NnNNNnnn:	to add hook arguments
Distinguish (non-expandable)	(hook-args). 288
document commands starting with	__hook_gremove_code:nn: Changes
__cmd_start_-	to add hook arguments
expandable:nNNNNn 147	(hook-args). 256
__cmd_show:x: Distinguish	__hook_if_execute_immediately:n:
(non-expandable) document	Changes to add hook arguments
commands starting with __cmd_-	(hook-args). 294
start_expandable:nNNNNn 152	__hook_init_structure:n: Changes
2023-03-22 ltspace.dtx v1.3p	to add hook arguments
\samepage: Add \predisplaypenalty	(hook-args). 235
setting (gh/1022) 433	__hook_initialize_all::: Changes
	to add hook arguments
	(hook-args). 269

<code>__hook_initialize_hook_code:n:</code> Changes to add hook arguments (hook-args).	270	Messages 'too-many-args', 'without-args' and 'one-time-args' added (hook-args).	298
<code>__hook_initialize_single:ccn:</code> Changes to add hook arguments (hook-args).	273	<code>\AddToHookNextWithArguments:</code> Add <code>\AddToHookNextWithArguments</code> (hook-args).	302
<code>__hook_log:nN:</code> Changes to add hook arguments (hook-args)	280	<code>\AddToHookWithArguments:</code> Add <code>\AddToHookWithArguments</code> (hook-args).	302
<code>__hook_log_next_code:n:</code> Changes to add hook arguments (hook-args).	285	<code>\c__hook_??_parameter_tl:</code> Token list added (hook-args).	265
<code>__hook_make_usable:nn:</code> Changes to add hook arguments (hook-args).	233	<code>\c__hook_nine_parameters_tl:</code> Add auxiliary token lists (hook-args).	229
<code>__hook_new:nn:</code> Add <code>\hook_new_with_args:nn</code> (hook-args).	232	<code>\g__hook_replacing_stack_seq:</code> Macro added (hook-args).	297
Update hook code after declaring.	233	<code>\hook_gput_next_code:nn:</code> Add <code>\hook_gput_next_code_with_</code> <code>args:nn</code> (hook-args).	286
<code>__hook_new_reversed:nn:</code> Add <code>\hook_new_reversed_with_</code> <code>args:nn</code> (hook-args).	236	<code>\hook_if_empty:n:</code> Changes to add hook arguments (hook-args).	295
<code>__hook_normalise_cs_args:nn:</code> Macro added (hook-args).	259	<code>\hook_new_pair_with_args:nnn:</code> Add <code>\hook_new_pair_with_args:nnn</code> (hook-args).	236
<code>__hook_parameter:n:</code> Macro added (hook-args).	264	<code>\hook_use_once:nnw:</code> Add <code>\hook_use_once:nnw</code> (hook-args).	292
<code>__hook_post_initialization_defs:::</code> Macro added (hook-args).	291	<code>\NewMirroredHookPairWithArguments:</code> Add <code>\NewHookWithArguments</code> (hook-args).	301
<code>__hook_prop_gput_labeled_do:Nnnn:</code> Add <code>\hook_gput_code_with_args:nnn</code> (hook-args).	244	<code>\UseOneTimeHookWithArguments:</code> Add <code>\UseHookWithArguments</code> (hook-args).	303
Changes to add hook arguments (hook-args).	244, 245	2023-04-11 ltfinal.dtx v2.3a <code>\@uclclist:</code> Use new generic mechanism to detect locale	1274
Macro added (hook-args).	246	2023-04-13 ltcmd.dtx v1.1d <code>__cmd_grab_v_group_end:::</code> Set <code>\tex_endlinechar:D</code> earlier (gh/876).	169
<code>__hook_set_normalise_fn:nn:</code> Macro added (hook-args).	260	2023-04-14 ltclass.dtx v1.5h <code>\load@onefilewithoptions:</code> Define <code>\load@onefilewithoptions</code> when in latexrelease (gh/992)	1065
<code>__hook_tl_set:cn:</code> Clean-up unused variants (hook-args).	229	2023-04-15 ltplain.dtx v2.3i <code>\extrafloats:</code> Protect box 255 in lualatex gh/1041	21
<code>__hook_try_declaring_generic_hook:wn:</code> Changes to add hook arguments (hook-args).	250	unwind numexpr and ifnum nesting	21
<code>__hook_use_i_delimit_by_s_mark:nw:</code> Use standard naming scheme (hook-args).	229	2023-04-16 ltfinal.dtx v2.3a <code>\BCPdata:</code> Command added	1273
<code>__hook_use_initialized:nnw:</code> Add <code>\hook_use:nnw</code> (hook-args).	290	2023-04-16 lthooks.dtx v1.1b <code>__hook_include_legacy_code_chunk:n:</code> <code>__hook_replacing_args_false:</code> in <code>__hook_include_legacy_</code> <code>code_chunk:n</code>	237
<code>__hook_use_once:nn:</code> Changes to add hook arguments (hook-args).	293		
<code>__hook_use_once_clear:n:</code> Changes to add hook arguments (hook-args).	293		
General: Add dedicated rollback code to revert data structures (hook-args).	307		

2023-04-19 ltfinal.dtx v2.3b			
\@uclclist: Add			
\DeclareLowercaseMapping,			
\DeclareTitlecaseMapping and			
\DeclareUppercaseMapping ...	1275		
2023-04-19 lthooks.dtx v1.1c			
__hook_gput_next_do:nn: Initialize			
hook structure when adding 'next'			
code (gh/1052).	288		
\hook_if_empty:n: Simpler and faster			
version (gh/1052).	295		
2023-05-12 ltxref.dtx v1.1q			
\label: (UFI)added a hook with			
argument	801		
(UFI)extended to store five			
arguments	801		
2023-05-13 ltmath.dtx v1.2n			
\leqno: add \ignorespaces gh/1059	856		
2023-05-15 ltfiles.dtx v1.2t			
\IfFileExists@: Use expl3 file			
existence test	466		
\IfFileExists@@: Macro added ...	466		
2023-05-21 ltcmdhooks.dtx v1.0h			
__hook_guess_arg_count:nw: Macro			
added (cmd-args)	330		
__hook_make_prefixes:w: Changes			
to allow support arguments in cmd			
hooks (cmd-args)	322		
__hook_patch_retokenize:Nnnn:			
Changes to allow support			
arguments in cmd hooks			
(cmd-args)	331		
2023-05-21 lthooks.dtx v1.1d			
__hook_if_cmd_hook:w: Changes to			
allow support arguments in cmd			
hooks (cmd-args).	297		
__hook_make_usable:nn: Changes to			
allow support arguments in cmd			
hooks (cmd-args).	234		
__hook_new:nn: Changes to allow			
support arguments in cmd hooks			
(cmd-args).	233		
__hook_try_declaring_generic_hook:wn:			
Changes to allow support			
arguments in cmd hooks			
(cmd-args).	250		
__hook_try_declaring_generic_next_hook:nn:			
Changes to allow support			
arguments in cmd hooks			
(cmd-args).	248		
\c__hook_parameter_cmd/./after_tl:			
Token lists added (cmd-args). . .	255		
2023-05-26 ltcmd.dtx v1.1e			
__cmd_defaults_error:w: Use			
simpler variant \cs_generate_-			
from_arg_count:NNno	124		
2023-05-30 ltfinal.dtx v2.3c			
\@uclclist: Fix a typo in			
implementation of			
\DeclareLowercaseMapping, etc.	1275		
2023-06-06 lthooks.dtx v1.1e			
__hook_code_gset_auxi:eeen:			
Short-circuit when the hook is			
declared without args (gh1078). .	257		
2023-06-14 ltmiscen.dtx v1.2b			
\@vobeytabs: Macro added	836		
\verbatim*: Support visible tabs in			
\verb*	838		
2023-06-14 ltspc.dtx v1.3q			
\@xobeytab: Macro added	447		
2023-06-15 ltmiscen.dtx v1.2b			
\@@sverb: Support visible tabs	840		
\@setupverbvisiblespace: Support			
visible tabs	839		
\@vobeyspaces: Support tabs	836		
2023-06-15 ltmiscen.dtx v1.2q			
\@setupverbvisibletab: Provide			
visible tab in \verb*	840		
2023-06-16 ltfiles.dtx v1.2u			
\IfFileExists@@: Support piped			
input	466		
2023-07-02 ltluatex.dtx v1.2c			
new_luafunction: Ensure existing			
table entries not overwritten			
gh/1100	59		
2023-08-03 ltluatex.dtx v1.2c			
mlist_to_hlist: Fix callback type of			
post_mlist_to_hlist_callback . . .	72		
2023-08-19 ltcmd.dtx v1.2a			
General: Removed commands that			
should have remained only in			
xparse.dtx	113		
2023-09-01 ltmiscen.dtx v1.2c			
\@vobeytabs: Provide global definition			
for active tab	836		
2023-09-06 ltmiscen.dtx v1.2b			
\enddocument: Test changes of values			
in \new@label@record	821		
2023-09-20 ltproperties.dtx v1.0c			
\RecordProperties: use			
\protected@edef for safer			
handling of active chars.	812		
\SetProperty: use \protected@edef	811		
2023-10-13 ltexpl.dtx v1.3g			
\IfExplAtLeastTF: Provide a test for			
expl3 date (gh/1004)	79		

2023-10-26 ltboxes.dtx v1.4e	2024-01-24 lthooks.dtx v1.1h
\@iframebox: Guard against unknown alignment gh/1072 886	__hook_if_usable_use:n: Correct usage of older
\@imakebox: Guard against unknown alignment gh/1072 881	__hook_if_file_hook:wTF (gh/1243) 292
\@parboxto: Guard against unknown alignment gh/1072 888	__hook_try_declaring_generic_hook_split:nNnnn: Correct usage of older
2023-10-26 ltspace.dtx v1.3r	__hook_if_file_hook:wTF (gh/1243) 249
\nobreakspace: Protected definition for tilde 446	2024-01-27 lttextcomp.dtx v1.1a
2023-11-07 ltcounts.dtx v1.1n	General: Added check file for encoding subset 787
\@definecounter: Do not change \the... if already defined (gh/823) 524	Adjusted/corrected TS1 sub-encoding declarations for various families (gh/1257) 766
2023-11-07 ltoutenc.dtx v2.1a	2024-01-29 ltmarks.dtx v1.0e
General: Add more explanation to error message (gh/1102) 519	__mark_extract_and_handle_marks:nn: Macro added 1015
2023-11-07 ltplain.dtx v2.3j	\mark_update_structure_from_material:nn: Macro renamed 1018
\tracingnone: Set \tracinglostchars to 2 in \tracingnone (gh/549) 35	\ShowMarksAt: Macro added 1028
2023-11-15 ltfiles.dtx v1.2v	2024-01-30 ltclass.dtx v1.5i
\if@listfiles@hashes: Extend file list information 472	\@fileswithoptions: Test group level 1062
\if@listfiles@sizes: Extend file list information 472	2024-03-18 ltthm.dtx v1.0g
\listfiles: Extend file list information 472	\@endtheorem: Insert link target in the label (UFI) 957
2023-11-16 ltpara.dtx v1.0m	\@thm: Use \@kernel@refstepcounter to avoid an unwanted target (UFI) 956
\g__para_standard_everypar_tl: Correct error message: hook left horizontal not vertical mode (gh/1182) 419	2024-03-21 ltcmd.dtx v1.2d
2023-12-01 ltcmd.dtx v1.2b	__cmd_grab_v_aux_put:N: Collect \endlinechar as \obeyedline . . 172
__cmd_cmd_type_cases:NnnnnnF: Extend for optimized commands 186	2024-03-22 ltclass.dtx v1.5j
__cmd_start_optimized::: Optimize cmd creation for all-m arguments 118	\load@onefilewithoptions: Apply one-step expansion to raw option list, to be consistent with change for gh/580 (gh/1298). 1066
2023-12-22 ltcmd.dtx v1.2c	2024-04-10 ltclass.dtx v1.5k
General: Generalize message invalid-bang (gh/1198) 188	\IfClassLoadedWithOptionsTF: Provide T and F conditionals not just TF (gh/1262) 1050
2024-01-03 lthooks.dtx v1.1g	\IfFileLoadedF: Provide \IfFileLoadedTF and variants (gh/1222) 1050
__hook_log:nN: Fix expansion of __hook__print_args:nn argument (gh/1221) 280	2024-04-17 ltcmd.dtx v1.2e
2024-01-13 ltkeys.dtx v1.0m	__cmd_cmd_type_cases:NnnnnnF: Use __kernel_cs_parameter_spec:N instead of \cs_argument_-spec:N/\cs_parameter_spec:N . . 186
__keys_options_class:n: Trim spaces off key names 1095	2024-04-17 ltcmdhooks.dtx v1.0j
__keys_options_package:n: Trim spaces off key names 1096	General: Use __kernel_cs_parameter_spec:N
2024-01-17 ltproperties.dtx v1.0d	
__property_record_value_aux:e: Use \protected@write 811	

instead of <code>\cs_argument_</code> - <code>spec:N/\cs_parameter_spec:N</code>	314	(gh/1359)	1025
2024-04-17 <code>ltdefs.dtx</code> v1.5t		2024-06-04 <code>ltclass.dtx</code> v1.5l	
General: Rename		<code>\cls@pkg</code> : Allow for extensions other than "cls" and "pkg"	1072
<code>\expl@cs@argument@spec@N</code> to <code>\expl@cs@parameter@spec@N</code> (gh/1014)	80	<code>\onefilewithoptions</code> : New argument <code>\pkgcls@ext</code> (gh/870)	1083
2024-04-17 <code>ltexpl.dtx</code> v1.3h		<code>\pkgcls@parse@date@arg</code> : New argument <code>\pkgcls@ext</code> (gh/870)	1085
General: Add a kernel-level copy of <code>\cs_parameter_spec:N</code>	77	2024-06-04 <code>ltxuatex.dtx</code> v1.2d	
Rename		<code>provides_module</code> : Add <code>v</code> to version string if required (gh/1364)	54
<code>\expl@cs@argument@spec@N</code> to <code>\expl@cs@parameter@spec@N</code> (gh/1014)	77	2024-06-10 <code>ltboxes.dtx</code> v1.4g	
Update name of <code>expl3</code> function	77	<code>\@finalstrut</code> : Always use a <code>\vrule</code> strut after all, but back up by a baseline if already in vertical mode. Otherwise empty table p-cells will not get the correct width (bug seen first with <code>colortbl</code>)	895
2024-04-17 <code>ltproperties.dtx</code> v1.0e		2024-06-10 <code>lttagging.dtx</code> v1.0b	
<code>\IfLabelExistsF</code> : Renamed <code>\IfLabelExistTF</code> to <code>\IfLabelExistsTF</code> (gh/1262)	815	<code>\tbl_crcr:n</code> : Always issue a <code>\crcr</code> even if we are at the start of a row to avoid problems with tabulary and similar code	1256
<code>\IfPropertyExistsF</code> : Renamed <code>\IfPropertyExistTF</code> to <code>\IfPropertyExistsTF</code> (gh/1262)	814	2024-06-17 <code>ltfssbas.dtx</code> v3.2m	
2024-04-18 <code>ltboxes.dtx</code> v1.4f		<code>\showhyphens</code> : set <code>\tracinglostchars</code> to 0 in <code>\showhyphens</code>	560
<code>\@finalstrut</code> : Use a <code>\hrule</code> strut not a <code>\vrule</code> if already in vertical mode (bug seen first with <code>footmisc/14</code>)	895	2024-06-19 <code>ltkeys.dtx</code> v1.0n	
2024-04-22 <code>lttextcomp.dtx</code> v2.1a		<code>_keys_options_class:n</code> : Refactor function	1095
General: Drop default option info (gh/1333)	776	<code>_keys_options_class:nn</code> : New function	1095
2024-04-24 <code>lttextcomp.dtx</code> v2.1b		<code>_keys_options_class:nnn</code> : Refactor function	1095
General: Load the 2018 version when rolling back prior to 2018-08-11 (gh/1333)	775	Track options used by classes	1095
2024-05-30 <code>ltmarks.dtx</code> v1.0f		<code>_keys_options_package:nn</code> : New functions	1096
<code>_mark_update_dblcol_structures::</code> Correct logic for first mark in page region if first column contains no marks (gh/1359)	1031	<code>_keys_options_package:nnn</code> : Skip options given to packages	1096
<code>_mark_value:nn</code> : Use sequence marker to make all marks unique on nearby regions (gh/1359)	1024	<code>_keys_scope:N</code> : New function	1092
<code>\mark_insert:nn</code> : Use sequence marker to make all marks unique on nearby regions (gh/1359)	1023	<code>_keys_scope:n</code> : New function	1092
<code>insertmark</code> : Use sequence marker to make all marks unique on nearby regions (gh/1359)	1024	<code>\l_keys_class_only_clist</code> : New variable	1093
2024-05-31 <code>ltmarks.dtx</code> v1.0g		<code>\l_keys_forced_global_clist</code> : New variable	1093
<code>\c_mark_empty_tl</code> : Initialize all marks with an id, use 0 when a new class is made (gh/1359)	1014	<code>.pass-to-packages</code> : New key property	1092
<code>\mark_use_last:nn</code> : Remove the id when returning the mark value		2024-06-20 <code>ltkeys.dtx</code> v1.0o	
		<code>_keys_scope:N</code> : Ensure only key name is stored	1092
		2024-06-23 <code>ltboxes.dtx</code> v1.4h	
		<code>\color@endgroup</code> : Adjust for new <code>@endpe</code> handling	883

2024-06-23 ltlists.dtx v1.0u	General: New message “empty-hook” (gh/1423).	299
\@doendpe: Set \if@endpe to false before calling \par (needed for tagging)	2024-08-16 ltluatex.dtx v1.2e	871
\propagate@doendpe: Set \if@endpe globally and also set up migration to the outside	provides_module: Support missing version string (gh/1443)	872
2024-06-23 ltmiscen.dtx v1.2d	2024-09-03 fontdef.dtx v3.0k	
\begin: Separate \begin and \end definitions for individual rollback	General: Load also t1cmss.fd and t1cmtt.fd	828
\end: Separate \begin and \end definitions for individual rollback	2024-09-04 ltlength.dtx v1.1e	829
\end\verbvisibleSPACE: Adjust for new @endpe handling	\@settodim: (UF) Suspend tagging .	831
2024-06-23 ltpara.dtx v1.0n	2024-09-05 ltproperties.dtx v1.0f	
\g__para_standard_everypar_tl: Append \everypar toks to \g__para_standard_everypar_tl, rollback 2023/06/01 (gh/1386) . .	__property_record_value_aux:e: Remove \if@filesW test to be in line with \label, tagging-project issue 696	420
2024-06-24 fontdef.dtx v3.0j	2024-09-10 ltspace.dtx v1.3s	
General: load ts1 cmr fd file in Unicode engines	\addpenalty: Drop unnecessary \@noitemerr and instead generate \@LRmoderr if we are in restricted hmode (gh/1460)	727
2024-07-06 ltcmd.dtx v1.2f	\addvspace: Drop unnecessary \@noitemerr and instead generate \@LRmoderr if we are in restricted hmode (gh/1460)	442
__cmd_declare_env:ennn: Use space-trimmed envname directly (gh/1399)	2024-09-11 lterror.dtx v1.2u	441
\NewDocumentEnvironment: Trim spaces from envname first (gh/1399)	\@LRmoderr: Error message added .	406
2024-07-10 ltmiscen.dtx v1.2e	2024-09-13 ltagging.dtx v1.0h	
General: Drop code chunks before adding them to avoid duplication in rollback (gh/1407)	tagsupport/float/hmode/end: Sockets for floats added	1248
\enddocument: Drop code chunks before adding them to avoid duplication in rollback (gh/1407)	2024-09-20 ltcounts.dtx v1.1o	821
2024-07-13 ltagging.dtx v1.0c	\@addtoreset: add the parent theHfoo if a counter is reset	525
tagsupport/tbl/leaders/end: Sockets for \cline leaders added (tagging/134)	\@definecounter: define theHfoo (used for internal links)	524
2024-08-03 ltclass.dtx v1.5m	2024-09-20 ltagging.dtx v1.0h	
\pkgcls@use@this@release: Add selected release to the file list (gh/1413)	General: moved \@kernel@refstepcounter into ltxref	1257
2024-08-09 lthooks.dtx v1.1i	2024-09-20 ltxref.dtx v1.1r	
__hook_normalize_hook_args_aux:Nn: Distinguish between empty label and empty hook (gh/1423)	\refstepcounter: add sockets	801
__hook_parse_dot_label:nN: Distinguish between empty label and empty hook (gh/1423)	provide a kernel copy kernel@refstepcounter	801
__hook_parse_label_default:nN: Distinguish between empty label and empty hook (gh/1423)	set also currentHref	801
	2024-09-21 ltmeta.dtx v1.0c	
	General: Added tagging support . . .	428
	2024-09-25 ltproperties.dtx v1.0g	
	\IfLabelExistsF: Fixed definitions of \IfLabelExistsT and \IfLabelExistsF	815
	2024-10-10 ltagging.dtx v1.0i	
	default: Restore also paratagging (tagging/723)	1245

2024-10-12 ltmiscen.dtx v1.2f	<code>\begin</code> : Make <code>\begin</code> engine-protected	828	<code>\mark_use_last:nn</code> :	1024
2024-10-21 lthooks.dtx v1.1j	<code>\IfHookEmptyF</code> : Define <code>\IfHookEmptyT</code> , <code>\IfHookEmptyF</code> .	304	2024-11-12 ltfinal.dtx v2.3d <code>\@uclclist</code> : Add option to titlecase first or all words	1274
2024-10-21 ltmarks.dtx v1.0h	<code>\IfMarksEqualF</code> : Define <code>\IfMarksEqualT</code> , <code>\IfMarksEqualF</code>	1029	Use updated titlecase-first function in expl3	1274
2024-10-21 ltproperties.dtx v1.0h	<code>\IfPropertyRecordedF</code> : Define <code>\IfPropertyRecordedT</code> , <code>\IfPropertyRecordedF</code>	815	2024-11-16 ltpage.dtx v1.0o <code>\markright</code> : Drop legacy mark support	1034
2024-10-21 ltagging.dtx v1.0k	<code>\UseExpandableTaggingSocket</code> : Added expandable variants . . .	1244	2024-11-19 fontdef.dtx v3.0l General: Preload the TS1 .fd file . .	727
2024-10-26 ltcounts.dtx v1.1p	Changed behavior of two argument tagging sockets when disabled. .	1244	Preload the TS1 <code>ts1cmr.fd</code> file in all engines	726
2024-10-27 ltsockets.dtx v0.9b	<code>\addtoreset</code> : Fully expand counter name in the <code>Hfoo</code> commands (gh/1508)	525	2024-11-21 ltagging.dtx v1.0l <code>\UseExpandableTaggingSocket</code> : Define <code>\tag_if_active:TF</code> conditionals here (github/1558)	1244
2024-10-27 ltsockets.dtx v0.9b	<code>\socket_set_plug:nnn</code> : Make plug definition non-protected	345	2024-11-26 lthooks.dtx v1.1l <code>__hook_initialize_all::</code> Adjust debugging message (gh/1459) . .	269
2024-10-29 lthooks.dtx v1.1k	<code>\socket_use_expandable:n</code> : Added <code>\socket_use_expandable:n</code>	348	<code>__hook_initialize_hook_code:n</code> : Adjust debugging message (gh/1459)	270
2024-11-02 ltdirchk.dtx v1.3b	<code>__hook_list_if_rule_exists:nnnF</code> : Skip mapping over undeclared <code>\g__hook_{hook}_code_prop</code> (gh/1513)	285	<code>__hook_initialize_single:ccn</code> : Adjust debugging message (gh/1459)	274
2024-11-02 ltplain.dtx v1.3k	<code>__hook_log:nN</code> : Skip mapping over undeclared <code>\g__hook_{hook}_code_prop</code> (gh/1513)	281	<code>__hook_prop_gput_labeled_do:Nnnn</code> : Adjust debugging message (gh/1459)	245
2024-11-06 ltoutenc.dtx v2.1b	General: Add tab char to <code>\dospecials</code>	4	<code>\hook_gput_next_code:nn</code> : Add debugging message (gh/1459)	286, 287
2024-11-06 ltoutenc.dtx v2.1b	General: Support for macron-i for OT1	493	2024-11-26 ltoutenc.dtx v2.1c <code>\@inmathwarn</code> : Declare command for logging (gh/1242)	479
2024-11-09 ltmarks.dtx v1.1a	Support for macron-i for T1 (gh/1453)	499	<code>\@strip@args</code> : Alter error and info message	483
2024-11-09 ltmarks.dtx v1.1a	<code>__mark_class_status:nnn</code> : Add a third argument	1027	<code>\UndeclareTextCommand</code> : Log text command change (gh/1242)	486, 487
2024-11-09 ltmarks.dtx v1.1a	<code>__mark_status:nn</code> : Add a second argument	1028	Use <code>\ifcsname</code> to avoid generating a csname	486
2024-11-09 ltmarks.dtx v1.1a	<code>\mark_clear_structure:n</code> :	1022	2024-11-27 ltfssdcl.dtx v3.1d <code>\freeze@math@version</code> : Reset math version if necessary (gh/1101 and gh/1028)	671
2024-11-09 ltmarks.dtx v1.1a	<code>\mark_copy_structure:nn</code> : Macro renamed	1022	Reset top-level alphabet definitions only for the current math version (gh/1101 and gh/1028)	672
			2024-11-30 ltfssbas.dtx v3.2o <code>\@math@level</code> : New approach to <code>localmathalphabets</code> (gh/1101) .	547
			<code>\frozen@everydisplay</code> : New approach to <code>localmathalphabets</code> (gh/1101)	548

<ul style="list-style-type: none"> <code>\frozen@everymath</code>: New approach to localmathalphabets (gh/1101) . 548 <code>\math@version</code>: New approach to localmathalphabets (gh/1101) 546, 547 	<ul style="list-style-type: none"> not expand mark content while debugging 1021
2024-12-03 ltmmarks.dtx v1.1b	2025-01-15 ltkeys.dtx v1.0p
<ul style="list-style-type: none"> <code>\c__mark_empty_tl</code>: Fix inconsistent local/global assignment (gh/1574) 1014 	<ul style="list-style-type: none"> <code>__keys_options_aux:n</code>: Only process global options on first pass . . . 1094
2024-12-11 ltfssaxes.dtx v1.0k	2025-01-21 ltcmd.dtx v1.2g
<ul style="list-style-type: none"> General: Add <code>ssc</code> shape change rules (gh/1581) 624 Rationalize <code>sw</code> shape change rules (gh/1581) 624 Reorganized the rollback data . . . 622 	<ul style="list-style-type: none"> <code>__cmd_grab_v_aux_catcodes::</code> Store spaces and tabs as active chars . 171 <code>__cmd_grab_v_aux_put:N</code>: Simplify catcode handling 172
2024-12-13 ltfssaxes.dtx 1.0k	2025-01-21 ltoutput.dtx v1.4l
<ul style="list-style-type: none"> General: Add numerous <code>\DeclareFontSeriesChangeRule</code> entries to support the full range of weights (from <code>ul</code> to <code>ub</code>) and widths (from <code>uc</code> to <code>ux</code>) (gh/1396) 565 Minor modifications to a few <code>\DeclareFontSeriesChangeRule</code> entries (gh/1396) 565 	<ul style="list-style-type: none"> <code>\@writsetup</code>: Support extended syntax for <code>\label</code>, <code>\index</code>, and <code>\glossary</code> (gh/311) 1190
2024-12-21 ltoutenc.dtx v2.1c	2025-01-21 ltsect.dtx v1.1g
<ul style="list-style-type: none"> General: Correct base letter (gh/1587) 499 	<ul style="list-style-type: none"> <code>\@gobble@with@sphack@som</code>: Support extended syntax for <code>\label</code>, <code>\index</code> and <code>\glossary</code> (gh/311) . 968 <code>\addtocontents</code>: Support extended syntax for <code>\label</code>, <code>\index</code>, and <code>\glossary</code> (gh/311) 968
2024-12-22 ltcmdhooks.dtx v1.0k	2025-01-22 ltcounts.dtx v1.2a
<ul style="list-style-type: none"> <code>__hook_try_put_cmd_hook:w</code>: Avoid defining command while adding a cmd hook (gh/1591) 316 	<ul style="list-style-type: none"> <code>\@ifbothcounters</code>: Guard against star alias. 525 <code>\c@*</code>: <code>\c@*</code> added (gh/1632) 523
2024-12-27 ltsockets.dtx v0.9c	2025-01-22 ltxref.dtx v1.1t
<ul style="list-style-type: none"> <code>\IfSocketPlugAssignedF</code>: Conditionals for sockets, plugs, and assignments (gh/1577) 349 <code>\socket_if_exist:n</code>: Conditionals for sockets, plugs, and assignments (gh/1577) 345 <code>\socket_if_plug_assigned:nn</code>: Conditionals for sockets, plugs, and assignments (gh/1577) 347 <code>\socket_if_plug_exist:nn</code>: Conditionals for sockets, plugs, and assignments (gh/1577) 346 	<ul style="list-style-type: none"> <code>\refstepcounter</code>: Guard * from causing infinite loop 801 Macro added 801
2025-01-02 ltclass.dtx v1.5n	2025-01-23 ltcmd.dtx v1.3a
<ul style="list-style-type: none"> <code>\AtEndDocument</code>: Do not make <code>\AtBeginDocument</code> preamble only (gh/1604) 1072 	<ul style="list-style-type: none"> <code>__cmd_add_grabber:N</code>: Extend to support c-type grabbing 140 <code>__cmd_add_type_b:w</code>: Extend to cover c-type grabbing 137 <code>__cmd_add_type_b_or_c:N</code>: New function 137 <code>__cmd_add_type_c:w</code>: New function . 137 <code>__cmd_grab_D_aux:NNnNNNN</code>: Extend to support c-type grabbing 162 <code>__cmd_grab_D_long_obey_spaces_no_strip:w</code>: Auto-generate D-grabbers 161 <code>__cmd_grab_D_verb_safe:NN</code>: New macro 162 <code>__cmd_grab_c:w</code>: New c-type grabbing function 157 <code>__cmd_normalize_type_b:w</code>: Extend to cover c-type grabbing 132 <code>__cmd_normalize_type_b_or_c:nn</code>: New function 132 <code>__cmd_normalize_type_c:w</code>: New function 132 <code>__cmd_prepare_signature:n</code>: Extend to cover c-type grabbing 135
2025-01-03 lthooks.dtx v1.1l	
<ul style="list-style-type: none"> General: Correct example to use <code>env/quote/begin</code> instead of <code>/before</code> (gh/1599) 203 	
2025-01-10 ltmmarks.dtx v1.1c	
<ul style="list-style-type: none"> <code>__mark_get_from_splitmarks::</code> Do 	

General: Generalize message			
arg-after-body	188		
New message			
chars-dropped-first-line	189		
New message			
chars-dropped-last-line	189		
New variable	114, 116		
2025-01-23 ltoutput.dtx v1.4m			
\writsetup : Make \label , \index			
and \glossary truly invisible			
when typesetting (gh/1638) . . .	1190		
2025-01-23 ltsect.dtx v1.1h			
\@gobble@with@sphack@som : Make			
\label , \index and \glossary			
truly invisible when typesetting			
(gh/1638)	968		
2025-01-26 ltlists.dtx v1.0v			
\propagate@doendpe : Only migrate			
\@doendpe out of simple and			
semi-simple groups (gh1641) . . .	872		
2025-01-27 lttagging.dtx 1.0m			
General: add sockets for math			
phantom commands	1251		
2025-01-29 ltoutput.dtx v1.4n			
\@addtocurcol : Save current value of			
\@reqcolroom (gh/1645)	1203		
\@addtonextcol : Save current value of			
\@reqcolroom (gh/1645)	1212		
\@flcheckspace : Reset current value			
of \@reqcolroom (gh/1645) . . .	1232		
2025-02-01 ltcmd.dtx v1.3b			
\ProcessList : Use			
\tl_map_tokens:n instead of			
\tl_map_function:n	198		
2025-02-11 ltoutenc.dtx v2.1c			
\@inmathwarn : Log text			
command/symbol redeclarations			
(gh/1242)	479		
2025-02-14 ltoutput.dtx v1.4m			
\@combinefloats : Macro got a new			
name	1194		
\@if@bottomfloats@TF : Macro			
added	1184		
\@if@flushbottom@TF : Macro added	1183		
\@if@footnotes@TF : Macro added	1183		
\@make@normalcolbox : Macro added	1179		
\@make@specialcolbox : Macro			
added	1179		
\@makecol : Use sockets and hooks	1178		
\@outputbox@append : Macro added	1182		
\@outputbox@appendfootnotes :			
Macro added	1182		
\@outputbox@attachbottomfloats :			
Macro added	1183		
\@outputbox@attachfloats : Macro			
added	1183		
\@outputbox@attachtopfloats :			
Macro added	1183		
\@outputbox@depth : Macro added	1179		
\@outputbox@removebskip : Finally fix			
bottom skip bug (from 2.09) . .	1181		
Macro added	1181		
\@writsetup : Hooks and sockets			
added	1188		
General: Use sockets and hooks in			
OR	1184		
build/column/after : Hook added	1186		
build/column/outputbox : Socket			
added	1184		
build/page/after : Hook added . .	1186		
build/page/reset : Hook added . .	1186		
2025-02-14 lttagging.dtx v1.0m			
General: Tagging support for output			
routines added	1248		
tagsupport/build/column/footins :			
Tagging socket added	1249		
tagsupport/build/column/outputbox :			
Tagging socket added	1249		
tagsupport/build/page/footer :			
Tagging socket added	1248		
tagsupport/build/page/header :			
Tagging socket added	1248		
2025-02-16 lttagging.dtx v1.0n			
tagsupport/build/page/footer :			
Initialize tagging sockets with 1 or			
2 arguments	1248		
tagsupport/caption/end : Initialize			
tagging sockets with 1 or 2			
arguments	1248		
tagsupport/tbl/colspan : Initialize			
tagging sockets with 1 or 2			
arguments	1247		
tagsupport/toc/contentsline/after :			
Initialize tagging sockets with 1 or			
2 arguments	1246		
tagsupport/toc/starttoc/after :			
Initialize tagging sockets with 1 or			
2 arguments	1246		
2025-02-19 lttagging.dtx v1.0n			
General: Moved math sockets into			
lttagging	1249		
2025-02-20 ltfssbas.dtx v3.2p			
\CheckEncodingSubset : If necessary,			
load the .fd before checking the			
encoding subset (gh/1669)	540		
2025-02-21 lttagging.dtx v1.0n			
tagsupport/marginpar/end : move			
marginpar sockets	1246		

v1.0b lttemplates.dtx 2024-02-15		contents	382
\IfInstanceExistsTF: New macros	389	v1.0d lttemplates.dtx 2025-01-08	
v1.0c lttemplates.dtx 2024-04-17		\SetTemplateKeys: Test for empty	
\IfInstanceExistsTF: Use plural		key/val list to speed up processing	390
names	389	v1.0e lttemplates.dtx 2025-01-20	
v1.0d lttemplates.dtx 2024-10-07		__template_declare_template_code:nnnnn:	
__template_assign_variable::		Speed up test for	
Correct passing of \KeyValue		\AssignTemplateKeys	370

Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

Symbols	
$\backslash!$	02:431, 02:433, 57:518
$\backslash!$	<u>38:214</u>
$\backslash"$	04:512, 04:513, 21:246, 21:398, 21:439, 21:477, 21:489, 21:600, 21:632, 21:659, 21:667, 21:673, 21:677, 21:683, 21:687, 21:693, 21:699, 21:706, 21:707, 21:713, 21:717, 21:768, 21:814, 21:1266, 21:1284, 21:1290, 21:1294, 21:1300, 21:1304, 21:1310, 21:1316, 21:1323, 21:1324, 21:1330, 21:1334, 21:1336, 21:1443, 24:513, 33:135, 33:162, 57:519
$\backslash\#$	01:46, 01:59, 02:6, 02:14, 02:531, 06:801, 09:463, 24:500, 57:502
$\backslash\$$	01:58, 02:4, 02:13, 06:800, 21:327, 21:464, 21:471, 21:586, 21:826, 21:833, 33:595, 33:602, 57:503
$\backslash\$$	<u>1299</u>
$\backslash\%$	01:59, 01:89, 01:91, 01:111, 02:14, 02:529, 06:801, 21:512, 21:514, 24:502, 33:604, 33:724, 37:210, 50:1566, 50:1567, 57:504
$\backslash\&$	01:58, 02:5, 02:13, 02:530, 06:800, 50:439, 57:505
\backslash'	02:551, 21:247, 21:399, 21:441, 21:475, 21:486, 21:602, 21:612, 21:618, 21:620, 21:623, 21:625, 21:633, 21:639, 21:645, 21:647, 21:650, 21:652, 21:660, 21:664, 21:671, 21:675, 21:680, 21:685, 21:688, 21:690, 21:697, 21:702, 21:703, 21:710, 21:715, 21:718, 21:769, 21:816, 21:835, 21:837, 21:838, 21:839, 21:842, 21:844, 21:845, 21:846, 21:848, 21:849, 21:1260, 21:1281, 21:1288, 21:1292, 21:1297, 21:1302, 21:1305, 21:1307, 21:1314, 21:1319, 21:1320, 21:1327, 21:1332, 21:1335, 21:1343, 21:1344, 21:1391, 21:1392, 21:1397, 21:1398, 21:1409, 21:1410, 21:1415, 21:1416, 21:1444, 21:1445, 21:1459, 21:1460, 21:1461, 21:1462, 21:1475, 21:1476, 24:512, 29:776, 30:264, 33:125, 37:738, 38:254, 40:387, 40:408, 41:72, 54:783, 57:520
$\backslash($	07:2619, 38:345
$\backslash($	<u>38:271</u>
$\backslash)$	02:551, 07:2641, 38:346
$\backslash)$	<u>38:271</u>
$\backslash*$	22:111, 24:505, 50:1247, 50:1378, 50:1492, 50:1568
$\backslash*$	<u>38:251</u>
$\backslash+$	41:72
$\backslash+$	<u>1297</u>
$\backslash,$	02:432, 02:434, 30:517, 37:738, 38:7, 38:8, 38:40, 38:167, 38:169, 38:172, 38:197, 38:220, 38:221, 38:237
$\backslash,$	<u>1338</u>
$\backslash-$	02:378, 06:24, 18:463, 18:554, 21:436, 21:437, 21:595, 21:810, 21:811, 24:507, 37:738, 40:386, 40:407, 41:72, 57:224, 57:264
$\backslash-$	<u>06:835</u> , <u>1339</u>
$\backslash.$	02:431, 02:433, 20:45, 20:115, 20:173, 21:248, 21:400, 21:472, 21:473, 21:495, 21:608, 21:609, 21:635, 21:636, 21:662, 21:770, 21:840, 21:847, 21:1265, 21:1347, 21:1348, 21:1357, 21:1358, 21:1367, 21:1368, 21:1385, 21:1446, 21:1447, 21:1483, 21:1484, 24:506, 33:137, 33:163
$\backslash.$	<u>1326</u>
$\backslash.\text{default}$	<u>564</u>
$\backslash.\text{code}$	<u>51:4</u>
$\backslash.\text{if}$	<u>51:4</u>
$\backslash.\text{ifnot}$	<u>51:4</u>
$\backslash.\text{pass-to-packages}$	<u>51:22</u>
$\backslash.\text{store}$	<u>51:4</u>
$\backslash.\text{usage}$	<u>51:4</u>
$\backslash/$	01:81, 06:25, 09:219, 09:294, 24:454, 24:508, 50:438
$\backslash/$	<u>1293</u>
$\backslash:$	<u>38:214</u> , 38:224, 38:225, 38:226, 38:242, <u>38:252</u> , 38:252, 06:795, 06:796, 02:432, 02:434
$\backslash;$	02:432, 02:434, 30:511, 38:198
$\backslash;$	<u>38:214</u> , <u>850</u>
$\backslash<$	21:596, 21:761, 24:503, 37:738, 41:71, 41:109
$\backslash=$	21:249, 21:401, 21:478, 21:494, 21:735, 21:771, 21:1263, 21:1337, 21:1338, 21:1353, 21:1354, 21:1376, 21:1377, 21:1378, 21:1403, 21:1404, 21:1429, 21:1430, 21:1463, 21:1464, 21:1465, 21:1466,

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpara.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

21:1481, 21:1482, 21:1489, 21:1490,	\@outputbox@reinsertbskip	1170
29:776, 33:143, 40:387, 40:408, 41:71	\@restorepar	408
\> 21:593, 21:762, 24:504, 37:738,	\@setpar	408
38:226, 38:241, 38:242, 38:252, 41:71	@topnum commands:	
\? 02:431, 02:433, 57:520	\@topnum:	54:1323
\@@par 408	\[. 24:509, 29:44,	
\@TeXversion 6, 1	29:55, 29:77, 29:88, 38:347, 57:521	
@botlist commands:	\[. 38:289, 38:462, 1328	
\@botlist:	\ \ 01:58, 01:231, 01:232, 01:233,	
. 54:1303, 54:1398, 54:1547, 54:1701	01:234, 01:237, 01:244, 01:245,	
@botnum commands:	01:246, 01:247, 01:250, 01:257,	
\@botnum: 54:1277	01:258, 01:259, 01:260, 01:263,	
\@car 82	01:270, 01:276, 01:277, 01:281,	
\@cdr 82	01:283, 01:284, 01:288, 01:293,	
@citeb commands:	01:294, 01:297, 01:303, 02:13,	
\@citeb: 47:36, 47:65, 47:82	02:377, 04:272, 04:424, 06:231,	
@colht commands:	06:288, 06:449, 06:572, 06:590,	
\@colht: 54:525, 54:538	06:800, 07:471, 07:535, 07:566,	
@colnum commands:	07:712, 07:724, 07:755, 07:1152,	
\@colnum: 54:1463, 54:1608, 54:1775	07:1158, 07:2802, 07:2830, 07:2856,	
\@cons 82	07:2863, 07:2892, 07:2989, 07:3017,	
\@currdir 5, 1	07:3028, 07:3054, 07:3060, 07:3067,	
@currname commands:	08:1778, 08:1789, 08:2597, 08:2611,	
\@currname: 20:761	08:2612, 08:2613, 08:2614, 08:2615,	
\@dblarg 81	08:2620, 08:2623, 08:2624, 08:2625,	
@dbldeferlist commands:	08:2635, 08:2636, 08:2650, 08:2671,	
\@dbldeferlist: 54:2255, 54:2297	08:2675, 08:2676, 08:2678, 08:2684,	
@dbltopnum commands:	08:2695, 08:2696, 08:2701, 08:2706,	
\@dbltopnum: 54:2154, 54:2281	08:2711, 08:2727, 08:2892, 08:2893,	
@deferlist commands:	08:2894, 09:460, 09:598, 11:972,	
\@deferlist: 54:1389, 54:1486,	11:980, 11:986, 11:987, 11:988,	
54:1538, 54:1631, 54:1691, 54:1799,	11:989, 11:1002, 11:1008, 11:1009,	
54:1844, 54:1873, 54:1925, 54:1954,	11:1015, 11:1021, 11:1022, 11:1028,	
54:2011, 54:2043, 54:2129, 54:2169	11:1029, 11:1036, 11:1043, 11:1050,	
\@if@bottomfloats@TF 1170	11:1057, 11:1058, 11:1067, 11:1085,	
\@if@footnotes@TF 1170	11:1086, 11:1116, 11:1117, 11:1118,	
\@ifclasslater 1041	11:1119, 11:1120, 11:1121, 11:1122,	
\@ifclassloaded 1041	11:1123, 11:1124, 11:1125, 11:1126,	
\@ifclasswith 1041	14:279, 16:146, 16:153, 16:160,	
\@ifdefinable 81	16:161, 16:162, 16:163, 18:556,	
\@ifnextchar 81	20:754, 20:773, 21:581, 24:497,	
\@ifpackagelater 1041	30:262, 36:248, 37:451, 37:456,	
\@ifpackageloaded 1041	37:461, 37:471, 37:475, 37:479,	
\@ifpackagewith 1041	37:520, 38:370, 38:523, 40:419,	
\@ifstar 81	40:578, 40:580, 41:73, 41:170,	
\@ifundefined 81	41:180, 41:194, 42:139, 57:506, 430	
\@missingfileerror 1042	\ \ 18:53, 1302	
\@namedef 81	\ { 01:6, 01:10, 01:58,	
\@nameuse 81	02:2, 02:13, 07:638, 07:2145, 09:461,	
\@outputbox@append 1170	14:22, 21:328, 21:583, 24:498,	
\@outputbox@appendfootnotes 1170	30:260, 37:519, 38:59, 38:167, 57:509	
\@outputbox@attachbottomfloats 1170	\ } 01:11, 01:58, 02:3,	
\@outputbox@attachfloats 1170	02:13, 09:462, 14:21, 21:329, 21:584,	
\@outputbox@attachtopfloats 1170	24:499, 30:261, 37:519, 38:59, 57:510	

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

<code>\<addto-cmd></code>	201	02:8, 02:14, 06:801, 57:508, 01:59, 1306
<code>\<cmd></code> _□	147	<code>_hook\textvisiblespace</code> _□ <code>\meta</code> _□ <code>{hook}</code>
<code>\<cmd></code> _□ <code>(arg</code> _□ <code>num)</code>	149 230
<code>\<cmd></code> _{□□}	147	<code>_hook_next\textvisiblespace</code> _□ <code>\meta</code> _□ <code>{hook}</code>
<code>\<cmd></code> _□ code	146 230
<code>\<cmd></code> _□ defaults	147	<code>_hook_toplevel\textvisiblespace</code> _□ <code>\meta</code> _□ <code>{hook}</code>
<code>\<filename></code>	1115 230
<code>\<function></code>	174	<code>\`</code>
<code>_</code>	01:58, 01:75, 02:13, 02:381, 02:437, 02:466, 02:467, 02:484, 06:800, 07:1818, 07:2183, 08:2676, 11:1009, 11:1022, 11:1029, 11:1037, 11:1058, 11:1086, 14:19, 14:20, 14:21, 14:22, 14:25, 18:468, 24:494, 24:744, 24:781, 24:806, 30:263, 37:503, 37:504, 37:513, 37:514, 37:659, 37:675, 37:689, 43:53, 43:55, 43:60, 43:62, 47:37, 50:432, 57:501	21:252, 21:403, 21:440, 21:474, 21:485, 21:601, 21:663, 21:670, 21:674, 21:679, 21:684, 21:689, 21:696, 21:700, 21:701, 21:709, 21:714, 21:773, 21:815, 21:1259, 21:1280, 21:1287, 21:1291, 21:1296, 21:1301, 21:1306, 21:1313, 21:1317, 21:1318, 21:1326, 21:1331, 24:511, 29:776, 33:139, 37:738, 40:387, 40:408, 41:72, 57:523
<code>\]</code>	02:551, 24:510, 38:348, 57:522	<code>\ </code>
<code>\]</code>	38:289, 38:486	22:222, 22:233, 30:584, 30:585, 57:524
<code>\^</code>	01:47, 01:56, 01:59, 01:103, 01:314, 02:7, 02:9, 02:11, 02:14, 02:380, 02:437, 02:438, 02:457, 02:458, 02:479, 02:480, 06:20, 06:801, 07:6, 07:7, 07:1603, 07:1606, 07:1610, 07:1643, 07:1689, 07:1728, 07:1792, 07:1816, 07:2084, 07:2184, 07:2487, 07:2492, 07:3282, 18:556, 18:558, 18:560, 21:251, 21:306, 21:402, 21:476, 21:487, 21:579, 21:665, 21:672, 21:676, 21:681, 21:686, 21:691, 21:698, 21:704, 21:705, 21:711, 21:716, 21:772, 21:1261, 21:1278, 21:1282, 21:1289, 21:1293, 21:1298, 21:1303, 21:1308, 21:1315, 21:1321, 21:1322, 21:1328, 21:1333, 21:1345, 21:1346, 21:1363, 21:1364, 21:1371, 21:1372, 21:1386, 21:1387, 21:1388, 21:1417, 21:1418, 21:1439, 21:1440, 21:1441, 21:1442, 24:495, 24:496, 24:501, 33:133, 33:161, 37:202, 37:211, 37:505, 37:506, 37:660, 50:1248, 50:1249, 50:1250, 50:1331, 50:1334, 50:1337, 50:1379, 50:1380, 50:1381, 50:1463, 50:1466, 50:1469, 50:1493, 50:1494, 50:1495, 50:1553, 50:1556, 50:1559, 57:255, 57:256, 57:257, 57:258, 57:259, 57:260, 57:261, 57:262, 57:263, 57:507, 57:513, 57:514, 57:515, 57:516, 57:550, 57:551, 57:552, 57:553, 57:554, 57:555, 57:556, 57:557, 57:558	<code>\~</code>
<code>_</code>	21:334, 30:265, 38:269, 38:270,	01:59, 02:10, 02:14, 06:801, 14:20, 18:469, 21:259, 21:307, 21:404, 21:488, 21:580, 21:666, 21:678, 21:682, 21:692, 21:708, 21:712, 21:774, 21:1262, 21:1279, 21:1283, 21:1295, 21:1299, 21:1309, 21:1325, 21:1329, 21:1373, 21:1374, 21:1375, 21:1427, 21:1428, 33:151, 33:171, 37:653, 37:670, 37:684, 37:699, 37:742, 57:511

A

<code>\A</code>	57:252, 57:526, 57:547
<code>\a</code>	21:243, 41:1, 57:243, 57:527, 57:538, 1309
<code>\AA</code>	21:260, 21:448, 21:547, 02:443, 1321
<code>\aa</code>	21:265, 21:442, 21:557, 02:443, 1321
<code>\abovedisplayshortskip</code> ..	38:531, 02:413
<code>\abovedisplayskip</code>	38:524, 38:526, 38:528, 38:529, 38:530, 38:531, 02:412, 1307
<code>\abspage</code>	809
<code>abspage</code>	36:232, 809
<code>\accent</code>	21:91, 21:413, 21:443, 21:500, 21:785, 1337
<code>\ActivateGenericHook</code>	08:2751, 08:2753, 08:2758, 08:2760, 201
<code>\active</code>	01:103, 37:503, 37:504, 37:505, 37:506, 37:513, 37:514, 37:652, 37:659, 37:660, 37:669, 37:675, 37:683, 37:689, 37:698, 37:740, 38:254, 38:269, 01:314, 02:10, 02:11, 50:1248, 50:1249, 50:1250, 50:1331, 50:1334, 50:1337, 50:1379, 50:1380, 50:1381, 50:1463, 50:1466, 50:1469, 50:1493, 50:1494,

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- 50:1495, 50:1553, 50:1556, 50:1559,
54:783, 01:48, 02:457, 02:458,
02:466, 02:467, 02:479, 02:480, 02:484
- \acute 30:527
- add commands:
- add_to_callback 04:817
- \add_to_callback 45
- \addcontentsline
 44:70, 44:80, 44:159, 45:16, 1306
- \AddEverypageHook 1131
- \Adding 1282
- \addpenalty 18:285,
 39:124, 39:201, 39:206, 44:50,
 54:336, 54:1428, 54:1577, 54:1738, 442
- \AddThispageHook 1131
- \addtocontents
 . 44:164, 44:171, 44:177, 44:188, 1306
- \addtocounter 521
- \addtocounter 22:6, 22:20, 1294
- \AddToHook 08:2763,
 08:2913, 26:153, 37:42, 37:43, 37:44,
 37:107, 37:108, 37:109, 37:396,
 37:397, 37:398, 37:399, 47:72,
 50:1128, 50:1129, 50:1130, 52:561,
 52:563, 52:573, 52:574, 52:575,
 52:576, 53:190, 53:473, 53:496, 1130
- \AddToHookNext
 . 08:2774, 08:2911, 52:562, 53:497, 285
- \AddToHookNextWithArguments 08:2774, 204
- \AddToHookWithArguments ... 08:2763, 218
- \addtolength 531
- \addtolength 23:16, 38:526, 38:528
- \AddToNoCaseChangeList 57:571, 1349
- \addtoversion 27:20, 27:139
- \advspace 18:234, 37:429, 39:124, 39:202,
 39:203, 39:207, 39:255, 44:50, 1293
- \adjdemerits 02:336
- \AE 21:261, 21:418, 21:548, 21:790,
 21:1149, 21:1459, 21:1463, 57:655
- \ae 21:266, 21:421, 21:558, 21:794,
 21:1155, 21:1461, 21:1465, 57:655
- \afterassignment 21:232,
 21:240, 24:391, 38:199, 06:268,
 06:274, 06:317, 02:501, 02:504, 93
- \AfterEndEnvironment 37:392, 223
- \aftergroup 24:92, 24:410,
 24:428, 24:433, 26:203, 26:269,
 28:115, 28:122, 28:130, 28:420,
 32:64, 37:656, 37:673, 37:687,
 37:702, 38:445, 38:446, 39:149,
 39:154, 40:187, 54:807, 54:808,
 54:882, 54:883, 54:940, 54:941, 870
- \AfterLastShipout 52:573
- \afterpreamble 455
- \aleph 30:319
- \allocationnumber 16:81, 04:52, 04:53,
 04:54, 04:91, 04:213, 41:4, 41:9,
 53:34, 02:37, 02:57, 02:69, 02:71,
 57:58, 57:59, 57:60, 02:143, 02:144,
 02:145, 02:195, 02:196, 02:237,
 02:238, 02:239, 02:252, 02:253,
 02:254, 02:271, 02:277, 02:283,
 02:284, 02:297, 02:298, 02:299, 421
- \allowbreak 38:40,
 06:877, 06:878, 06:897, 06:899, 02:508
- \Alph 521
- \Alph 22:193, 1129
- \alph 521
- \alpha 22:192
- \alpha 30:279
- \amalg 30:390
- \AmSfont 619
- \and 958
- \and 44:14, 44:27
- \angle 30:348
- \approx 30:433
- \arabic 521
- \arabic 22:132,
 22:140, 22:189, 43:46, 53:350, 1129
- \arccos 38:13
- \arcsin 38:10
- \arctan 38:16
- \arg 38:26
- array (env.) 41:168
- \array 41:168
- \arraycolsep 38:373,
 38:374, 38:536, 38:537, 41:258, 41:338
- \arrayrulewidth
 ... 41:324, 41:338, 41:346, 41:347,
 41:359, 41:363, 41:366, 41:376, 41:378
- \arraystretch 41:186, 41:187, 41:342
- \Arrowvert 30:580
- \arrowvert 30:578
- \asciispace 37:576, 37:578, 37:581,
 37:595, 37:606, 37:607, 37:620, 37:621
- \AssignSocketPlug
 ... 10:182, 10:208, 35:122, 54:680,
 55:45, 55:58, 55:62, 55:63, 55:66,
 55:67, 55:83, 55:100, 55:107, 55:108,
 55:123, 55:125, 55:127, 55:129,
 55:132, 55:137, 55:139, 55:141,
 55:143, 55:151, 55:154, 55:157, 337
- \AssignTemplateKeys
 11:389, 11:394, 11:1200, 353
- \ast 30:243, 30:406
- \asympt 30:460
- \AtBeginDocument 20:126,
 20:181, 25:3257, 47:54, 50:1119, 224

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
11=ltemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx,
16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
56=lthyphen.dtx, 57=ltfinal.dtx

- `\atbegindocumenthook` 455
`\AtBeginDvi` ... 53:412, 53:448, 54:82, 1131
`\AtBeginEnvironment` 37:392, 223
`\AtBeginShipout` 53:453, 53:496, 1131
`\AtBeginShipoutAddToBox` ... 53:501, 1130
`\AtBeginShipoutAddToBoxForeground` .
 53:501, 1130
`\AtBeginShipoutBox` 53:494, 1130
`\AtBeginShipoutDiscard` 53:500, 1131
`\AtBeginShipoutFirst` 53:456, 53:498, 1131
`\AtBeginShipoutInit` 53:495, 1131
`\AtBeginShipoutNext` . 53:454, 53:496, 1131
`\AtBeginShipoutOriginalShipout`
 53:509, 1130
`\AtBeginShipoutUpperLeft` ... 53:501, 1130
`\AtBeginShipoutUpperLeftForeground` .
 53:501, 1130
`\AtEndAfterFileList` 52:575
`\AtEndDocument` 37:144, 50:1119, 52:579, 225
`\AtEndDvi` 53:463, 53:468, 1131
`\AtEndEnvironment` 37:392, 223
`\AtEndOfClass` 38:461, 50:1119, 1102
`\AtEndOfPackage` 50:606, 50:625,
 50:723, 50:734, 50:1119, 51:88, 1102
`\AtEndPreamble` 224
`\AtNextShipout` 1131
`\atopwithdelims` 38:57, 38:58, 38:59
`\attribute` 04:79, 42
`\attributedef` 04:79, 04:223
`\attributezero` 04:223
`\AtVeryEndDocument` 52:574
`\AtVeryVeryEnd` 52:576
`\author` 958
`\author` 44:8, 44:24, 44:32
- B**
- `\b` 21:253,
 21:409, 21:496, 21:781, 21:1270, 1309
`\backslash` 30:262, 30:601
`\bar` 30:531
`\baselineskip`
 ... 26:187, 26:188, 26:189, 26:191,
 26:192, 30:521, 38:171, 38:172,
 38:191, 38:197, 38:201, 40:396,
 40:415, 40:601, 41:198, 42:136,
 42:314, 42:373, 53:222, 53:273,
 54:240, 54:271, 54:830, 54:852,
 54:900, 54:915, 54:959, 54:974,
 02:409, 02:436, 02:499, 02:535, 1318
`\baselinestretch` 24:382, 26:121, 26:122,
 26:167, 26:168, 26:185, 26:246, 1280
`\baselinestretch` 1300
- `\batchmode` 20:691,
 20:722, 20:723, 27:106, 05:89,
 05:95, 05:105, 57:673, 57:694, 471
`\BCPdata` 57:560, 57:587, 57:591
`\BeforeBeginEnvironment` 37:392, 223
`\BeforeClearDocument` 52:577
`\begin` 09:83,
 14:246, 14:248, 20:345, 20:397,
 26:7, 30:4, 31:4, 37:242, 37:243,
 37:313, 37:335, 37:360, 37:376,
 37:384, 38:466, 38:478, 44:14, 44:17,
 50:744, 06:709, 51:256, 53:394,
 56:3, 07:1460, 07:1570, 07:2844, 1284
`\begingroup` 1323
`\belowdisplayskip` .. 38:530, 02:415
`\belowdisplayskip` 38:529, 02:414
`\beta` 30:280
`\bezier` 922
`\bezier` 42:681,
 42:682, 42:804, 42:805, 42:820, 42:822
`\bf` 1288
`\bfdefault` 29:16, 29:262, 29:268, 29:269,
 29:270, 29:271, 29:311, 29:312,
 29:313, 29:314, 29:323, 29:359,
 29:383, 29:402, 29:443, 29:477,
 30:104, 30:115, 30:117, 30:125, 622
`\bfseries` 29:14, 29:15, 29:254,
 29:255, 29:303, 29:308, 29:309,
 29:350, 29:353, 29:354, 29:379,
 29:381, 29:382, 29:475, 29:476,
 32:19, 35:17, 35:34, 35:51, 43:53,
 43:55, 43:60, 43:62, 47:40, 53:395, 696
`bfseries` 29:421
`bfseries/defaults` 29:421
`\bgroup` 02:450, 1281
`\bibcite` 47:7, 47:9, 47:10, 1313
`\bibdata` 47:45, 47:49
`\bibitem` 47:3
`\bibliography` 998
`\bibliography` 47:47
`\bibliographystyle` 998
`\bibliographystyle` 47:52
`\bibstyle` 47:45, 47:57
`\Big` 30:634, 30:637,
 30:646, 30:648, 38:44, 38:45, 38:46
`\big` 30:635, 30:647, 38:41
`\bigbreak` 06:879, 06:900, 02:515
`\bigcap` 30:356
`\bigcirc` 30:403
`\bigcup` 30:357
`\Bigg` ... 30:641, 30:650, 38:50, 38:51, 38:52
`\bigg` ... 30:639, 30:649, 38:47, 38:48, 38:49
`\Biggl` 38:50
`\biggl` 38:47

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpara.dtx,
16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
56=lthyphen.dtx, 57=ltfinal.dtx

<code>\Biggm</code>	38:51	08:2466, 08:2934, 53:66, 53:112,
<code>\biggm</code>	38:48	53:376, 07:748, 07:765, 07:956, 08:275
<code>\Biggr</code>	38:52	<code>\bool_lazy_and_p:nn</code> .. 08:2469, 07:115
<code>\biggr</code>	38:49	<code>\bool_lazy_any:nTF</code> ... 07:111, 07:486
<code>\Bigl</code>	38:44	<code>\bool_lazy_or:nnTF</code>
<code>\bigl</code>	38:41 08:1867, 07:76, 07:1419,
<code>\Bigm</code>	38:45	07:1428, 07:2617, 07:2639, 08:887
<code>\bigm</code>	38:42	<code>\bool_new:N</code> 10:6, 11:21, 11:22,
<code>\bigodot</code>	30:364	36:28, 48:325, 51:57, 53:6, 53:188,
<code>\bigoplus</code>	30:363	53:203, 07:18, 07:20, 07:22, 07:33,
<code>\bigotimes</code>	30:362	07:34, 07:36, 07:37, 07:39, 07:42,
<code>\Bigr</code>	38:46	07:45, 07:46, 07:47, 07:48, 08:6, 08:24
<code>\bigr</code>	38:43	<code>\bool_set_false:N</code> 08:1906, 08:2006,
<code>\bigskip</code>	18:447, 02:520	11:277, 11:663, 11:744, 51:87,
<code>\bigskipamount</code>		07:60, 07:86, 07:235, 07:257, 55:41,
.....	18:449, 18:450, 45:391, 02:519	07:302, 07:452, 07:453, 07:454,
<code>\bigscup</code>	30:367	07:455, 07:456, 07:457, 07:478,
<code>\bigtriangledown</code>	30:372, 30:373	07:540, 07:570, 07:595, 07:609,
<code>\bigtriangleup</code>	30:371, 30:374	07:631, 07:632, 07:633, 07:648,
<code>\biguplus</code>	30:355	07:715, 07:758, 07:772, 07:773,
<code>\bigvee</code>	30:353	07:795, 07:796, 07:797, 07:799,
<code>\bigwedge</code>	30:354	07:803, 07:813, 07:817, 07:973,
<code>\binoppenalty</code>	02:323	07:974, 07:975, 07:976, 07:1589, 07:2071
<code>\bmod</code>	38:35	<code>\bool_set_true:N</code> 08:1903,
<code>\boldmath</code>	19:14, 29:614	08:2003, 11:248, 11:290, 11:622,
bool commands:		11:753, 51:84, 07:65, 07:236, 07:258,
<code>\bool_gset_false:N</code>	10:16,	55:43, 07:291, 07:451, 07:476,
36:31, 48:334, 53:15, 53:81, 53:90, 08:15		07:547, 07:554, 07:555, 07:569,
<code>\bool_gset_true:N</code> ...	10:11, 36:33,	07:761, 07:762, 07:812, 07:833,
48:329, 53:10, 53:120, 53:344, 08:10		07:834, 07:840, 07:841, 07:847,
<code>\bool_if:NTF</code>	08:1914,	07:848, 07:855, 07:856, 07:857,
08:1923, 08:2014, 08:2023, 09:179,		07:992, 07:1010, 07:1594, 07:2076
09:183, 09:195, 09:254, 09:258,		<code>\bool_while_do:nn</code> ... 08:1593, 08:1678
09:270, 09:397, 10:22, 10:24, 11:222,		<code>\c_false_bool</code> 09:158, 09:168,
11:228, 11:665, 11:768, 11:817,		09:387, 07:2463, 07:2766, 07:3069,
11:834, 11:856, 36:71, 48:340,		07:3237, 08:397, 08:403, 08:411, 196
51:24, 51:219, 53:21, 53:88, 53:365,		<code>\l_tmpa_bool</code>
07:124, 07:133, 07:145, 07:146,		354
07:160, 07:172, 07:176, 07:185,		<code>\c_true_bool</code>
07:187, 07:197, 07:206, 07:207,		09:163, 07:2464, 07:2768, 07:3070,
07:210, 07:215, 07:217, 07:321,		07:3235, 08:404, 08:412, 08:414, 196
07:466, 07:475, 07:477, 07:532,		bool internal commands:
07:563, 07:578, 07:627, 07:664,		<code>\g__mark_debug_bool</code>
07:709, 07:719, 07:752, 07:760,	 48:325, 48:329, 48:334, 48:340
07:769, 07:777, 07:782, 07:807,		<code>\BooleanFalse</code> .. 07:2064, 07:2456, 07:3069
07:826, 07:908, 07:954, 07:955,		<code>\BooleanTrue</code> ... 07:2063, 07:2455, 07:3069
07:1025, 07:1038, 07:1055, 07:1075,		<code>\bordermatrix</code>
07:1741, 07:2185, 07:2196, 07:2462,		38:185
07:2794, 07:2805, 08:21, 08:340, 196		<code>\bot</code>
<code>\bool_if_exist:NTF</code>	36:27	30:332
<code>\bool_lazy_all:nTF</code>	07:960	<code>\botfigrule</code>
<code>\bool_lazy_and:nnTF</code>		54:1016, 54:2949
.....	08:1916, 08:2016,	<code>\botmark</code> 49:119, 54:925, 54:984, 1003
		<code>\bottomfraction</code>
		45:275, 54:2918
		<code>\bowtie</code>
		30:493
		<code>\Box</code>
		29:727
		<code>\box</code>
		413

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpara.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

box commands:

`\box_dp:N` 53:194
`\box_gc clear:N` 16:89
`\box_gset_to_last:N` 16:16, 16:49, 16:117
`\box_ht:N` 53:193, 53:302, 1135
`\box_if_empty:NTF` . 48:87, 53:78, 53:97
`\box_if_horizontal:NTF` 53:234, 53:286
`\box_if_vertical:NTF`
. 48:82, 48:404, 48:436, 53:208, 53:259
`\box_move_up:nn` 53:248, 53:302
`\box_new:N` 16:84, 48:62,
48:63, 53:23, 53:25, 53:187, 53:204
`\box_set_dp:Nn` ... 53:221, 53:230,
53:247, 53:272, 53:283, 53:301, 53:332
`\box_set_eq:NN` . 53:149, 53:179, 53:184
`\box_set_eq_drop:NN` 53:93
`\box_set_ht:Nn` ... 53:220, 53:229,
53:246, 53:271, 53:282, 53:300, 53:331
`\box_set_to_last:N` 48:79
`\box_set_wd:Nn`
..... 53:219, 53:245, 53:270, 53:299
`\box_use:N` 53:126,
53:225, 53:250, 53:278, 53:303, 53:333
`\box_use_drop:N` 16:86
`\box_wd:N` 53:195, 53:296, 53:304
box internal commands:
`\l_mark_box` .. 48:62, 48:75, 48:79,
48:82, 48:83, 48:86, 48:87, 48:97, 1016
`\l_mark_ii_box` .. 48:62, 48:86, 48:96
`\boxmaxdepth` 42:474,
42:502, 42:532, 42:610, 42:627,
54:577, 54:695, 54:998, 54:1038, 02:395
`\brace` 38:59
`\braceld`
30:564, 30:568, 30:569, 30:571, 30:573
`\bracelu` 30:566, 30:570, 30:572
`\bracerd` 30:565, 30:570, 30:572
`\braceru` 30:567, 30:569, 30:573
`\bracevert` 30:619
`\brack` 38:58
`\break` 18:116,
06:880, 06:901, 02:508, 02:513, 1314
`\breve` 30:532
`\brokenpenalty` 24:764, 02:328
`build/column/after (hook)` 54:684
`build/column/before (hook)` 54:684
`build/column/footnotes (socket)` .. 54:681
`build/column/outputbox (socket)` .. 54:631
`build/page/after (hook)` 54:682
`build/page/before (hook)` 54:682
`build/page/reset (hook)` 54:683
`\buildrel` 30:480, 38:162
`\bullet` 30:392

C

`\c` 11:385, 21:254, 21:355, 21:357,
21:359, 21:361, 21:363, 21:365,
21:367, 21:369, 21:371, 21:392,
21:394, 21:412, 21:480, 21:499,
21:627, 21:629, 21:654, 21:656,
21:669, 21:695, 21:722, 21:725,
21:726, 21:727, 21:728, 21:729,
21:730, 21:731, 21:732, 21:733,
21:784, 21:1272, 21:1286, 21:1312,
21:1369, 21:1370, 21:1389, 21:1390,
21:1393, 21:1394, 21:1399, 21:1400,
21:1411, 21:1412, 21:1419, 21:1420,
21:1423, 21:1424, 33:131, 33:160, 1323
`\cal` 29:777
call commands:
`call_callback` 04:798
`\call_callback` 46
callback commands:
`callback_descriptions` 04:983
`callback_register` 04:686
`\callback_descriptions` 46
`\cap` 30:383
`\capitalacute` 21:861, 33:124,
33:125, 33:157, 33:654, 33:904, 33:1270
`\capitalbreve` 21:868, 33:126,
33:127, 33:158, 33:655, 33:911, 33:1277
`\capitalcaron` 21:867, 33:128,
33:129, 33:159, 33:656, 33:910, 33:1276
`\capitalcedilla` 21:854, 33:130,
33:131, 33:160, 33:657, 33:901, 33:1280
`\capitalcircumflex` ... 21:862, 33:132,
33:133, 33:161, 33:658, 33:905, 33:1271
`\capitaldieresis` 21:864, 33:134,
33:135, 33:162, 33:659, 33:907, 33:1273
`\capitaldotaccent` ... 21:870, 33:136,
33:137, 33:163, 33:660, 33:913, 33:1279
`\capitalgrave` 21:860, 33:138,
33:139, 33:164, 33:661, 33:903, 33:1269
`\capitalhungarumlaut` . 21:865, 33:140,
33:141, 33:165, 33:662, 33:908, 33:1274
`\capitalmacron` 21:869, 33:142,
33:143, 33:166, 33:663, 33:912, 33:1278
`\capitalnewtie`
... 21:874, 33:154, 33:155, 33:167,
33:664, 33:978, 33:979, 33:1285, 760
`\capitalogonek` 21:857, 33:144,
33:145, 33:168, 33:665, 33:902, 33:1281
`\capitalring` 21:866, 33:146,
33:147, 33:169, 33:666, 33:909, 33:1275
`\capitaltie` 21:872, 33:148, 33:149,
33:170, 33:667, 33:974, 33:975, 33:1283
`\capitaltilde` 21:863, 33:150,
33:151, 33:171, 33:668, 33:906, 33:1272

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
11=ltemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx,
16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
46=ltidglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
56=ltthyphen.dtx, 57=ltfinal.dtx

- \caption [45:4](#), [310](#)
- \cases [38:166](#), [38:167](#), [38:177](#), [38:179](#)
- \CaseSwitch [57:571](#), [1349](#)
- \catcode [826](#)
- \catcodetable [04:89](#), [04:109](#), [42](#)
- \catcodetable@atletter [43](#)
- \catcodetable@initex [43](#)
- \catcodetable@latex [43](#)
- \catcodetable@string [43](#)
- \catcoding [1284](#)
- \cdot [30:405](#)
- \cdotp [30:513](#), [30:519](#)
- \cdots [30:519](#)
- center (env.) [37:444](#)
- \center [37:444](#)
- \centering [37:444](#), [37:449](#),
[37:450](#), [37:468](#), [37:470](#), [37:485](#), [37:487](#)
- \centerline [40:614](#)
- \changes [22:38](#), [29:668](#), [35:158](#), [1324](#)
- \chaptermark [1009](#)
- \char [21:411](#), [21:414](#), [21:450](#), [21:453](#),
[21:464](#), [21:471](#), [21:498](#), [21:502](#),
[21:507](#), [21:510](#), [21:512](#), [21:514](#),
[21:755](#), [21:783](#), [21:786](#), [21:819](#),
[21:826](#), [21:833](#), [21:856](#), [21:859](#),
[21:888](#), [21:918](#), [21:1028](#), [21:1063](#),
[21:1187](#), [21:1189](#), [21:1191](#), [21:1238](#),
[29:627](#), [29:634](#), [33:595](#), [33:602](#),
[33:604](#), [33:724](#), [37:576](#), [37:743](#),
[38:251](#), [42:232](#), [42:282](#), [42:296](#),
[42:304](#), [42:307](#), [42:448](#), [42:562](#),
[42:567](#), [42:575](#), [42:579](#), [42:615](#),
[42:616](#), [42:618](#), [42:631](#), [42:632](#),
[42:635](#), [42:662](#), [06:842](#), [06:857](#), [1317](#)
- char commands:
- \char_generate:nn [05:145](#), [07:1893](#), [07:2205](#)
- \char_set_catcode_active:N [07:2487](#)
- \char_set_catcode_active:n
 [07:2183](#), [07:2184](#), [07:2664](#)
- \char_set_catcode_escape:N [09:460](#)
- \char_set_catcode_group_begin:N
 [09:461](#)
- \char_set_catcode_group_end:N [09:462](#)
- \char_set_catcode_other:N
 [07:1604](#), [07:1610](#),
 [07:1643](#), [07:1689](#), [07:1728](#), [07:1792](#),
 [07:1814](#), [07:1816](#), [07:2181](#), [07:2194](#)
- \char_set_catcode_other:n
 [07:1606](#), [07:2186](#), [07:2197](#)
- \char_set_catcode_parameter:N [09:463](#)
- \char_set_catcode_parameter:n
 [07:2187](#), [07:2198](#)
- \char_set_catcode_space:n [07:1818](#)
- \char_set_lccode:nn [07:2492](#), [07:2674](#)
- \char_value_catcode:n [09:392](#)
- \chardef [12:2](#), [04:22](#), [04:26](#), [04:38](#),
[04:47](#), [04:48](#), [04:89](#), [20:52](#), [20:122](#),
[04:157](#), [21:36](#), [04:224](#), [24:15](#), [41:4](#),
[41:9](#), [02:10](#), [02:16](#), [50:1254](#), [02:17](#),
[50:1385](#), [02:18](#), [02:19](#), [50:1502](#),
[02:20](#), [02:58](#), [02:64](#), [02:66](#), [57:42](#),
[57:44](#), [57:48](#), [57:67](#), [02:73](#), [57:171](#),
[57:172](#), [57:173](#), [57:174](#), [57:175](#),
[57:176](#), [57:177](#), [02:79](#), [02:82](#), [02:84](#),
[02:94](#), [02:96](#), [02:97](#), [02:98](#), [02:99](#),
[02:108](#), [02:114](#), [02:115](#), [02:128](#),
[02:130](#), [02:194](#), [01:48](#), [02:253](#),
[02:257](#), [02:259](#), [01:54](#), [02:283](#), [01:55](#),
[02:298](#), [02:529](#), [02:530](#), [02:531](#), [1332](#)
- \charsubdef [57:353](#)
- \charzero [04:224](#)
- \check [30:533](#)
- \CheckCommand [06:187](#), [1301](#)
- \CheckEncodingSubset
 [24:204](#), [33:16](#), [33:73](#), [33:74](#), [33:75](#),
 [33:120](#), [33:122](#), [33:316](#), [33:585](#),
 [33:794](#), [33:843](#), [33:899](#), [33:900](#),
 [33:968](#), [33:1085](#), [33:1088](#), [33:1102](#), [756](#)
- \chi [30:299](#)
- \choose [38:57](#)
- \circ [30:402](#)
- \circle [42:450](#), [42:604](#), [42:806](#), [42:823](#), [1284](#)
- \citation [47:11](#), [47:39](#), [47:67](#), [47:84](#)
- \cite [998](#)
- \cite [47:12](#), [999](#)
- \clap [40:618](#), [1338](#)
- class/.../after [1101](#)
- class/.../before [1101](#)
- class/after [1101](#)
- class/before [1101](#)
- \ClassError [14:84](#)
- \ClassInfo [14:84](#)
- \ClassNote [14:136](#), [1345](#)
- \ClassNoteNoLine [14:136](#)
- \ClassWarning [14:84](#)
- \ClassWarningNoLine [14:84](#)
- \cleaders [30:559](#), [30:562](#), [02:549](#)
- \cleardoublepage [54:136](#)
- \ClearHookNext [08:2785](#), [204](#)
- \ClearHookRule [08:2822](#), [08:2927](#), [208](#)
- \clearpage [20:297](#), [20:324](#),
[20:328](#), [20:359](#), [20:382](#), [20:385](#),
[20:406](#), [20:424](#), [37:17](#), [37:83](#), [37:147](#),
[37:240](#), [54:123](#), [54:136](#), [54:141](#),
[54:198](#), [54:405](#), [54:408](#), [54:412](#),
[54:453](#), [54:459](#), [54:2770](#), [54:2787](#), [225](#)
- \cline [41:367](#), [1356](#)

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- clist commands:
- \clist_clear:N 51:63
 - \clist_gclear:N 08:1592, 08:1677
 - \clist_gput_left:Nn . 08:1496, 08:1531
 - \clist_gput_right:Nn 08:1498, 08:1535
 - \clist_if_empty:NTF
.... 08:1933, 08:2034, 11:541, 11:570
 - \clist_if_exist:NTF 08:121
 - \clist_if_in:NnTF . 11:266, 11:577,
51:51, 51:52, 51:130, 51:153, 51:155
 - \clist_map_function:nN 36:59
 - \clist_map_inline:Nn
..... 11:543, 51:111, 51:142, 51:207
 - \clist_map_inline:nn
..... 51:242, 08:918, 08:927
 - \clist_map_variable:NNn 51:85
 - \clist_new:N 11:32,
51:54, 51:55, 51:56, 08:123, 08:140
 - \clist_put_right:Nn . 51:33, 51:53,
51:124, 51:131, 51:138, 51:163, 51:172
 - \clist_remove_all:Nn
..... 11:579, 51:35, 51:137, 51:164
 - \clist_set:Nn 11:531
 - \clist_use:Nn 08:1935, 08:2036
 - \l_tmpa_clist 354
 - \clubpenalty
20:8, 20:25, 20:95, 20:152, 24:762,
39:128, 39:225, 39:227, 44:100,
44:106, 44:130, 44:135, 02:325, 420
 - \clubsuit 30:342
- cmd internal commands:
- __cmd_add_arg:n 07:1739,
07:1808, 07:1840, 07:1845, 07:1846,
07:1880, 07:1968, 07:1975, 07:2050,
07:2063, 07:2064, 07:2138, 07:2211,
07:2218, 07:2262, 07:2262, 07:2267, 157
 - __cmd_add_arg_spec:n ... 07:593,
07:607, 07:671, 07:746, 07:746, 07:789
 - __cmd_add_arg_spec_mandatory:n .
07:639, 07:649, 07:655, 07:746, 07:775
 - __cmd_add_default: 07:869,
07:907, 07:924, 07:986, 07:989,
07:996, 07:1006, 07:1073, 07:1082, 136
 - __cmd_add_default:n
..... 07:876, 07:916, 07:932,
07:986, 07:986, 07:1003, 07:1017, 136
 - __cmd_add_default_E:nn
.... 07:884, 07:986, 07:1001, 07:1051
 - __cmd_add_expandable_grabber:nn
..... 07:1030, 07:1043, 07:1054,
07:1074, 07:1084, 07:1091, 07:1091
 - __cmd_add_expandable_type_+ :w ..
..... 07:1008
 - __cmd_add_expandable_type_D:w ..
..... 07:1013, 07:1013
 - __cmd_add_expandable_type_D_-
aux:NN ... 07:1013, 07:1019, 07:1036
 - __cmd_add_expandable_type_D_-
aux:NNN .. 07:1013, 07:1020, 07:1023
 - __cmd_add_expandable_type_D_-
aux:NNNn
.. 07:1013, 07:1014, 07:1015, 07:1079
 - __cmd_add_expandable_type_E:w ..
..... 07:1049, 07:1049
 - __cmd_add_expandable_type_E_-
aux:n 07:1049, 07:1053, 07:1065
 - __cmd_add_expandable_type_m:w ..
..... 07:1071, 07:1071
 - __cmd_add_expandable_type_R:w ..
..... 07:1078, 07:1078
 - __cmd_add_expandable_type_t:w ..
..... 07:1080, 07:1080
 - __cmd_add_grabber:N 07:870,
07:877, 07:890, 07:909, 07:917,
07:925, 07:933, 07:947, 07:947, 139
 - __cmd_add_type_! :w 07:837
 - __cmd_add_type_+ :w 07:830
 - __cmd_add_type_= :w 07:852
 - __cmd_add_type_> :w 07:844
 - __cmd_add_type_b:w .. 07:862, 07:862
 - __cmd_add_type_b_or_c:N
..... 07:862, 07:863, 07:865, 07:866
 - __cmd_add_type_c:w .. 07:862, 07:864
 - __cmd_add_type_D:w .. 07:873, 07:873
 - __cmd_add_type_E:w .. 07:881, 07:881
 - __cmd_add_type_m:w .. 07:905, 07:905
 - __cmd_add_type_R:w .. 07:913, 07:913
 - __cmd_add_type_t:w .. 07:921, 07:921
 - __cmd_add_type_v:w .. 07:929, 07:929
 - __cmd_all_m_check:n
..... 07:99, 07:99, 07:119
 - __cmd_all_m_check_aux:n
..... 07:99, 07:100, 07:101
 - __cmd_allowed_token_check:N ...
..... 07:591,
07:603, 07:624, 07:645, 07:688, 07:688
 - \l__cmd_arg_spec_tl
..... 07:12, 07:93, 07:450,
07:538, 07:571, 07:625, 07:763, 129
 - __cmd_arg_to_keyvalue:nn
..... 07:859, 07:2546, 07:2546
 - __cmd_arg_to_keyvalue_auxi:nnn .
..... 07:2546, 07:2561, 07:2563
 - __cmd_arg_to_keyvalue_auxii:Nnnn
..... 07:2546, 07:2566, 07:2569
 - __cmd_arg_to_keyvalue_auxiii:nnn
..... 07:2546, 07:2572, 07:2575

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=ltparams.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- `__cmd_arg_to_keyvalue_auxiv:Nnnn` 07:2546, 07:2578, 07:2581
- `__cmd_arg_to_keyvalue_auxv:nn` 07:2546, 07:2567, 07:2573, 07:2579, 07:2585, 07:2587
- `__cmd_arg_to_keyvalue_braces:nnn` 07:2546, 07:2548, 07:2551
- `__cmd_arg_to_keyvalue_loop:w` 07:2546, 07:2589, 07:2592, 07:2604, 07:2606, 07:2621, 07:2642
- `__cmd_arg_to_keyvalue_loop_-group:n` 07:2546, 07:2598, 07:2603
- `__cmd_arg_to_keyvalue_loop_N_-type:N` 07:2546, 07:2595, 07:2607
- `__cmd_arg_to_keyvalue_loop_-space:w` 07:2546, 07:2599, 07:2605
- `__cmd_arg_to_keyvalue_math:w` 07:2546, 07:2620, 07:2624, 07:2643, 07:2646, 07:2648
- `__cmd_arg_to_keyvalue_math_-group:n` 07:2546, 07:2630, 07:2645
- `__cmd_arg_to_keyvalue_math_N_-type:N` 07:2546, 07:2627, 07:2635
- `__cmd_arg_to_keyvalue_math_-space:w` 07:2546, 07:2631, 07:2647
- `__cmd_arg_to_keyvalue_set_-default:nn` 07:2546, 07:2610, 07:2638, 07:2649, 182
- `__cmd_arg_to_keyvalue_set_-keyvalue:nn` 07:2546, 07:2614, 07:2651, 182
- `\l__cmd_args_i_tl` 07:14, 07:330, 07:336, 07:341, 07:342, 07:344, 113
- `\l__cmd_args_ii_tl` 07:15, 07:340, 07:342, 07:344, 07:382, 07:387, 07:394, 113
- `__cmd_args_process:` 07:320, 07:380, 07:380, 137
- `__cmd_args_process_aux:n` 07:380, 07:393, 07:397
- `__cmd_args_process_loop:nn` 07:380, 07:386, 07:389
- `\l__cmd_args_tl` 07:13, 07:283, 07:308, 07:325, 07:330, 07:336, 07:355, 07:384, 07:387, 07:400, 07:401, 07:1572, 07:1573, 07:1578, 07:1581, 07:1946, 07:1980, 07:2264, 113
- `__cmd_bad_arg_spec:wn` 07:508, 07:513, 07:522, 07:531, 07:577, 07:590, 07:600, 07:601, 07:606, 07:617, 07:623, 07:644, 07:740, 07:740
- `__cmd_bad_def:wn` 07:464, 07:472, 07:501, 07:536, 07:567, 07:582, 07:668, 07:677, 07:685, 07:704, 07:713, 07:725, 07:740, 07:745, 07:756, 07:786, 146
- `__cmd_bool_reverse:N` 07:2460, 07:2460, 07:3268
- `__cmd_break_point:n` 07:96, 07:98, 07:98, 07:740, 07:745, 07:1141, 07:1148, 07:1359
- `__cmd_cant_copy:nwn` 07:1138, 07:1148, 07:1148, 07:1264, 07:1327, 07:1357
- `__cmd_check_definable:nNTF` 07:2661, 07:2661, 07:3073, 07:3086, 07:3099, 07:3104, 07:3192, 07:3205, 07:3218, 07:3226
- `__cmd_check_definable_aux:nN` 07:2661, 07:2662, 07:2665, 185
- `__cmd_check_end:n` 07:1323, 07:1325, 07:1331
- `__cmd_check_end:Nn` 07:1310, 07:1311, 07:1323, 07:1323, 07:1469
- `__cmd_check_end:w` 07:1323, 07:1333, 07:1336
- `__cmd_chk_if_free_cs:N` 07:3273, 07:3274
- `__cmd_cmd_if_xparse:NTF` 1344
- `__cmd_cmd_if_xparse_aux:N` 07:2728
- `__cmd_cmd_type_cases:NnnnnnTF` 07:1132, 07:1351, 07:2728, 07:2728, 07:2751, 145
- `__cmd_cmd_type_cases:NnnnnTF` 1348
- `__cmd_copy:NN` 07:1125, 07:1127, 07:1127, 07:1162, 07:1177, 07:1223, 07:1341, 146
- `__cmd_copy_command:nnNN` 07:1133, 07:1163, 07:1163, 146
- `__cmd_copy_command:NnNNnnnn` 07:1163, 07:1168, 07:1170, 146
- `__cmd_copy_environment:nnNN` 07:1136, 07:1297, 07:1297
- `__cmd_copy_environment:Nnnnnnn` 07:1297, 07:1303, 07:1306
- `__cmd_copy_environment_end:nnNN` 07:1137, 07:1308, 07:1308
- `__cmd_copy_environment_end_-aux:nnNN` 07:1308, 07:1312, 07:1315
- `__cmd_copy_expandable:nnN` 07:1242, 07:1247, 07:1250, 07:1277, 07:1287, 07:1293, 07:1295
- `__cmd_copy_expandable:nnNN` 07:1134, 07:1176, 07:1180, 07:1182, 07:1195, 07:1197, 07:1208
- `__cmd_copy_expandable:NnNNNNnnn` 07:1176, 07:1192, 07:1205, 07:1225, 147

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

`__cmd_copy_expandable_signature:NnNNNNnnn__cmd_declare_env_internal:nmmm`
 07:1188, 07:1203, 07:1242, 07:1242, 147
`__cmd_copy_grabber_(type):w` 148
`__cmd_copy_grabber_D:w`
 07:1267, 07:1267, 07:1280, 07:1281
`__cmd_copy_grabber_D_alt:w`
 07:1267, 07:1279, 07:1282
`__cmd_copy_grabber_E:w`
 07:1267, 07:1283, 07:1289
`__cmd_copy_grabber_E_long:w`
 07:1267, 07:1289
`__cmd_copy_grabber_m:w`
 07:1267, 07:1295, 07:1296
`__cmd_copy_grabber_m_long:w`
 07:1267, 07:1296
`__cmd_copy_grabber_R:w`
 07:1267, 07:1281
`__cmd_copy_grabber_R_alt:w`
 07:1267, 07:1282
`__cmd_copy_grabber_t:w`
 07:1267, 07:1290
`__cmd_copy_optimized:nnNN`
 07:1135, 07:1211, 07:1211
`__cmd_copy_parse_grabber:w`
 07:1242, 07:1254, 07:1258, 148
`\l__cmd_current_arg_int`
 07:16, 07:27, 07:149,
 07:157, 07:186, 07:193, 07:273,
 07:354, 07:358, 07:363, 07:364,
 07:448, 07:460, 07:483, 07:539,
 07:585, 07:610, 07:793, 07:794,
 07:965, 07:991, 07:998, 07:1245,
 07:1253, 07:1271, 07:1275, 07:1276,
 07:1286, 07:1478, 07:1544, 07:1553, 149
`__cmd_declare_cmd:Nnn`
 07:58, 07:58,
 07:3081, 07:3089, 07:3100, 07:3105
`__cmd_declare_cmd_aux:Nnn`
 07:58, 07:61, 07:66, 07:68
`__cmd_declare_cmd_code:Nnn`
 07:94, 07:109, 07:109
`__cmd_declare_cmd_code_aux:Nnn`
 07:109, 07:126, 07:153
`__cmd_declare_cmd_code_expandable:Nnn`
 07:109, 07:125, 07:181
`__cmd_declare_cmd_internal:Nnnn`
 07:58, 07:87, 07:89, 07:266, 122
`__cmd_declare_cmd_optimized:Nnn`
 07:109, 07:128, 07:131
`__cmd_declare_env:nmmm`
 07:227, 07:227,
 07:229, 07:239, 07:241, 07:243,
 07:3123, 07:3138, 07:3144, 07:3154,
 07:3169, 07:3175, 07:3179, 07:3181, 194
`__cmd_declare_expandable_-`
 cmd:Nnn 07:58, 07:63,
 07:3200, 07:3208, 07:3221, 07:3227
`__cmd_defaults:` 07:319, 07:327, 07:327
`__cmd_defaults_aux:`
 07:327, 07:331, 07:332, 07:333, 07:338
`\l__cmd_defaults_bool`
 07:18, 07:172,
 07:187, 07:210, 07:799, 07:992, 114
`__cmd_defaults_def:`
 07:327, 07:329, 07:351
`__cmd_defaults_def:nn`
 07:327, 07:356, 07:361
`__cmd_defaults_def:nnn`
 07:327, 07:364, 07:366
`__cmd_defaults_error:w`
 07:327, 07:334, 07:346
`\l__cmd_defaults_tl` 07:19,
 07:173, 07:194, 07:311, 07:319,
 07:355, 07:800, 07:993, 07:999, 114
`__cmd_delimiter_check:nnn`
 07:638, 07:647, 07:729, 07:729
`__cmd_end_expandable:NNw`
 07:407, 07:409, 07:409
`__cmd_end_expandable_aux:nNNNN`
 07:409, 07:412, 07:413
`__cmd_end_expandable_aux:w`
 07:409, 07:410, 07:411
`__cmd_end_expandable_defaults:nnNNNn`
 07:409, 07:416,
 07:425, 07:434, 07:444, 07:445, 126
`__cmd_end_expandable_defaults:nnw`
 07:409, 07:433, 07:437
`__cmd_end_expandable_defaults:nw`
 07:409, 07:440, 07:441, 07:443
`\l__cmd_environment_bool`
 07:20, 07:86, 07:113, 07:160,
 07:236, 07:258, 07:291, 07:302,
 07:321, 07:477, 07:664, 07:2805, 114
`__cmd_environment_or_command:`
 07:349, 07:463, 07:495, 07:499,
 07:581, 07:667, 07:675, 07:684,
 07:699, 07:743, 07:780, 07:2048,
 07:2208, 07:2215, 07:2803, 07:2803
`\l__cmd_environment_str`
 07:21, 07:163, 07:231,
 07:245, 07:246, 07:247, 07:248,
 07:251, 07:255, 07:260, 07:290,
 07:293, 07:294, 07:322, 07:2806, 114
`\l__cmd_expandable_aux_name_tl`
 07:23, 07:24,
 07:1028, 07:1032, 07:1041, 07:1045, 114

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
 11=ltparams.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx,
 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
 56=lthyphen.dtx, 57=ltfinal.dtx

`\l__cmd_expandable_bool`
 07:22, 07:60, 07:65, 07:133, 07:145,
 07:185, 07:197, 07:235, 07:257,
 07:466, 07:475, 07:532, 07:563,
 07:709, 07:719, 07:752, 07:807, 114
`__cmd_expandable_grab_D:nnNNwNN`
 07:2268, 07:2292, 07:2298
`__cmd_expandable_grab_D:NNwNNn`
 07:2268, 07:2269, 07:2272
`__cmd_expandable_grab_D:NNwNNnnn`
 07:2268,
 07:2283, 07:2290, 07:2316, 07:2411, 175
`__cmd_expandable_grab_D:Nw`
 07:2268, 07:2294, 07:2297
`__cmd_expandable_grab_D:w`
 07:2268, 07:2268
`__cmd_expandable_grab_D_-`
 alt:NNwn . 07:2335, 07:2342, 07:2438
`__cmd_expandable_grab_D_-`
 alt:NNwNNn 07:2320, 07:2321, 07:2324
`__cmd_expandable_grab_D_alt:Nwn`
 07:2320
`__cmd_expandable_grab_D_alt:w`
 07:2320, 07:2320
`__cmd_expandable_grab_E:w`
 07:2355, 07:2355
`__cmd_expandable_grab_E_aux:w`
 07:2355,
 07:2356, 07:2358, 07:2359, 07:2387
`__cmd_expandable_grab_E_end:nnw`
 07:2355, 07:2371, 07:2388
`__cmd_expandable_grab_E_-`
 find:nnw . 07:2355, 07:2385, 07:2386
`__cmd_expandable_grab_E_find:w`
 07:2355, 07:2376, 07:2384
`__cmd_expandable_grab_E_long:w`
 07:2355, 07:2357
`__cmd_expandable_grab_E_-`
 loop:nnnNNw
 . . 07:2355, 07:2363, 07:2367, 07:2378
`__cmd_expandable_grab_E_-`
 test:nnw . 07:2355, 07:2360, 07:2361
`__cmd_expandable_grab_m:w`
 07:2390, 07:2390, 149
`__cmd_expandable_grab_m_aux:wNn`
 . . 07:2390, 07:2391, 07:2393, 07:2394
`__cmd_expandable_grab_m_long:w`
 07:2390, 07:2392
`__cmd_expandable_grab_R:w`
 07:2396, 07:2396
`__cmd_expandable_grab_R_alt:w`
 07:2423, 07:2423
`__cmd_expandable_grab_R_alt_-`
 aux:NNwNNn 07:2423, 07:2424, 07:2427
`__cmd_expandable_grab_R_-`
 aux:NNwNNn 07:2396, 07:2397, 07:2400
`__cmd_expandable_grab_t:w`
 07:2450, 07:2450
`__cmd_expandable_grab_t_-`
 aux:NNwn . 07:2450, 07:2451, 07:2452
`\l__cmd_final_verb_bool`
 . . . 07:48, 07:812, 07:813, 07:962, 116
`__cmd_flush_m_args:` 07:807,
 07:832, 07:839, 07:846, 07:854,
 07:868, 07:875, 07:883, 07:915,
 07:923, 07:931, 07:936, 07:936, 136
`\l__cmd_fn_code_tl`
 07:31, 07:310, 07:325, 115
`\l__cmd_fn_tl`
 . . . 07:30, 07:122, 07:309, 07:1574,
 07:1834, 07:1858, 07:1864, 07:1874,
 07:1884, 07:1893, 07:1898, 07:1902,
 07:1933, 07:1959, 07:1967, 07:1969,
 07:1974, 07:1976, 07:1987, 07:1988,
 07:1993, 07:1994, 07:1999, 07:2000,
 07:2005, 07:2006, 07:2011, 07:2012,
 07:2017, 07:2018, 07:2023, 07:2024,
 07:2029, 07:2030, 07:2060, 07:2066, 118
`\l__cmd_function_tl`
 07:26, 07:32, 07:91, 07:123,
 07:139, 07:142, 07:156, 07:167,
 07:168, 07:184, 07:192, 07:202,
 07:205, 07:209, 07:211, 07:219,
 07:225, 07:471, 07:535, 07:566,
 07:712, 07:724, 07:755, 07:2809, 115
`__cmd_get_grabber:NN`
 07:1067, 07:1083, 07:1096, 07:1096, 143
`__cmd_get_grabber_auxi:NN`
 . . 07:1096, 07:1099, 07:1102, 07:1110
`__cmd_get_grabber_auxii:NN`
 07:1096, 07:1117, 07:1120
`__cmd_grab_b:w` 07:1560, 07:1560
`__cmd_grab_b_aux:NNw`
 07:1560, 07:1561,
 07:1563, 07:1565, 07:1567, 07:1568
`__cmd_grab_b_end:Nw`
 07:1560, 07:1571, 07:1576
`__cmd_grab_b_long:w` 07:1560, 07:1562
`__cmd_grab_b_long_obey_spaces:w`
 07:1560, 07:1566
`__cmd_grab_b_obey_spaces:w`
 07:1560, 07:1564
`__cmd_grab_c:w` 07:1587, 07:1587
`__cmd_grab_c_auxi:w`
 07:1623, 07:1628, 07:1630
`__cmd_grab_c_auxii:w`
 07:1628, 07:1640, 07:1646

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
 06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
 11=ltparams.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx,
 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
 56=lthyphen.dtx, 57=ltfinal.dtx

`__cmd_grab_c_auxiii:N`
 .. [07:1628](#), [07:1652](#), [07:1654](#), [07:1663](#)
`__cmd_grab_c_auxiv:`
 .. [07:1628](#), [07:1659](#), [07:1674](#), [07:1697](#)
`__cmd_grab_c_auxv:` [07:1628](#),
 [07:1671](#), [07:1683](#), [07:1702](#), [07:1724](#)
`__cmd_grab_c_auxvi:N`
 .. [07:1628](#), [07:1668](#), [07:1690](#), [07:1712](#)
`__cmd_grab_c_auxvii:` .. [07:1628](#),
 [07:1696](#), [07:1701](#), [07:1716](#), [07:1723](#)
`__cmd_grab_c_auxviii:`
 [07:1628](#), [07:1707](#), [07:1718](#)
`__cmd_grab_c_end:n`
 [07:1727](#), [07:1743](#), [07:1749](#)
`__cmd_grab_c_end:w`
 [07:1721](#), [07:1727](#), [07:1729](#)
`__cmd_grab_c_end_auxi:w`
 [07:1727](#), [07:1751](#), [07:1754](#)
`__cmd_grab_c_end_auxii:w`
 [07:1727](#), [07:1755](#), [07:1756](#)
`__cmd_grab_c_end_auxiii:w`
 [07:1727](#), [07:1757](#), [07:1758](#)
`__cmd_grab_c_first:w`
 [07:1587](#), [07:1607](#), [07:1611](#)
`__cmd_grab_c_loop:w`
 [07:1587](#), [07:1619](#),
 [07:1621](#), [07:1638](#), [07:1681](#), [07:1686](#)
`__cmd_grab_c_obey_spaces:w`
 [07:1587](#), [07:1592](#)
`__cmd_grab_c_start:n`
 .. [07:1587](#), [07:1590](#), [07:1595](#), [07:1597](#)
`__cmd_grab_D:w` [07:1760](#)
`__cmd_grab_D_aux:NNnNN` . [07:1570](#),
 [07:1791](#), [07:1795](#), [07:1831](#), [07:2043](#)
`__cmd_grab_D_aux:NNnNNNN`
 [07:1772](#), [07:1791](#), [07:1793](#)
`__cmd_grab_D_call:Nw`
 [07:1804](#), [07:1888](#), [07:1888](#), [07:2045](#), [161](#)
`__cmd_grab_D_long:w` [07:1760](#)
`__cmd_grab_D_long_no_strip:w` [07:1760](#)
`__cmd_grab_D_long_obey_spaces:w`
 [07:1760](#)
`__cmd_grab_D_long_obey_spaces_-`
 no_strip:w [07:1760](#)
`__cmd_grab_D_nested:NNnN`
 [07:1837](#), [07:1851](#), [07:1854](#)
`__cmd_grab_D_nested:w`
 [07:1851](#), [07:1869](#), [07:1886](#)
`__cmd_grab_D_no_strip:w` ... [07:1760](#)
`__cmd_grab_D_obey_spaces:w` [07:1760](#)
`__cmd_grab_D_obey_spaces_no_-`
 strip:w [07:1760](#)
`__cmd_grab_D_verb_safe:NN`
 [07:1791](#), [07:1799](#), [07:1812](#)
`__cmd_grab_E:nnNN` [07:1907](#), [07:1909](#),
 [07:1915](#), [07:1921](#), [07:1927](#), [07:1931](#)
`__cmd_grab_E:w` [07:1907](#), [07:1907](#)
`__cmd_grab_E_finalise:`
 .. [07:1907](#), [07:1940](#), [07:1956](#), [07:1963](#)
`__cmd_grab_E_long:w` [07:1907](#), [07:1913](#)
`__cmd_grab_E_long_obey_spaces:w`
 [07:1907](#), [07:1925](#)
`__cmd_grab_E_loop:NnN` .. [07:1907](#),
 [07:1936](#), [07:1951](#), [07:1953](#), [07:1960](#)
`__cmd_grab_E_obey_spaces:w`
 [07:1907](#), [07:1919](#)
`\l__cmd_grab_expandably_bool` ...
 [07:33](#), [07:124](#),
 [07:451](#), [07:476](#), [07:478](#), [07:540](#),
 [07:570](#), [07:595](#), [07:609](#), [07:631](#),
 [07:648](#), [07:715](#), [07:758](#), [07:826](#), [127](#)
`__cmd_grab_m:w` [07:1964](#), [07:1964](#)
`__cmd_grab_m_1:w` [07:1978](#)
`__cmd_grab_m_2:w` [07:1978](#)
`__cmd_grab_m_3:w` [07:1978](#)
`__cmd_grab_m_4:w` [07:1978](#)
`__cmd_grab_m_5:w` [07:1978](#)
`__cmd_grab_m_6:w` [07:1978](#)
`__cmd_grab_m_7:w` [07:1978](#)
`__cmd_grab_m_8:w` [07:1978](#)
`__cmd_grab_m_9:w` [07:1978](#)
`__cmd_grab_m_aux:Nnnnnnnnn`
 [07:1978](#), [07:1978](#),
 [07:1987](#), [07:1993](#), [07:1999](#), [07:2005](#),
 [07:2011](#), [07:2017](#), [07:2023](#), [07:2029](#)
`__cmd_grab_m_long:w` [07:1964](#), [07:1971](#)
`__cmd_grab_R:w` [07:2037](#), [07:2037](#)
`__cmd_grab_R_aux:NNnN`
 .. [07:2037](#), [07:2038](#), [07:2040](#), [07:2041](#)
`__cmd_grab_R_long:w` [07:2037](#), [07:2039](#)
`__cmd_grab_t:w` [07:2053](#), [07:2053](#)
`__cmd_grab_t_aux:NNw`
 .. [07:2053](#), [07:2054](#), [07:2056](#), [07:2057](#)
`__cmd_grab_t_obey_spaces:w`
 [07:2053](#), [07:2055](#)
`__cmd_grab_v:w` [07:2069](#), [07:2069](#)
`__cmd_grab_v_aux:w`
 [07:2069](#), [07:2072](#), [07:2077](#), [07:2079](#), [171](#)
`__cmd_grab_v_aux_abort:n`
 [07:2096](#), [07:2114](#), [07:2120](#), [07:2133](#),
 [07:2152](#), [07:2175](#), [07:2177](#), [07:2201](#), [170](#)
`__cmd_grab_v_aux_catcodes:`
 [07:2111](#), [07:2143](#), [07:2177](#),
 [07:2177](#), [07:2179](#), [07:2190](#), [07:2192](#), [169](#)
`__cmd_grab_v_aux_loop:N`
 .. [07:2106](#), [07:2112](#), [07:2116](#), [07:2130](#)
`__cmd_grab_v_aux_loop:NN`
 [07:2106](#), [07:2119](#), [07:2122](#)

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx,
16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
56=ltthyphen.dtx, 57=ltfinal.dtx

- __cmd_grab_v_aux_loop_end:
.. [07:2106](#), [07:2127](#), [07:2135](#), [07:2166](#)
- __cmd_grab_v_aux_put:N
[07:2110](#), [07:2129](#), [07:2145](#), [07:2163](#),
[07:2171](#), [07:2221](#), [07:2221](#), [07:2223](#),
[07:2233](#), [07:2235](#), [07:2249](#), [07:2251](#)
- __cmd_grab_v_aux_test:N
.. [07:2095](#), [07:2106](#), [07:2106](#)
- __cmd_grab_v_bgroup:
.. [07:2091](#), [07:2141](#), [07:2141](#)
- __cmd_grab_v_bgroup_loop:
.. [07:2141](#),
[07:2146](#), [07:2148](#), [07:2164](#), [07:2172](#)
- __cmd_grab_v_bgroup_loop:N
.. [07:2141](#), [07:2151](#), [07:2154](#)
- __cmd_grab_v_group_end:
[07:2069](#), [07:2100](#), [07:2137](#), [07:2203](#), [169](#)
- __cmd_grab_v_long:w [07:2069](#), [07:2074](#)
- __cmd_grab_v_token_if_char:NTF
.. [07:2108](#),
[07:2124](#), [07:2156](#), [07:2260](#), [07:2260](#), [170](#)
- \g__cmd_grabber_int
.. [07:29](#), [07:1109](#), [07:1113](#), [114](#)
- __cmd_if_recursion_tail_stop_
do:Nn . [07:49](#), [07:51](#), [07:2609](#), [07:2637](#)
- \c__cmd_ignore_def_tl . . [07:2801](#),
[07:2817](#), [07:2824](#), [07:2838](#), [07:2871](#),
[07:2900](#), [07:2907](#), [07:2914](#), [07:2922](#),
[07:2930](#), [07:2939](#), [07:2946](#), [07:2953](#),
[07:2960](#), [07:2967](#), [07:2979](#), [07:2986](#), [188](#)
- \l__cmd_last_delimiters_tl
.. [07:35](#), [07:449](#),
[07:468](#), [07:594](#), [07:608](#), [07:630](#),
[07:670](#), [07:721](#), [07:731](#), [07:788](#), [130](#)
- \l__cmd_long_bool . . [07:36](#), [07:453](#),
[07:546](#), [07:547](#), [07:633](#), [07:749](#),
[07:760](#), [07:766](#), [07:772](#), [07:795](#),
[07:833](#), [07:954](#), [07:973](#), [07:1010](#),
[07:1025](#), [07:1038](#), [07:1055](#), [07:1075](#),
[07:2071](#), [07:2076](#), [07:2185](#), [07:2196](#), [115](#)
- \l__cmd_m_args_int
.. [07:38](#), [07:798](#), [07:910](#),
[07:938](#), [07:941](#), [07:943](#), [07:945](#), [115](#)
- \l__cmd_nesting_a_tl [07:1851](#)
- \l__cmd_nesting_b_tl [07:1851](#)
- __cmd_new_env:nnnn
.. [07:3107](#), [07:3111](#),
[07:3125](#), [07:3156](#), [07:3182](#), [07:3183](#)
- __cmd_normalize_arg_spec:n
.. [07:92](#), [07:446](#), [07:446](#)
- __cmd_normalize_arg_spec_loop:n
.. [07:446](#), [07:458](#),
[07:480](#), [07:541](#), [07:586](#), [07:596](#),
[07:611](#), [07:634](#), [07:640](#), [07:650](#), [07:656](#)
- __cmd_normalize_check_gv:N
.. [07:654](#), [07:707](#), [07:707](#)
- __cmd_normalize_check_lu:N
.. [07:707](#), [07:717](#)
- __cmd_normalize_E_unique_
check:w [07:588](#), [07:604](#), [07:613](#), [07:618](#)
- __cmd_normalize_type_!:w . . . [07:529](#)
- __cmd_normalize_type_+:w . . . [07:529](#)
- __cmd_normalize_type_=:w . . . [07:529](#)
- __cmd_normalize_type_>:w . . . [07:529](#)
- __cmd_normalize_type_aux:NnNn
.. [07:529](#), [07:545](#), [07:551](#), [07:560](#), [07:575](#)
- __cmd_normalize_type_b:w
.. [07:658](#), [07:658](#)
- __cmd_normalize_type_b_or_c:nn
.. [07:658](#), [07:659](#), [07:661](#), [07:662](#)
- __cmd_normalize_type_c:w
.. [07:658](#), [07:660](#)
- __cmd_normalize_type_D:w
.. [07:509](#), [07:517](#), [07:519](#), [07:588](#), [07:588](#)
- __cmd_normalize_type_d:w
.. [07:504](#), [07:506](#)
- __cmd_normalize_type_E:w
.. [07:514](#), [07:588](#), [07:598](#)
- __cmd_normalize_type_e:w
.. [07:504](#), [07:511](#)
- __cmd_normalize_type_m:w
.. [07:636](#), [07:636](#)
- __cmd_normalize_type_0:w
.. [07:504](#), [07:518](#)
- __cmd_normalize_type_o:w
.. [07:504](#), [07:516](#)
- __cmd_normalize_type_R:w
.. [07:523](#), [07:636](#), [07:642](#)
- __cmd_normalize_type_r:w
.. [07:504](#), [07:520](#)
- __cmd_normalize_type_s:w
.. [07:504](#), [07:525](#)
- __cmd_normalize_type_t:w
.. [07:526](#), [07:588](#), [07:621](#)
- __cmd_normalize_type_v:w
.. [07:636](#), [07:652](#)
- \l__cmd_obey_spaces_bool
.. [07:34](#), [07:452](#), [07:552](#), [07:554](#),
[07:627](#), [07:632](#), [07:769](#), [07:773](#),
[07:782](#), [07:796](#), [07:840](#), [07:955](#),
[07:974](#), [07:1589](#), [07:1594](#), [07:1741](#), [140](#)
- __cmd_peek_cs_check_equal:NNN
.. [07:2765](#), [07:2779](#), [07:2785](#)
- __cmd_peek_meaning:NTF
.. [07:2756](#), [07:2765](#), [07:2765](#)
- __cmd_peek_meaning_aux:NNTF
.. [07:2765](#), [07:2766](#), [07:2768](#), [07:2769](#)

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

`__cmd_peek_meaning_remove:NTF` 07:1990, 07:1996, 07:2002, 07:2008,
 07:1778, 07:1923, 07:1929, 07:2014, 07:2020, 07:2026, 07:2037,
 07:2056, 07:2758, 07:2765, 07:2767 07:2039, 07:2057, 07:2079, 07:2265, 122
`__cmd_peek_nonnamespace:NTF`
 07:2755, 07:2755 `\l__cmd_saved_args_tl`
 07:43, 07:1572, 07:1580, 116
`__cmd_peek_nonnamespace_aux:nNNTF`
 07:2755, `__cmd_set_environment_end:n`
 07:227, 07:278, 07:322
`__cmd_peek_nonnamespace_remove:NTF`
 07:1779, 07:1817, 07:1911, 07:1917, 07:1166, 07:1185, 07:1186, 07:1187,
 07:2044, 07:2054, 07:2755, 07:2757 07:1200, 07:1201, 07:1202, 07:1300
`__cmd_peek_true_remove:NNw` 07:2765 `__cmd_show:N`
`__cmd_peek_true_remove:Nw` 07:1344, 07:1346, 07:1346, 07:1373,
 07:2776, 07:2780, 07:2788, 07:2792 07:1396, 07:1473, 07:1475, 07:1557, 154
`\l__cmd_prefixed_bool`
 07:39, 07:817, 07:834, `__cmd_show:n` 07:1363, 07:1459, 07:1465
 07:841, 07:847, 07:855, 07:908, 115 `__cmd_show_command:N`
 07:1352, 07:1363, 07:1363
`__cmd_prepare_signature:N`
 07:791, 07:806, 07:815, `__cmd_show_command:NnNNwN`
 07:1363, 07:1364, 07:1365
 07:871, 07:879, 07:892, 07:911, `__cmd_show_command_aux:NnNNn`
 07:919, 07:927, 07:934, 07:1011, 07:1363, 07:1367,
 07:1021, 07:1063, 07:1076, 07:1089, 137 07:1380, 07:1389, 07:1398, 07:1459, 153
`__cmd_prepare_signature:n`
 07:93, 07:791, 07:791 `__cmd_show_delim:Nw` 07:1505, 07:1505
`__cmd_prepare_signature_-` `__cmd_show_delims:Nw` 07:1505, 07:1507
`bypass:N` 07:791, 07:818, 07:820, `__cmd_show_delims_opt:Nw`
 07:835, 07:842, 07:850, 07:860, 137 07:1505, 07:1509
`__cmd_prepare_signature_verb_-` `__cmd_show_E:Nw` 07:1505, 07:1524
`chk:n` 07:791, 07:805, 07:809 `__cmd_show_e:Nw` 07:1505, 07:1513
`\l__cmd_process_all_tl`
 07:40, 07:177, 07:312, 07:320, `__cmd_show_environment:N`
 07:385, 07:801, 07:942, 07:977, 116 07:1355, 07:1363, 07:1445, 07:1470
`\l__cmd_process_one_tl`
 07:41, 07:802, 07:849, 07:858, `__cmd_show_environment:Nnw`
 07:888, 07:897, 07:981, 07:984, 139 07:1447, 07:1455
`\l__cmd_process_some_bool` . 07:42, `__cmd_show_environment_end:N`
 07:176, 07:803, 07:848, 07:857, 116 07:1356, 07:1467
`__cmd_provide_env:nnnn`
 07:3107, 07:3119, `__cmd_show_expandable:N`
 07:3150, 07:3158, 07:3186, 07:3187 07:1353, 07:1363, 07:1370
`__cmd_put_arg_expandable:nw`
 07:2304, `__cmd_show_expandable:NnNNNNnN`
 07:2309, 07:2310, 07:2345, 07:2350, 07:1363, 07:1371, 07:1376,
 07:2351, 07:2458, 07:2458, 07:2459 07:1378, 07:1385, 07:1387, 07:1393
`__cmd_renew_env:nnnn`
 07:3107, 07:3115, `__cmd_show_opt:Nw` 07:1505, 07:1511
 07:3141, 07:3157, 07:3184, 07:3185 `__cmd_show_optimized:N`
 07:1354, 07:1363, 07:1408
`__cmd_replicate_processor:nn`
 07:887, 07:894, 07:894 `__cmd_show_optimized:NN`
 07:1410, 07:1414
`__cmd_run_code:` . . 07:314, 07:317, `__cmd_show_optimized_aux:N`
 07:317, 07:1568, 07:1576, 07:1584, 07:1416, 07:1439
 07:1587, 07:1592, 07:1770, 07:1907, `__cmd_show_prefix:Nw` 07:1505, 07:1537
 07:1913, 07:1919, 07:1925, 07:1949, `__cmd_show_processor:Nw`
 07:1964, 07:1971, 07:1982, 07:1984, 07:1505, 07:1539
`\c__cmd_show_type_!_tl` 07:1499
`\c__cmd_show_type+_tl` 07:1499
`\c__cmd_show_type>_tl` 07:1499
`\c__cmd_show_type_D_tl` 07:1499
`\c__cmd_show_type_d_tl` 07:1499

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx,
 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
 26=ltfssstrc.dtx, 27=ltfsscscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
 56=ltthyphen.dtx, 57=ltfinal.dtx

`\c__cmd_show_type_E_tl` [07:1499](#)
`\c__cmd_show_type_e_tl` [07:1499](#)
`\c__cmd_show_type_0_tl` [07:1499](#)
`\c__cmd_show_type_R_tl` [07:1499](#)
`\c__cmd_show_type_r_tl` [07:1499](#)
`\c__cmd_show_type_t_tl` [07:1499](#)
`\l__cmd_signature_tl` [07:44](#), [07:170](#),
[07:213](#), [07:804](#), [07:878](#), [07:891](#),
[07:918](#), [07:926](#), [07:940](#), [07:949](#),
[07:1093](#), [07:1571](#), [07:1599](#), [07:1833](#),
[07:1939](#), [07:1949](#), [07:1966](#), [07:1973](#),
[07:1982](#), [07:1986](#), [07:1992](#), [07:1998](#),
[07:2004](#), [07:2010](#), [07:2016](#), [07:2022](#),
[07:2028](#), [07:2059](#), [07:2081](#), [07:2265](#), [116](#)
`__cmd_single_token_check:n`
. [07:591](#), [07:592](#), [07:602](#),
[07:624](#), [07:645](#), [07:646](#), [07:679](#), [07:679](#)
`\l__cmd_some_long_bool`
. [07:46](#), [07:117](#), [07:146](#), [07:207](#),
[07:215](#), [07:456](#), [07:750](#), [07:761](#), [127](#)
`\l__cmd_some_obey_spaces_bool`
. [07:45](#), [07:455](#), [07:555](#), [07:777](#), [116](#)
`\l__cmd_some_short_bool`
. [07:47](#), [07:116](#),
[07:206](#), [07:217](#), [07:457](#), [07:762](#), [116](#)
`__cmd_split_add_item:n`
. [07:1538](#), [07:1541](#), [07:1540](#), [07:1546](#), [156](#)
`__cmd_split_argument:nnn`
. [07:2501](#), [07:2501](#), [07:3269](#)
`__cmd_split_argument_aux:n`
. [07:2501](#), [07:2517](#), [07:2536](#)
`__cmd_split_argument_aux:nnnn`
. [07:2501](#), [07:2504](#), [07:2508](#)
`__cmd_split_argument_aux:wn`
. [07:2501](#), [07:2537](#), [07:2538](#)
`__cmd_split_end_item:n` [07:1497](#),
[07:1506](#), [07:1508](#), [07:1510](#), [07:1512](#),
[07:1518](#), [07:1529](#), [07:1541](#), [07:1548](#), [156](#)
`__cmd_split_list:nn`
. [07:2466](#), [07:2468](#), [07:2503](#), [07:3270](#)
`__cmd_split_list_multi:nn`
. [07:2466](#), [07:2473](#),
[07:2476](#), [07:2478](#), [07:2485](#), [07:2498](#)
`\l__cmd_split_list_seq`
. [07:2466](#), [07:2480](#), [07:2482](#), [180](#)
`__cmd_split_list_single:Nn`
. [07:2466](#), [07:2474](#), [07:2488](#)
`\l__cmd_split_list_tl`
. [07:2467](#), [07:2490](#), [07:2496](#), [07:2498](#), [180](#)
`__cmd_split_N_head_apply:Nn`
. [07:2546](#), [07:2566](#), [07:2578](#), [07:2653](#)
`__cmd_split_N_head_apply_-`
aux:NNw [07:2546](#), [07:2654](#), [07:2655](#)
`__cmd_split_signature:n`
. [07:1400](#), [07:1476](#), [07:1476](#), [152](#)
`__cmd_split_signature_loop:Nw`
. [07:1481](#), [07:1483](#), [07:1483](#), [07:1497](#),
[07:1506](#), [07:1508](#), [07:1510](#), [07:1512](#),
[07:1520](#), [07:1533](#), [07:1538](#), [07:1540](#)
`__cmd_split_start_item:` [07:1486](#),
[07:1517](#), [07:1528](#), [07:1541](#), [07:1541](#), [154](#)
`__cmd_start:nNNnnn`
. [07:166](#), [07:286](#), [07:297](#), [07:2736](#), [119](#)
`__cmd_start_aux:NNnnnn`
. [07:292](#), [07:303](#), [07:306](#), [07:306](#), [07:316](#)
`__cmd_start_env:nnnnn`
. [07:162](#), [07:286](#), [07:286](#), [07:2739](#), [186](#)
`__cmd_start_expandable:nNNNNn`
. [07:200](#), [07:404](#), [07:404](#), [07:2737](#), [1351](#)
`__cmd_start_optimized:` [07:109](#),
[07:138](#), [07:152](#), [07:1219](#), [07:2738](#), [186](#)
`\l__cmd_suppress_strip_bool`
. [07:37](#), [07:454](#), [07:561](#), [07:569](#),
[07:797](#), [07:856](#), [07:957](#), [07:975](#), [115](#)
`__cmd_tl_mapthread_function:NNN`
. [07:355](#), [07:383](#), [07:2702](#), [07:2702](#)
`__cmd_tl_mapthread_function:nnN`
. [07:431](#), [07:1531](#), [07:2702](#), [07:2715](#), [155](#)
`__cmd_tl_mapthread_loop:w`
. [07:2702](#),
[07:2705](#), [07:2717](#), [07:2721](#), [07:2726](#)
`__cmd_tmp:w` [07:52](#),
[07:55](#), [07:341](#), [07:357](#), [07:399](#),
[07:401](#), [07:504](#), [07:528](#), [07:1098](#),
[07:1104](#), [07:1122](#), [07:1329](#), [07:1339](#),
[07:1499](#), [07:1502](#), [07:1504](#), [07:1522](#),
[07:1526](#), [07:1532](#), [07:1536](#), [07:2270](#),
[07:2289](#), [07:2322](#), [07:2341](#), [07:2398](#),
[07:2422](#), [07:2425](#), [07:2449](#), [07:2797](#), [174](#)
`\l__cmd_tmp_prop`
. [07:52](#), [07:1935](#), [07:1938](#), [07:1944](#), [117](#)
`\l__cmd_tmpa_tl`
. [07:53](#), [07:353](#), [07:358](#),
[07:368](#), [07:1067](#), [07:1069](#), [07:1083](#),
[07:1086](#), [07:1240](#), [07:1246](#), [07:1261](#),
[07:1269](#), [07:1285](#), [07:1292](#), [07:1310](#),
[07:1313](#), [07:1404](#), [07:1469](#), [07:1470](#),
[07:1479](#), [07:1550](#), [07:1650](#), [07:1662](#),
[07:1678](#), [07:1685](#), [07:1717](#), [07:1872](#),
[07:1875](#), [07:2771](#), [07:2796](#), [07:2799](#), [152](#)
`\l__cmd_tmpb_tl`
. [07:54](#), [07:1052](#), [07:1057](#),
[07:1068](#), [07:1311](#), [07:1313](#), [07:1480](#),
[07:1486](#), [07:1543](#), [07:1547](#), [07:1551](#),
[07:1552](#), [07:1667](#), [07:1706](#), [07:1711](#),
[07:1717](#), [07:1720](#), [07:1944](#), [07:1945](#),
[07:1947](#), [07:2772](#), [07:2783](#), [07:2789](#), [143](#)

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
11=ltemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx,
16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
56=ltthyphen.dtx, 57=ltfinal.dtx

<code>__cmd_token_if_cs:NTF</code>	<code>\cr</code>
07:1896, 07:2693, 07:2693, 07:2778, 133	21:521, 21:527, 21:537, 21:543,
<code>\l__cmd_total_args_int</code>	33:65, 33:69, 33:891, 33:895, 38:188,
. 07:17, 07:793, 07:966, 114	38:192, 38:378, 38:424, 38:540,
<code>__cmd_trim_spaces:n</code>	41:192, 41:203, 41:210, 41:219,
. 07:2544, 07:2544, 07:3271	41:224, 41:230, 41:377, 42:141,
<code>__cmd_use_i_delimit_by_q-</code>	42:143, 42:148, 42:154, 02:446, 1251
recursion_stop:nw . 07:49, 07:2613	<code>\crrcr</code> 21:347, 21:382, 21:383,
<code>\l__cmd_v_arg_tl</code>	21:410, 21:414, 21:417, 21:497,
07:1601, 07:1634, 07:1637, 07:1676,	21:501, 21:505, 21:507, 21:510,
07:1685, 07:1738, 07:1744, 07:2068,	21:754, 21:782, 21:786, 21:789,
07:2085, 07:2104, 07:2138, 07:2209,	21:856, 21:859, 21:917, 21:1275,
07:2216, 07:2225, 07:2237, 07:2253, 169	29:652, 30:348, 30:349, 30:351,
<code>\l__cmd_v_nesting_int</code> . . 07:2140,	30:470, 30:473, 30:477, 30:541,
07:2144, 07:2160, 07:2161, 07:2170, 171	30:542, 30:543, 30:544, 30:545,
<code>\l__cmd_verb_safe_bool</code> 07:976	30:546, 30:548, 30:549, 30:550,
<code>\cmrFont</code> 33:1213, 33:1216	30:551, 30:552, 30:554, 33:70,
<code>\colon</code> 30:514	33:896, 38:168, 38:170, 38:171,
<code>\color</code> 419	38:172, 38:188, 38:190, 38:191,
<code>\columnbreak</code> 1015	38:192, 38:210, 38:211, 41:171,
<code>\columnsep</code> 20:27, 20:97, 20:154, 54:77, 54:200	41:172, 42:141, 55:298, 02:536, 1256
<code>\columnseprule</code>	create commands:
. 54:78, 54:2810, 54:2853, 54:2884	create_callback 04:773
<code>\columnwidth</code>	<code>\create_callback</code> 46
. 20:24, 20:27, 20:28, 20:30, 20:94,	<code>\CS</code> 82
20:97, 20:98, 20:100, 20:151, 20:154,	<code>\cs</code> 35:158
20:155, 20:158, 40:444, 40:483,	cs commands:
40:501, 40:518, 45:99, 45:168,	<code>\cs:w</code>
45:470, 45:489, 45:507, 54:76,	08:2223, 08:2242, 08:2248, 08:2262,
54:142, 54:143, 54:144, 54:199,	08:2272, 08:2294, 08:2348, 08:2435,
54:200, 54:201, 54:202, 54:203,	08:2500, 08:2550, 08:2933, 08:2953,
54:2343, 54:2345, 54:2808, 54:2812,	08:2962, 08:2963, 09:34, 28:335,
54:2851, 54:2855, 54:2880, 54:2886	28:338, 51:183, 52:211, 52:225,
<code>\cong</code> 30:468	52:237, 52:238, 07:1899, 07:2742,
<code>\contentsline</code>	08:381, 08:1067, 08:1124, 08:1275,
. 44:164, 44:171, 44:177, 44:209, 1310	08:1415, 08:1522, 08:1523, 08:1584,
<code>\coprod</code> 30:352	08:1604, 08:1607, 08:1668, 08:1693,
<code>\copyright</code> . . 21:305, 21:336, 29:650, 1309	08:1696, 08:1714, 08:1715, 08:1739,
<code>\coremissesfalse</code> 33:1220	08:1740, 08:1741, 08:1748, 08:1755, 316
<code>\coremisestrue</code> 33:1198	<code>\cs_argument_spec:N</code> 05:141, 1354
<code>\cos</code> 38:12	<code>\cs_end:</code> 08:2100, 08:2222, 08:2226,
<code>\cosh</code> 38:14	08:2230, 08:2242, 08:2247, 08:2248,
<code>\cot</code> 38:18	08:2262, 08:2268, 08:2282, 08:2295,
<code>\coth</code> 38:19	08:2300, 08:2348, 08:2434, 08:2435,
<code>\countdef</code> 04:75, 04:85,	08:2500, 08:2550, 08:2933, 08:2953,
04:174, 04:190, 04:198, 04:206,	08:2963, 09:34, 28:335, 28:338,
04:225, 34:3, 02:37, 02:38, 02:39,	51:183, 52:211, 52:225, 52:237,
02:41, 02:51, 57:75, 02:90, 01:50	52:238, 07:1899, 07:2744, 08:381,
counter 36:242, 810	08:768, 08:1067, 08:1124, 08:1278,
<code>\counterwithin</code> 521	08:1418, 08:1522, 08:1523, 08:1584,
<code>\counterwithin</code> 22:128, 1345	08:1604, 08:1607, 08:1668, 08:1693,
<code>\counterwithout</code> 521	08:1696, 08:1714, 08:1715, 08:1730,
<code>\counterwithout</code> 22:128, 1345	08:1740, 08:1741, 08:1748, 08:1755
<code>\CountZero</code> 04:225	<code>\cs_generate_from_arg_count:NNnn</code>
 10:86,

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- 10:103, 11:404, 11:466, 11:825,
07:141, 07:155, 07:183, 07:191,
07:271, 07:357, 07:360, 07:399, 1353
- `\cs_generate_variant:Nn` . . . 09:20,
11:64, 11:349, 11:509, 11:710,
11:741, 11:814, 20:528, 36:37,
36:52, 36:62, 36:79, 36:96, 36:116,
36:117, 36:230, 48:284, 51:51,
51:52, 51:53, 51:180, 52:461, 52:469,
07:11, 07:239, 07:316, 07:360,
57:600, 57:609, 57:610, 07:1147,
07:1444, 07:2267, 07:2459, 07:2485,
07:3156, 07:3157, 07:3158, 08:37,
08:38, 08:39, 08:40, 08:41, 08:42,
08:43, 08:53, 08:54, 08:57, 08:66,
08:1081, 08:1104, 08:1634, 08:1717
- `\cs_gset:Npn`
08:2061, 08:2305, 08:2375, 11:818,
28:286, 28:340, 36:24, 52:242,
53:173, 08:1080, 08:1100, 08:1630
- `\cs_gset:Npx` 52:213, 08:1048
- `\cs_gset_eq:NN`
08:2211, 08:2312, 08:2313, 08:2314,
08:2315, 08:2579, 09:73, 09:556,
09:584, 11:649, 24:414, 28:284,
28:294, 28:332, 28:346, 05:136,
05:137, 05:138, 05:140, 05:141,
05:142, 05:143, 05:144, 05:145,
53:52, 53:151, 53:166, 53:167,
53:172, 53:181, 53:185, 53:321,
53:387, 07:3273, 08:771, 08:1425,
08:1451, 08:1466, 08:1467, 08:1470
- `\cs_gset_nopar:Npn` 08:2961
- `\cs_gset_nopar:Npx` 08:50, 08:52
- `\cs_gset_protected:Npn`
. 08:2156, 08:2159,
08:2187, 08:2192, 08:2210, 08:2369,
08:2399, 08:2795, 08:2938, 09:248,
09:440, 09:562, 11:406, 11:835,
51:67, 08:88, 08:91, 08:100, 08:133,
08:160, 08:188, 08:191, 08:197,
08:221, 08:226, 08:251, 08:310,
08:328, 08:582, 08:585, 08:603,
08:629, 08:678, 08:686, 08:1449, 08:1510
- `\cs_gset_protected:Npx` 10:21,
10:23, 48:339, 53:20, 08:20, 08:1162
- `\cs_if_eq:NNTF` 51:100,
51:102, 51:168, 07:1104, 07:1955
- `\cs_if_exist:N` 1096
- `\cs_if_exist:NTF` . 08:2481, 08:2510,
08:2518, 09:90, 09:98, 10:118, 11:54,
11:60, 11:73, 11:86, 11:402, 11:460,
11:495, 11:647, 11:670, 11:686,
05:139, 36:67, 36:157, 51:64, 51:170,
52:93, 52:438, 52:442, 07:70, 07:232,
07:248, 07:1107, 07:1146, 07:3075,
07:3088, 07:3100, 07:3127, 07:3133,
07:3143, 07:3152, 07:3164, 07:3167,
07:3174, 07:3179, 07:3194, 07:3207,
07:3220, 08:1052, 08:1117, 1117
- `\cs_if_exist_p:N` . 07:77, 07:78, 08:277
- `\cs_if_exist_use:NTF`
. 11:735, 05:184, 53:313,
53:314, 53:318, 53:319, 07:484,
07:1263, 08:1284, 08:1328, 08:1353
- `\cs_if_free:NTF` 36:8, 51:109
- `\cs_meaning:N` 10:69
- `\cs_new:Npn` 08:2057, 08:2064,
08:2220, 08:2245, 08:2253, 08:2266,
08:2274, 08:2283, 08:2292, 08:2327,
08:2508, 08:2516, 08:2530, 08:2548,
08:2557, 08:2562, 08:2803, 08:2804,
08:2805, 08:2806, 08:2810, 08:2811,
09:320, 09:330, 09:340, 09:342,
09:344, 09:355, 09:378, 09:379,
09:382, 09:502, 09:610, 10:88,
10:154, 11:350, 11:351, 11:461,
11:515, 11:915, 11:1193, 11:1195,
11:1197, 16:85, 16:88, 16:113,
16:125, 20:519, 05:182, 36:63,
36:65, 36:89, 36:97, 36:104, 36:167,
36:168, 36:182, 36:183, 36:197,
36:198, 48:265, 48:275, 48:276,
48:277, 48:278, 48:283, 49:111,
49:112, 51:177, 51:179, 52:30,
52:38, 52:44, 52:57, 52:95, 52:100,
52:105, 52:112, 52:127, 52:234,
52:240, 52:411, 52:413, 52:415,
52:418, 52:422, 52:436, 52:452,
52:462, 52:582, 53:54, 53:59, 53:77,
53:141, 53:146, 53:163, 53:178,
53:183, 53:189, 53:192, 53:206,
53:257, 53:310, 53:323, 53:337,
53:340, 53:350, 53:392, 53:511,
53:512, 53:513, 07:99, 07:101,
07:152, 55:9, 55:23, 55:189, 55:195,
55:247, 55:250, 55:293, 07:404,
07:409, 07:411, 07:413, 07:425,
07:437, 07:443, 07:1170, 07:1225,
07:1306, 07:1331, 07:1336, 07:1749,
07:1754, 07:1756, 07:1758, 07:1886,
07:2268, 07:2272, 07:2290, 07:2297,
07:2298, 07:2320, 07:2324, 07:2342,
07:2355, 07:2357, 07:2359, 07:2361,
07:2367, 07:2384, 07:2386, 07:2388,
07:2390, 07:2392, 07:2394, 07:2396,
07:2400, 07:2423, 07:2427, 07:2450,
07:2452, 07:2458, 07:2536, 07:2538,

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpara.dtx,
16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
56=lthyphen.dtx, 57=ltfinal.dtx

07:2563, 07:2569, 07:2575, 07:2581,
07:2653, 07:2655, 07:2702, 07:2715,
07:2721, 07:2803, 07:3229, 07:3246,
07:3247, 07:3251, 07:3252, 07:3253,
08:45, 08:46, 08:330, 08:336, 08:351,
08:361, 08:362, 08:364, 08:378,
08:384, 08:905, 08:907, 08:909,
08:1203, 08:1215, 08:1220, 08:1228,
08:1237, 08:1239, 08:1246, 08:1273,
08:1280, 08:1282, 08:1397, 08:1413,
08:1547, 08:1548, 08:1719, 08:1720
\cs_new_eq:NN . . . 08:1796, 08:1797,
08:1798, 08:1799, 08:2282, 08:2824,
08:2825, 08:2826, 08:2827, 08:2897,
08:2899, 09:13, 09:14, 09:457, 10:7,
10:8, 10:150, 10:151, 10:152, 10:153,
10:157, 10:182, 10:183, 10:184,
10:185, 10:186, 10:187, 10:188,
10:189, 10:190, 10:191, 10:192,
10:193, 10:194, 10:195, 10:196,
10:197, 10:198, 11:356, 11:357,
11:358, 11:373, 16:135, 20:797,
20:798, 05:162, 05:163, 05:164,
05:165, 05:181, 36:166, 36:181,
36:196, 48:326, 48:342, 48:343,
48:385, 48:387, 48:499, 48:501,
51:8, 52:252, 52:253, 52:509, 52:511,
52:513, 52:515, 52:517, 52:519,
52:521, 52:523, 52:525, 52:527,
53:7, 53:152, 53:347, 53:349, 53:411,
53:414, 53:415, 53:510, 07:55,
07:98, 55:3, 55:4, 55:184, 57:630,
57:631, 07:1281, 07:1282, 07:1289,
07:1296, 07:3069, 07:3070, 07:3248,
07:3249, 07:3250, 07:3256, 07:3257,
07:3258, 07:3268, 07:3269, 07:3270,
07:3271, 07:3272, 08:7, 08:23,
08:34, 08:63, 08:1187, 08:1370,
08:1377, 08:1420, 08:1794, 08:1795
\cs_new_protected:Npe . . . 11:275,
07:1621, 07:1646, 07:1690, 07:1768
\cs_new_protected:Npn
. 08:1800, 08:1807, 08:1819,
08:1824, 08:1829, 08:1831, 08:1835,
08:1953, 08:2075, 08:2093, 08:2098,
08:2107, 08:2125, 08:2139, 08:2161,
08:2174, 08:2201, 08:2205, 08:2215,
08:2228, 08:2237, 08:2251, 08:2257,
08:2280, 08:2287, 08:2310, 08:2333,
08:2343, 08:2354, 08:2360, 08:2380,
08:2410, 08:2412, 08:2422, 08:2564,
08:2570, 08:2576, 08:2747, 08:2748,
08:2749, 08:2772, 08:2783, 08:2813,
08:2814, 08:2815, 08:2816, 08:2828,
08:2835, 08:2842, 08:2849, 08:2856,
08:2858, 08:2860, 08:2867, 08:2874,
08:2881, 08:2888, 09:18, 09:23,
09:25, 09:40, 09:42, 09:51, 09:53,
09:63, 09:71, 09:80, 09:94, 09:114,
09:134, 09:147, 09:152, 09:161,
09:166, 09:173, 09:313, 09:384,
09:421, 09:458, 09:486, 09:515, 10:9,
10:14, 10:19, 10:27, 10:58, 10:77,
10:78, 10:98, 10:122, 10:144, 11:42,
11:52, 11:58, 11:65, 11:71, 11:96,
11:104, 11:120, 11:128, 11:136,
11:146, 11:162, 11:172, 11:182,
11:188, 11:203, 11:219, 11:241,
11:253, 11:270, 11:296, 11:320,
11:352, 11:354, 11:359, 11:361,
11:363, 11:365, 11:367, 11:369,
11:371, 11:374, 11:400, 11:408,
11:419, 11:421, 11:435, 11:439,
11:453, 11:510, 11:529, 11:547,
11:568, 11:585, 11:592, 11:600,
11:605, 11:614, 11:620, 11:625,
11:634, 11:639, 11:652, 11:661,
11:680, 11:693, 11:705, 11:711,
11:717, 11:722, 11:742, 11:750,
11:755, 11:766, 11:772, 11:792,
11:809, 11:815, 11:821, 11:832,
11:838, 11:850, 11:860, 11:879,
11:881, 11:892, 11:897, 11:903,
11:909, 11:916, 11:918, 11:920,
11:929, 11:938, 11:947, 11:955,
11:1163, 11:1165, 11:1167, 11:1169,
11:1171, 11:1173, 11:1175, 11:1177,
11:1179, 11:1181, 11:1183, 11:1185,
11:1187, 11:1189, 11:1191, 11:1199,
11:1200, 11:1201, 11:1211, 16:93,
24:413, 28:279, 36:6, 36:17, 36:22,
36:38, 36:43, 36:48, 36:50, 36:53,
36:128, 36:142, 36:149, 36:199,
36:206, 36:214, 48:7, 48:17, 48:53,
48:66, 48:74, 48:118, 48:123, 48:148,
48:157, 48:199, 48:210, 48:221,
48:229, 48:232, 48:263, 48:327,
48:332, 48:337, 48:345, 48:358,
48:374, 48:403, 48:435, 51:22,
51:28, 51:44, 51:58, 51:60, 51:98,
51:107, 51:119, 51:135, 51:140,
51:149, 51:161, 51:166, 51:181,
51:185, 51:203, 51:217, 52:49,
52:62, 52:70, 52:79, 52:208, 52:222,
52:392, 52:397, 52:420, 52:547, 53:8,
53:13, 53:18, 53:343, 53:473, 07:58,
07:63, 07:68, 07:89, 07:109, 07:131,
07:153, 07:181, 07:229, 07:243,

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpara.dtx,
16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
56=ltthyphen.dtx, 57=ltfinal.dtx

07:264, 55:5, 55:6, 55:10, 55:11,
55:12, 07:278, 55:13, 55:33, 55:34,
07:286, 55:185, 55:216, 55:219,
55:239, 07:306, 55:243, 55:253,
55:256, 55:259, 55:262, 55:265,
55:282, 07:317, 07:327, 07:338,
07:346, 07:351, 07:361, 07:366,
57:580, 57:611, 57:616, 57:621,
57:629, 07:380, 07:389, 07:397,
07:446, 07:480, 07:506, 07:511,
07:516, 07:518, 07:520, 07:525,
07:529, 07:543, 07:549, 07:558,
07:575, 07:588, 07:598, 07:613,
07:621, 07:636, 07:642, 07:652,
07:658, 07:660, 07:662, 07:679,
07:688, 07:707, 07:717, 07:729,
07:740, 07:745, 07:746, 07:775,
07:791, 07:809, 07:815, 07:820,
07:830, 07:837, 07:844, 07:852,
07:862, 07:864, 07:866, 07:873,
07:881, 07:894, 07:905, 07:913,
07:921, 07:929, 07:936, 07:947,
07:986, 07:996, 07:1001, 07:1008,
07:1013, 07:1015, 07:1023, 07:1036,
07:1049, 07:1065, 07:1071, 07:1078,
07:1080, 07:1091, 07:1096, 07:1102,
07:1120, 07:1127, 07:1145, 07:1148,
07:1163, 07:1182, 07:1197, 07:1211,
07:1242, 07:1250, 07:1258, 07:1267,
07:1279, 07:1283, 07:1290, 07:1295,
07:1297, 07:1308, 07:1315, 07:1323,
07:1346, 07:1363, 07:1365, 07:1370,
07:1378, 07:1387, 07:1398, 07:1408,
07:1414, 07:1445, 07:1455, 07:1465,
07:1467, 07:1476, 07:1483, 07:1505,
07:1507, 07:1509, 07:1511, 07:1513,
07:1524, 07:1537, 07:1539, 07:1541,
07:1546, 07:1548, 07:1560, 07:1562,
07:1564, 07:1566, 07:1568, 07:1576,
07:1587, 07:1592, 07:1597, 07:1611,
07:1630, 07:1654, 07:1674, 07:1683,
07:1716, 07:1718, 07:1729, 07:1793,
07:1812, 07:1831, 07:1854, 07:1907,
07:1913, 07:1919, 07:1925, 07:1931,
07:1953, 07:1963, 07:1964, 07:1971,
07:1984, 07:1990, 07:1996, 07:2002,
07:2008, 07:2014, 07:2020, 07:2026,
07:2037, 07:2039, 07:2041, 07:2053,
07:2055, 07:2057, 07:2069, 07:2074,
07:2079, 07:2100, 07:2106, 07:2116,
07:2122, 07:2135, 07:2148, 07:2154,
07:2179, 07:2192, 07:2201, 07:2223,
07:2235, 07:2251, 07:2260, 07:2262,
07:2460, 07:2468, 07:2478, 07:2488,
07:2501, 07:2508, 07:2544, 07:2546,
07:2551, 07:2587, 07:2592, 07:2603,
07:2605, 07:2607, 07:2624, 07:2635,
07:2645, 07:2647, 07:2649, 07:2651,
07:2661, 07:2665, 07:2693, 07:2728,
07:2749, 07:2755, 07:2757, 07:2759,
07:2765, 07:2767, 07:2769, 07:2785,
07:2792, 07:3071, 07:3084, 07:3097,
07:3102, 07:3109, 07:3113, 07:3117,
07:3121, 07:3125, 07:3141, 07:3150,
07:3162, 07:3172, 07:3178, 07:3180,
07:3190, 07:3203, 07:3216, 07:3224,
08:8, 08:13, 08:18, 08:47, 08:49,
08:51, 08:55, 08:58, 08:64, 08:69,
08:71, 08:73, 08:104, 08:148, 08:172,
08:174, 08:176, 08:201, 08:203,
08:205, 08:231, 08:266, 08:268,
08:285, 08:290, 08:292, 08:385,
08:394, 08:399, 08:407, 08:431,
08:433, 08:449, 08:457, 08:467,
08:477, 08:488, 08:494, 08:500,
08:516, 08:529, 08:556, 08:573,
08:633, 08:652, 08:660, 08:667,
08:699, 08:705, 08:711, 08:721,
08:763, 08:765, 08:773, 08:775,
08:778, 08:780, 08:961, 08:963,
08:998, 08:1035, 08:1047, 08:1050,
08:1079, 08:1092, 08:1094, 08:1096,
08:1098, 08:1115, 08:1139, 08:1146,
08:1304, 08:1311, 08:1342, 08:1364,
08:1371, 08:1378, 08:1384, 08:1389,
08:1394, 08:1395, 08:1423, 08:1476,
08:1556, 08:1638, 08:1721, 08:1728,
08:1737, 08:1744, 08:1751, 08:1758,
08:1766, 08:1772, 08:1783, 1147
\cs_new_protected:Npx . . . 09:492,
48:101, 07:297, 07:2032, 07:2141
\cs_new_protected_nopar:Npn
. 57:626, 07:1888, 07:1978
\cs_parameter_spec:N
05:139, 05:140, 07:1436, 08:1231, 1355
\cs_prefix_spec:N 09:224, 09:299, 05:138
\cs_replacement_spec 258
\cs_replacement_spec:N
. 08:1877, 08:1890, 08:1899,
05:143, 07:1405, 07:1441, 07:1451,
08:37, 08:41, 08:1206, 08:1441, 08:1463
\cs_set:Npe 07:1416
\cs_set:Npn
. 08:1874, 08:2568, 08:2574,
08:2947, 09:216, 09:291, 09:473,
09:532, 09:568, 10:105, 11:328,
11:819, 16:37, 16:68, 48:360, 07:193,
07:272, 07:357, 07:399, 07:1098,

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpara.dtx,
16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
56=ltthyphen.dtx, 57=ltfinal.dtx

- 08:1148, 08:1159, 08:1166, 08:1175, 157
- `\cs_set:Npx` 52:211,
52:225, 07:222, 07:1026, 07:1039
- `\cs_set_eq:NN`
. 08:1821, 08:1826, 09:188, 09:189,
09:209, 09:210, 09:215, 09:263,
09:264, 09:284, 09:285, 09:290,
09:469, 09:470, 09:528, 09:529,
09:564, 09:565, 09:623, 11:688,
16:91, 16:92, 16:136, 16:137, 16:138,
16:140, 16:141, 16:142, 16:176,
16:177, 16:178, 16:181, 28:299,
28:349, 36:231, 48:100, 48:266,
48:267, 48:268, 51:88, 53:24, 53:44,
53:83, 53:95, 53:125, 53:132, 53:417,
53:419, 53:421, 53:423, 53:425,
07:274, 07:275, 55:179, 55:182,
07:1122, 07:1146, 07:1165, 07:1184,
07:1199, 07:1213, 07:1274, 07:1299,
07:1304, 07:1319, 07:1321, 07:1604,
07:1814, 07:1987, 07:1993, 07:1999,
07:2005, 07:2011, 07:2017, 07:2023,
07:2029, 07:2181, 07:2194, 07:2797,
08:1495, 08:1496, 08:1497, 08:1498,
08:1528, 08:1530, 08:1532, 08:1534
- `\cs_set_nopar:Npe` 07:134, 07:1216
- `\cs_set_nopar:Npn` . 57:582, 57:583, 157
- `\cs_set_nopar:Npx`
..... 07:198, 07:219, 07:224,
07:269, 07:280, 07:1027, 07:1040,
07:1190, 07:1204, 07:1317, 08:48
- `\cs_set_protected:Npe` 11:672
- `\cs_set_protected:Npn`
..... 08:2079, 08:2110, 09:190,
09:265, 11:324, 11:836, 28:327,
28:359, 51:5, 51:62, 51:81, 51:191,
53:46, 53:135, 53:412, 07:157,
57:598, 07:504, 07:1329, 07:1499,
07:1522, 07:1526, 07:1563, 07:1567,
07:1775, 07:1916, 07:1928, 07:1940,
07:1974, 07:2040, 07:2060, 07:2270,
07:2322, 07:2398, 07:2425, 08:420
- `\cs_set_protected_nopar:Npe`
..... 07:135, 07:1215
- `\cs_set_protected_nopar:Npn`
..... 07:1561, 07:1565, 07:1776,
07:1910, 07:1922, 07:1967, 07:2038
- `\cs_set_protected_nopar:Npx`
..... 07:158,
07:198, 07:1167, 07:1190, 07:1302
- `\cs_show:N` 11:919, 384
- `\cs_to_str:N`
... 09:129, 09:150, 09:164, 09:169,
05:136, 07:77, 07:78, 07:91, 07:1139,
- 07:1141, 07:1359, 07:1411, 07:1450,
07:1452, 07:1469, 07:1899, 07:2744, 241
- `\cs_undefine:N` 08:2320,
08:2406, 08:2544, 09:76, 09:227,
09:511, 09:519, 52:227, 07:3182,
07:3183, 07:3184, 07:3185, 07:3186,
07:3187, 07:3262, 07:3263, 07:3264,
07:3274, 08:79, 08:80, 08:90, 08:132,
08:190, 08:196, 08:271, 08:650,
08:1085, 08:1086, 08:1087, 08:1088,
08:1108, 08:1109, 08:1110, 08:1111,
08:1135, 08:1191, 08:1211, 08:1269,
08:1290, 08:1291, 08:1302, 08:1396
- cs\check@icr commands:
- `\cs_gset_eq:NN` 76
- `\csc` 38:21
- `\csname` 1334
- `\csname\endcsname` 1113
- `\cup` 30:384
- `\CurrentFile` 50:938, 52:3,
52:67, 52:84, 52:166, 52:171, 52:174,
52:377, 52:381, 52:555, 52:556, 1116
- `\CurrentFilePath` 52:3, 52:67,
52:83, 52:165, 52:378, 52:381, 1107
- `\CurrentFilePathUsed` 52:3, 52:66,
52:81, 52:163, 52:375, 52:378, 1101
- `\CurrentFileUsed`
50:938, 52:3, 52:66, 52:82, 52:164,
52:375, 52:377, 52:554, 52:555, 1101
- `\currentgrouplevel` 50:780
- `\currentgrouptype` . 39:148, 39:151, 39:153
- `\CurrentOption`
..... 21:1558, 21:1561, 21:1566,
21:1582, 50:14, 50:520, 50:531,
50:544, 50:545, 50:550, 50:563,
50:564, 50:566, 50:580, 50:581,
50:584, 50:598, 50:603, 50:604,
50:617, 50:622, 50:623, 50:636,
50:638, 50:644, 50:646, 50:658,
50:659, 50:660, 50:668, 50:669,
50:670, 50:988, 50:1053, 50:1172,
50:1173, 50:1183, 51:85, 51:86, 1287
- CurrentOption commands:
- `\CurrentOption:` 50:543,
50:562, 50:579, 50:597, 50:602,
50:616, 50:621, 50:657, 50:667, 50:1182
- `\currsubencoding` 33:1212,
33:1223, 33:1379, 33:1381, 33:1382
- `\CYRA` 21:1530
- `\cyra` 21:1530, 21:1585
- `\CYRABHCH` 21:1530
- `\cyrahch` 21:1530
- `\CYRABHCHDSC` 21:1530
- `\cyrahchdsc` 21:1530

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpara.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

\CYRABHDZE	21:1531	\CYRH	21:1537
\cyrabhdze	21:1530	\cyrh	21:1537
\CYRABHHA	21:1531	\CYRHDSC	21:1538
\cyrabhha	21:1531	\cyrhds	21:1538
\CYRAE	21:1531	\CYRHHCRS	21:1538
\cyrae	21:1531	\cyrhcrs	21:1538
\CYRB	21:1531	\CYRHHK	21:1538
\cyrb	21:1531	\cyrhk	21:1538
\CYRBYUS	21:1532	\CYRHRDSN	21:1539
\cyrbyus	21:1531	\cyhrdsn	21:1538
\CYRC	21:1532	\CYRI	21:1539
\cyrc	21:1532	\cyri	21:1539
\CYRCH	21:1532	\CYRIE	21:1539
\cyrch	21:1532	\cyrie	21:1539
\CYRCHLDSC	21:1532	\CYRII	21:1539
\cyrchldsc	21:1532	\cyrii	21:1539
\CYRCHRDSC	21:1533	\CYRISHRT	21:1539
\cyrchrdsc	21:1532	\cyrishrt	21:1539
\CYRCHVCRS	21:1533	\CYRISHRTDSC	21:1540
\cyrchvcrs	21:1533	\cyrishrtdsc	21:1540
\CYRD	21:1533	\CYRIZH	21:1540
\cyrd	21:1533	\cyrizh	21:1540
\CYRDELTA	21:1533	\CYRJE	21:1540
\cyrdelta	21:1533	\cyrje	21:1540
\CYRDJE	21:1534	\CYRK	21:1540
\cyrdje	21:1534	\cyrk	21:1540
\CYRDZE	21:1534	\CYRKBEAK	21:1541
\cyrdze	21:1534	\cyrkbeak	21:1541
\CYRDZHE	21:1534	\CYRKDSC	21:1541
\cyrdzhe	21:1534	\cyrkds	21:1541
\CYRE	21:1534	\CYRKHCRS	21:1541
\cyre	21:1534	\cyrkcrs	21:1541
\CYREPS	21:1535	\CYRKHK	21:1542
\cyreps	21:1534	\cyrkhk	21:1541
\CYREREV	21:1535	\CYRKVCRS	21:1542
\cyrerev	21:1535	\cyrkvcrs	21:1542
\CYRERY	21:1535	\CYRL	21:1542
\cyrery	21:1535	\cyrl	21:1542
\CYRF	21:1535	\CYRLDSC	21:1542
\cyrf	21:1535	\cyrldsc	21:1542
\CYRFITA	21:1536	\CYRLHK	21:1543
\cyrfita	21:1535	\cyrlhk	21:1542
\CYRG	21:1536	\CYRLJE	21:1543
\cyrg	21:1536	\cyrlje	21:1543
\CYRGDSC	21:1536	\CYRM	21:1543
\cyrgdsc	21:1536	\cyrm	21:1543
\CYRGDSCHCRS	21:1536	\CYRMDSC	21:1543
\cyrgdschcrs	21:1536	\cyrmdsc	21:1543
\CYRGHCRS	21:1537	\CYRMHK	21:1543
\cyrghcrs	21:1537	\cyrmhk	21:1543
\CYRGHK	21:1537	\CYRN	21:1544
\cyrghk	21:1537	\cyrn	21:1544
\CYRGUP	21:1537	\CYRNDSC	21:1544
\cyrgup	21:1537	\cyrndsc	21:1544

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=ltthyphen.dtx, 57=ltfinal.dtx

<code>\CYRNG</code>	21:1544	<code>\CYRU</code>	21:1551
<code>\cyrng</code>	21:1544	<code>\cyrus</code>	21:1551
<code>\CYRNHK</code>	21:1544	<code>\CYRUSHRT</code>	21:1551
<code>\cyrnhk</code>	21:1544	<code>\cyrushrt</code>	21:1551
<code>\CYRNJE</code>	21:1545	<code>\CYRV</code>	21:1551
<code>\cyrnje</code>	21:1544	<code>\cyrv</code>	21:1551
<code>\CYRNLHK</code>	21:1545	<code>\CYRW</code>	21:1551
<code>\cyrnlhk</code>	21:1545	<code>\cyrw</code>	21:1551
<code>\CYRO</code>	21:1545	<code>\CYRY</code>	21:1551
<code>\cyro</code>	21:1545	<code>\cyrz</code>	21:1551
<code>\CYROTLD</code>	21:1545	<code>\CYRYA</code>	21:1552
<code>\cyrotld</code>	21:1545	<code>\cyrza</code>	21:1552
<code>\CYRP</code>	21:1545	<code>\CYRYAT</code>	21:1552
<code>\cyrp</code>	21:1545	<code>\cyrzat</code>	21:1552
<code>\CYRPHK</code>	21:1546	<code>\CYRYHCRS</code>	21:1552
<code>\cyrphk</code>	21:1546	<code>\cyrzhcrs</code>	21:1552
<code>\CYRQ</code>	21:1546	<code>\CYRYI</code>	21:1552
<code>\cyrq</code>	21:1546	<code>\cyrzi</code>	21:1552
<code>\CYRR</code>	21:1546	<code>\CYRYO</code>	21:1553
<code>\cyr</code>	21:1546	<code>\cyrzo</code>	21:1552
<code>\CYRRDSC</code>	21:1546	<code>\CYRYU</code>	21:1553
<code>\cyrddsc</code>	21:1546	<code>\cyrzu</code>	21:1553
<code>\CYRRHK</code>	21:1547, 1325	<code>\CYRZ</code>	21:1553
<code>\cyrhkh</code>	21:1546, 1325	<code>\cyrz</code>	21:1553
<code>\CYRRHOOK</code>	1325	<code>\CYRZDSC</code>	21:1553
<code>\cyrhook</code>	1325	<code>\cyrzdsc</code>	21:1553
<code>\CYRRTICK</code>	21:1547	<code>\CYRZH</code>	21:1553
<code>\cyrtrick</code>	21:1547	<code>\cyrzh</code>	21:1553
<code>\CYRS</code>	21:1547	<code>\CYRZHDSC</code>	21:1554
<code>\cyr</code>	21:1547	<code>\cyrzhdsc</code>	21:1554
<code>\CYRSACRS</code>	21:1547		
<code>\cyr</code>	21:1547		
<code>\CYRSCHWA</code>	21:1548		
<code>\cyr</code>	21:1548		
<code>\CYRSDSC</code>	21:1548		
<code>\cyr</code>	21:1548		
<code>\CYRSEMISFTSN</code>	21:1548		
<code>\cyr</code>	21:1548		
<code>\CYRSFTSN</code>	21:1549		
<code>\cyr</code>	21:1549		
<code>\CYRSH</code>	21:1549		
<code>\cyr</code>	21:1549		
<code>\CYRSHCH</code>	21:1549		
<code>\cyr</code>	21:1549		
<code>\CYRSHHA</code>	21:1549		
<code>\cyr</code>	21:1549		
<code>\CYRT</code>	21:1550		
<code>\cyr</code>	21:1550		
<code>\CYRTDSC</code>	21:1550		
<code>\cyr</code>	21:1550		
<code>\CYRTETSE</code>	21:1550		
<code>\cyr</code>	21:1550		
<code>\CYRTSHE</code>	21:1550		
<code>\cyr</code>	21:1550		

D

<code>\d</code>	21:255, 21:415, 21:503, 21:787, 21:1271, 21:1485, 21:1486, 21:1487, 21:1488, 21:1491, 21:1492, 21:1493, 21:1494, 21:1495, 21:1496, 21:1497, 21:1498, 21:1499, 21:1500, 21:1501, 21:1502, 21:1503, 21:1504, 21:1505, 21:1506, 21:1507, 21:1508, 21:1509, 21:1510, 21:1511, 21:1512, 21:1513, 21:1514, 21:1515, 21:1516, 21:1517, 21:1518, 21:1519, 21:1520, 21:1521, 21:1522, 21:1523, 21:1524, 1300
<code>\dag</code>	21:332, 1305
<code>\dagger</code>	21:332, 22:218, 22:224, 22:232, 22:233, 30:386
<code>\dashbox</code>	42:309, 42:807, 42:824
<code>\dashv</code>	30:414
<code>\date</code>	958
<code>\date</code>	44:9, 44:25
<code>\day</code>	03:11, 01:169, 50:1298, 50:1431, 50:1520, 02:384
<code>\dblfigrule</code>	54:1041, 54:2949, 1296

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

<code>\dblfloatpagefraction</code>	33:387, 33:388, 33:389, 33:390,
..... 45:287, 45:301, 54:2926	33:391, 33:392, 33:393, 33:394,
<code>\dblfloatsep</code>	33:395, 33:396, 33:397, 33:398,
54:1039, 54:2140, 54:2266, 54:2933	33:399, 33:400, 33:401, 33:402,
<code>\dbltextfloatsep</code>	33:403, 33:404, 33:405, 33:406,
54:1043, 54:2139, 54:2265, 54:2933	33:407, 33:408, 33:409, 33:410,
<code>\dbltopfraction</code> ..	33:411, 33:412, 33:413, 33:414,
45:284, 45:298, 54:2925	33:415, 33:416, 33:417, 33:418,
<code>\ddag</code>	21:333, 1305
<code>\ddagger</code>	21:333,
22:219, 22:225, 22:232, 22:234, 30:385	33:419, 33:420, 33:421, 33:422,
<code>\ddot</code>	30:529
<code>\ddots</code>	30:524
<code>\deadcycles</code> 20:335, 20:391, 20:430, 37:33,	33:423, 33:424, 33:425, 33:426,
37:98, 37:174, 37:240, 54:297, 1321	33:427, 33:428, 33:429, 33:430,
debug commands:	33:431, 33:432, 33:433, 33:434,
<code>\debug_resume:</code>	33:435, 33:436, 33:437, 33:438,
.. 11:102, 11:118, 11:126, 11:134, 229	33:439, 33:440, 33:441, 33:442,
<code>\debug_suspend:</code>	33:443, 33:444, 33:445, 33:446,
... 11:98, 11:106, 11:122, 11:130, 229	33:447, 33:448, 33:449, 33:450,
<code>\DebugHooksOff</code>	08:2815, 08:2923, 211
<code>\DebugHooksOn</code>	08:2815, 08:2922, 310
<code>\DebugMarksOff</code>	48:342, 1008
<code>\DebugMarksOn</code>	48:342, 1008
<code>\DebugShipoutsOff</code> ..	53:414, 53:446, 1129
<code>\DebugShipoutsOn</code> ...	53:414, 53:445, 1129
<code>\DebugSocketsOff</code>	10:182, 10:212, 343
<code>\DebugSocketsOn</code>	10:182, 10:211, 343
<code>\DebugTablesOff</code>	55:178
<code>\DebugTablesOn</code>	55:178
declare commands:	33:439, 33:440, 33:441, 33:442,
<code>declare_callback_rule</code>	04:864
<code>\declare_callback_rule</code>	46
<code>\DeclareAccent</code>	1289
<code>\DeclareCaseChangeEquivalent</code> 57:571, 1349	33:463, 33:464, 33:465, 33:466,
<code>\DeclareCommandCopy</code>	33:467, 33:468, 33:469, 33:470,
..... 06:472, 06:473, 06:475, 1339	33:471, 33:472, 33:473, 33:474,
<code>\DeclareCurrentRelease</code> ..	33:475, 33:476, 33:477, 33:478,
33:777, 50:1763	33:479, 33:480, 33:481, 33:482,
<code>\DeclareDefaultHookRule</code>	33:483, 33:484, 33:485, 33:486,
..... 08:2819, 08:2926, 209	33:487, 33:488, 33:489, 33:490,
<code>\DeclareDocumentCommand</code>	33:491, 33:492, 33:493, 33:494,
..... 44:186, 44:187, 07:3071, 89	33:495, 33:496, 33:497, 33:498,
<code>\DeclareDocumentEnvironment</code> ...	07:3107
<code>\DeclareEmphSequence</code>	713
<code>\DeclareEmphSequence</code>	713
.. 29:542, 29:578, 29:579, 29:591, 714	33:499, 33:500, 33:501, 33:502,
<code>\DeclareEncoding</code>	1289
<code>\DeclareEncodingSubset</code>	24:186, 33:361, 33:362,
..... 24:186, 33:361, 33:362,	33:363, 33:364, 33:365, 33:366,
33:367, 33:368, 33:369, 33:370,	33:371, 33:372, 33:373, 33:374,
33:375, 33:376, 33:377, 33:378,	33:379, 33:380, 33:381, 33:382,
33:383, 33:384, 33:385, 33:386,	33:1161, 33:1162, 33:1163, 33:1164,

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpara.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=ltthyphen.dtx, 57=ltfinal.dtx

	33:1165, 33:1166, 33:1167, 33:1168,	25:160, 25:161, 25:162, 25:163,
	33:1169, 33:1170, 33:1171, 33:1172, 540	25:164, 25:165, 25:166, 25:167,
\DeclareEncodingsubset	787	25:168, 25:169, 25:170, 25:171,
\DeclareEnvironmentCopy		25:172, 25:173, 25:174, 25:175,
	06:513, 06:514, 06:516, 1351	25:176, 25:177, 25:178, 25:179,
\DeclareErrorFont		25:180, 25:181, 25:182, 25:183,
	24:574, 28:399, 29:736, 30:61	25:184, 25:185, 25:186, 25:187,
\DeclareExpandableDocumentCommand		25:188, 25:189, 25:190, 25:191,
	44:184, 44:185, 07:3190	25:192, 25:193, 25:194, 25:195,
\DeclareFixedFont		25:196, 25:197, 25:198, 25:199,
	24:76, 33:1213, 33:1215, 788	25:200, 25:201, 25:202, 25:203,
\DeclareFontEncoding		25:204, 25:205, 25:206, 25:207,
	21:397, 21:484, 21:738, 21:760,	25:208, 25:209, 25:210, 25:211,
	21:766, 21:852, 21:1060, 24:119,	25:212, 25:213, 25:214, 25:215,
	30:137, 30:138, 30:139, 30:140, 1290	25:216, 25:217, 25:218, 25:219,
\DeclareFontEncodingDefaults		25:220, 25:221, 25:222, 25:223,
	24:169, 27:90, 27:91, 30:36	25:224, 25:225, 25:226, 25:227,
\DeclareFontFamily	24:94, 27:85, 27:86, 552	25:228, 25:229, 25:230, 25:231,
\DeclareFontFamilySubstitution	24:534	25:232, 25:233, 25:234, 25:235,
\DeclareFontSeriesChangeRule		25:236, 25:237, 25:238, 25:239,
	25:4, 25:5, 25:7, 25:8, 25:9,	25:240, 25:241, 25:242, 25:243,
	25:10, 25:11, 25:12, 25:13, 25:14,	25:244, 25:245, 25:246, 25:247,
	25:15, 25:16, 25:17, 25:18, 25:19,	25:248, 25:249, 25:250, 25:251,
	25:20, 25:21, 25:22, 25:23, 25:24,	25:252, 25:253, 25:254, 25:255,
	25:25, 25:26, 25:27, 25:28, 25:29,	25:256, 25:257, 25:258, 25:259,
	25:30, 25:31, 25:32, 25:33, 25:34,	25:260, 25:261, 25:262, 25:263,
	25:35, 25:36, 25:37, 25:38, 25:39,	25:264, 25:265, 25:266, 25:267,
	25:40, 25:41, 25:42, 25:43, 25:44,	25:268, 25:269, 25:270, 25:271,
	25:45, 25:46, 25:47, 25:48, 25:49,	25:272, 25:273, 25:274, 25:275,
	25:50, 25:51, 25:52, 25:53, 25:54,	25:276, 25:277, 25:278, 25:279,
	25:55, 25:56, 25:57, 25:58, 25:59,	25:280, 25:281, 25:282, 25:283,
	25:60, 25:61, 25:62, 25:63, 25:64,	25:284, 25:285, 25:286, 25:287,
	25:65, 25:66, 25:67, 25:68, 25:69,	25:288, 25:289, 25:290, 25:291,
	25:70, 25:71, 25:72, 25:73, 25:74,	25:292, 25:293, 25:294, 25:295,
	25:75, 25:76, 25:77, 25:78, 25:79,	25:296, 25:297, 25:298, 25:299,
	25:80, 25:81, 25:82, 25:83, 25:84,	25:300, 25:301, 25:302, 25:303,
	25:85, 25:86, 25:87, 25:88, 25:89,	25:304, 25:305, 25:306, 25:307,
	25:90, 25:91, 25:92, 25:93, 25:94,	25:308, 25:309, 25:310, 25:311,
	25:95, 25:96, 25:97, 25:98, 25:99,	25:312, 25:313, 25:314, 25:315,
	25:100, 25:101, 25:102, 25:103,	25:316, 25:317, 25:318, 25:319,
	25:104, 25:105, 25:106, 25:107,	25:320, 25:321, 25:322, 25:323,
	25:108, 25:109, 25:110, 25:111,	25:324, 25:325, 25:326, 25:327,
	25:112, 25:113, 25:114, 25:115,	25:328, 25:329, 25:330, 25:331,
	25:116, 25:117, 25:118, 25:119,	25:332, 25:333, 25:334, 25:335,
	25:120, 25:121, 25:122, 25:123,	25:336, 25:337, 25:338, 25:339,
	25:124, 25:125, 25:126, 25:127,	25:340, 25:341, 25:342, 25:343,
	25:128, 25:129, 25:130, 25:131,	25:344, 25:345, 25:346, 25:347,
	25:132, 25:133, 25:134, 25:135,	25:348, 25:349, 25:350, 25:351,
	25:136, 25:137, 25:138, 25:139,	25:352, 25:353, 25:354, 25:355,
	25:140, 25:141, 25:142, 25:143,	25:356, 25:357, 25:358, 25:359,
	25:144, 25:145, 25:146, 25:147,	25:360, 25:361, 25:362, 25:363,
	25:148, 25:149, 25:150, 25:151,	25:364, 25:365, 25:366, 25:367,
	25:152, 25:153, 25:154, 25:155,	25:368, 25:369, 25:370, 25:371,
	25:156, 25:157, 25:158, 25:159,	25:372, 25:373, 25:374, 25:375,

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpara.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=ltthyphen.dtx, 57=ltfinal.dtx

25:376, 25:377, 25:378, 25:379, 25:592, 25:593, 25:594, 25:595,
 25:380, 25:381, 25:382, 25:383, 25:596, 25:597, 25:598, 25:599,
 25:384, 25:385, 25:386, 25:387, 25:600, 25:601, 25:602, 25:603,
 25:388, 25:389, 25:390, 25:391, 25:604, 25:605, 25:606, 25:607,
 25:392, 25:393, 25:394, 25:395, 25:608, 25:609, 25:610, 25:611,
 25:396, 25:397, 25:398, 25:399, 25:612, 25:613, 25:614, 25:615,
 25:400, 25:401, 25:402, 25:403, 25:616, 25:617, 25:618, 25:619,
 25:404, 25:405, 25:406, 25:407, 25:620, 25:621, 25:622, 25:623,
 25:408, 25:409, 25:410, 25:411, 25:624, 25:625, 25:626, 25:627,
 25:412, 25:413, 25:414, 25:415, 25:628, 25:629, 25:630, 25:631,
 25:416, 25:417, 25:418, 25:419, 25:632, 25:633, 25:634, 25:635,
 25:420, 25:421, 25:422, 25:423, 25:636, 25:637, 25:638, 25:639,
 25:424, 25:425, 25:426, 25:427, 25:640, 25:641, 25:642, 25:643,
 25:428, 25:429, 25:430, 25:431, 25:644, 25:645, 25:646, 25:647,
 25:432, 25:433, 25:434, 25:435, 25:648, 25:649, 25:650, 25:651,
 25:436, 25:437, 25:438, 25:439, 25:652, 25:653, 25:654, 25:655,
 25:440, 25:441, 25:442, 25:443, 25:656, 25:657, 25:658, 25:659,
 25:444, 25:445, 25:446, 25:447, 25:660, 25:661, 25:662, 25:663,
 25:448, 25:449, 25:450, 25:451, 25:664, 25:665, 25:666, 25:667,
 25:452, 25:453, 25:454, 25:455, 25:668, 25:669, 25:670, 25:671,
 25:456, 25:457, 25:458, 25:459, 25:672, 25:673, 25:674, 25:675,
 25:460, 25:461, 25:462, 25:463, 25:676, 25:677, 25:678, 25:679,
 25:464, 25:465, 25:466, 25:467, 25:680, 25:681, 25:682, 25:683,
 25:468, 25:469, 25:470, 25:471, 25:684, 25:685, 25:686, 25:687,
 25:472, 25:473, 25:474, 25:475, 25:688, 25:689, 25:690, 25:691,
 25:476, 25:477, 25:478, 25:479, 25:692, 25:693, 25:694, 25:695,
 25:480, 25:481, 25:482, 25:483, 25:696, 25:697, 25:698, 25:699,
 25:484, 25:485, 25:486, 25:487, 25:700, 25:701, 25:702, 25:703,
 25:488, 25:489, 25:490, 25:491, 25:704, 25:705, 25:706, 25:707,
 25:492, 25:493, 25:494, 25:495, 25:708, 25:709, 25:710, 25:711,
 25:496, 25:497, 25:498, 25:499, 25:712, 25:713, 25:714, 25:715,
 25:500, 25:501, 25:502, 25:503, 25:716, 25:717, 25:718, 25:719,
 25:504, 25:505, 25:506, 25:507, 25:720, 25:721, 25:722, 25:723,
 25:508, 25:509, 25:510, 25:511, 25:724, 25:725, 25:726, 25:727,
 25:512, 25:513, 25:514, 25:515, 25:728, 25:729, 25:730, 25:731,
 25:516, 25:517, 25:518, 25:519, 25:732, 25:733, 25:734, 25:735,
 25:520, 25:521, 25:522, 25:523, 25:736, 25:737, 25:738, 25:739,
 25:524, 25:525, 25:526, 25:527, 25:740, 25:741, 25:742, 25:743,
 25:528, 25:529, 25:530, 25:531, 25:744, 25:745, 25:746, 25:747,
 25:532, 25:533, 25:534, 25:535, 25:748, 25:749, 25:750, 25:751,
 25:536, 25:537, 25:538, 25:539, 25:752, 25:753, 25:754, 25:755,
 25:540, 25:541, 25:542, 25:543, 25:756, 25:757, 25:758, 25:759,
 25:544, 25:545, 25:546, 25:547, 25:760, 25:761, 25:762, 25:763,
 25:548, 25:549, 25:550, 25:551, 25:764, 25:765, 25:766, 25:767,
 25:552, 25:553, 25:554, 25:555, 25:768, 25:769, 25:770, 25:771,
 25:556, 25:557, 25:558, 25:559, 25:772, 25:773, 25:774, 25:775,
 25:560, 25:561, 25:562, 25:563, 25:776, 25:777, 25:778, 25:779,
 25:564, 25:565, 25:566, 25:567, 25:780, 25:781, 25:782, 25:783,
 25:568, 25:569, 25:570, 25:571, 25:784, 25:785, 25:786, 25:787,
 25:572, 25:573, 25:574, 25:575, 25:788, 25:789, 25:790, 25:791,
 25:576, 25:577, 25:578, 25:579, 25:792, 25:793, 25:794, 25:795,
 25:580, 25:581, 25:582, 25:583, 25:796, 25:797, 25:798, 25:799,
 25:584, 25:585, 25:586, 25:587, 25:800, 25:801, 25:802, 25:803,
 25:588, 25:589, 25:590, 25:591, 25:804, 25:805, 25:806, 25:807,

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
 06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpara.dtx,
 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
 56=ltthyphen.dtx, 57=ltfinal.dtx

25:808, 25:809, 25:810, 25:811, 25:1024, 25:1025, 25:1026, 25:1027,
 25:812, 25:813, 25:814, 25:815, 25:1028, 25:1029, 25:1030, 25:1031,
 25:816, 25:817, 25:818, 25:819, 25:1032, 25:1033, 25:1034, 25:1035,
 25:820, 25:821, 25:822, 25:823, 25:1036, 25:1037, 25:1038, 25:1039,
 25:824, 25:825, 25:826, 25:827, 25:1040, 25:1041, 25:1042, 25:1043,
 25:828, 25:829, 25:830, 25:831, 25:1044, 25:1045, 25:1046, 25:1047,
 25:832, 25:833, 25:834, 25:835, 25:1048, 25:1049, 25:1050, 25:1051,
 25:836, 25:837, 25:838, 25:839, 25:1052, 25:1053, 25:1054, 25:1055,
 25:840, 25:841, 25:842, 25:843, 25:1056, 25:1057, 25:1058, 25:1059,
 25:844, 25:845, 25:846, 25:847, 25:1060, 25:1061, 25:1062, 25:1063,
 25:848, 25:849, 25:850, 25:851, 25:1064, 25:1065, 25:1066, 25:1067,
 25:852, 25:853, 25:854, 25:855, 25:1068, 25:1069, 25:1070, 25:1071,
 25:856, 25:857, 25:858, 25:859, 25:1072, 25:1073, 25:1074, 25:1075,
 25:860, 25:861, 25:862, 25:863, 25:1076, 25:1077, 25:1078, 25:1079,
 25:864, 25:865, 25:866, 25:867, 25:1080, 25:1081, 25:1082, 25:1083,
 25:868, 25:869, 25:870, 25:871, 25:1084, 25:1085, 25:1086, 25:1087,
 25:872, 25:873, 25:874, 25:875, 25:1088, 25:1089, 25:1090, 25:1091,
 25:876, 25:877, 25:878, 25:879, 25:1092, 25:1093, 25:1094, 25:1095,
 25:880, 25:881, 25:882, 25:883, 25:1096, 25:1097, 25:1098, 25:1099,
 25:884, 25:885, 25:886, 25:887, 25:1100, 25:1101, 25:1102, 25:1103,
 25:888, 25:889, 25:890, 25:891, 25:1104, 25:1105, 25:1106, 25:1107,
 25:892, 25:893, 25:894, 25:895, 25:1108, 25:1109, 25:1110, 25:1111,
 25:896, 25:897, 25:898, 25:899, 25:1112, 25:1113, 25:1114, 25:1115,
 25:900, 25:901, 25:902, 25:903, 25:1116, 25:1117, 25:1118, 25:1119,
 25:904, 25:905, 25:906, 25:907, 25:1120, 25:1121, 25:1122, 25:1123,
 25:908, 25:909, 25:910, 25:911, 25:1124, 25:1125, 25:1126, 25:1127,
 25:912, 25:913, 25:914, 25:915, 25:1128, 25:1129, 25:1130, 25:1131,
 25:916, 25:917, 25:918, 25:919, 25:1132, 25:1133, 25:1134, 25:1135,
 25:920, 25:921, 25:922, 25:923, 25:1136, 25:1137, 25:1138, 25:1139,
 25:924, 25:925, 25:926, 25:927, 25:1140, 25:1141, 25:1142, 25:1143,
 25:928, 25:929, 25:930, 25:931, 25:1144, 25:1145, 25:1146, 25:1147,
 25:932, 25:933, 25:934, 25:935, 25:1148, 25:1149, 25:1150, 25:1151,
 25:936, 25:937, 25:938, 25:939, 25:1152, 25:1153, 25:1154, 25:1155,
 25:940, 25:941, 25:942, 25:943, 25:1156, 25:1157, 25:1158, 25:1159,
 25:944, 25:945, 25:946, 25:947, 25:1160, 25:1161, 25:1162, 25:1163,
 25:948, 25:949, 25:950, 25:951, 25:1164, 25:1165, 25:1166, 25:1167,
 25:952, 25:953, 25:954, 25:955, 25:1168, 25:1169, 25:1170, 25:1171,
 25:956, 25:957, 25:958, 25:959, 25:1172, 25:1173, 25:1174, 25:1175,
 25:960, 25:961, 25:962, 25:963, 25:1176, 25:1177, 25:1178, 25:1179,
 25:964, 25:965, 25:966, 25:967, 25:1180, 25:1181, 25:1182, 25:1183,
 25:968, 25:969, 25:970, 25:971, 25:1184, 25:1185, 25:1186, 25:1187,
 25:972, 25:973, 25:974, 25:975, 25:1188, 25:1189, 25:1190, 25:1191,
 25:976, 25:977, 25:978, 25:979, 25:1192, 25:1193, 25:1194, 25:1195,
 25:980, 25:981, 25:982, 25:983, 25:1196, 25:1197, 25:1198, 25:1199,
 25:984, 25:985, 25:986, 25:987, 25:1200, 25:1201, 25:1202, 25:1203,
 25:988, 25:989, 25:990, 25:991, 25:1204, 25:1205, 25:1206, 25:1207,
 25:992, 25:993, 25:994, 25:995, 25:1208, 25:1209, 25:1210, 25:1211,
 25:996, 25:997, 25:998, 25:999, 25:1212, 25:1213, 25:1214, 25:1215,
 25:1000, 25:1001, 25:1002, 25:1003, 25:1216, 25:1217, 25:1218, 25:1219,
 25:1004, 25:1005, 25:1006, 25:1007, 25:1220, 25:1221, 25:1222, 25:1223,
 25:1008, 25:1009, 25:1010, 25:1011, 25:1224, 25:1225, 25:1226, 25:1227,
 25:1012, 25:1013, 25:1014, 25:1015, 25:1228, 25:1229, 25:1230, 25:1231,
 25:1016, 25:1017, 25:1018, 25:1019, 25:1232, 25:1233, 25:1234, 25:1235,
 25:1020, 25:1021, 25:1022, 25:1023, 25:1236, 25:1237, 25:1238, 25:1239,

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
 06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
 11=lttemplates.dtx, 12=ltaloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpara.dtx,
 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
 56=ltthyphen.dtx, 57=ltfinal.dtx

25:1240, 25:1241, 25:1242, 25:1243,
 25:1244, 25:1245, 25:1246, 25:1247,
 25:1248, 25:1249, 25:1250, 25:1251,
 25:1252, 25:1253, 25:1254, 25:1255,
 25:1256, 25:1257, 25:1258, 25:1259,
 25:1260, 25:1261, 25:1262, 25:1263,
 25:1264, 25:1265, 25:1266, 25:1267,
 25:1268, 25:1269, 25:1270, 25:1271,
 25:1272, 25:1273, 25:1274, 25:1275,
 25:1276, 25:1277, 25:1278, 25:1279,
 25:1280, 25:1281, 25:1282, 25:1283,
 25:1284, 25:1285, 25:1286, 25:1287,
 25:1288, 25:1289, 25:1290, 25:1291,
 25:1292, 25:1293, 25:1294, 25:1295,
 25:1296, 25:1297, 25:1298, 25:1299,
 25:1300, 25:1301, 25:1302, 25:1303,
 25:1304, 25:1305, 25:1306, 25:1307,
 25:1308, 25:1309, 25:1310, 25:1311,
 25:1312, 25:1313, 25:1314, 25:1315,
 25:1316, 25:1317, 25:1318, 25:1319,
 25:1320, 25:1321, 25:1322, 25:1323,
 25:1324, 25:1325, 25:1326, 25:1327,
 25:1328, 25:1329, 25:1330, 25:1331,
 25:1332, 25:1333, 25:1334, 25:1335,
 25:1336, 25:1337, 25:1338, 25:1339,
 25:1340, 25:1341, 25:1342, 25:1343,
 25:1344, 25:1345, 25:1346, 25:1347,
 25:1348, 25:1349, 25:1350, 25:1351,
 25:1352, 25:1353, 25:1354, 25:1355,
 25:1356, 25:1357, 25:1358, 25:1359,
 25:1360, 25:1361, 25:1362, 25:1363,
 25:1364, 25:1365, 25:1366, 25:1367,
 25:1368, 25:1369, 25:1370, 25:1371,
 25:1372, 25:1373, 25:1374, 25:1375,
 25:1376, 25:1377, 25:1378, 25:1379,
 25:1380, 25:1381, 25:1382, 25:1383,
 25:1384, 25:1385, 25:1386, 25:1387,
 25:1388, 25:1389, 25:1390, 25:1391,
 25:1392, 25:1393, 25:1394, 25:1395,
 25:1396, 25:1397, 25:1398, 25:1399,
 25:1400, 25:1401, 25:1402, 25:1403,
 25:1404, 25:1405, 25:1406, 25:1407,
 25:1408, 25:1409, 25:1410, 25:1411,
 25:1412, 25:1413, 25:1414, 25:1415,
 25:1416, 25:1417, 25:1418, 25:1419,
 25:1420, 25:1421, 25:1422, 25:1423,
 25:1424, 25:1425, 25:1426, 25:1427,
 25:1428, 25:1429, 25:1433, 25:1435,
 25:1438, 25:1439, 25:1440, 25:1441,
 25:1442, 25:1443, 25:1444, 25:1445,
 25:1446, 25:1447, 25:1448, 25:1449,
 25:1450, 25:1451, 25:1452, 25:1453,
 25:1454, 25:1455, 25:1456, 25:1457,
 25:1458, 25:1459, 25:1460, 25:1461,
 25:1462, 25:1463, 25:1464, 25:1465,
 25:1466, 25:1467, 25:1468, 25:1469,
 25:1470, 25:1471, 25:1472, 25:1473,
 25:1474, 25:1475, 25:1476, 25:1477,
 25:1478, 25:1479, 25:1480, 25:1481,
 25:1482, 25:1483, 25:1484, 25:1485,
 25:1486, 25:1487, 25:1488, 25:1489,
 25:1490, 25:1491, 25:1492, 25:1493,
 25:1494, 25:1495, 25:1496, 25:1497,
 25:1498, 25:1499, 25:1500, 25:1501,
 25:1502, 25:1503, 25:1504, 25:1505,
 25:1506, 25:1507, 25:1508, 25:1509,
 25:1510, 25:1511, 25:1512, 25:1513,
 25:1514, 25:1515, 25:1516, 25:1517,
 25:1518, 25:1519, 25:1520, 25:1521,
 25:1522, 25:1523, 25:1524, 25:1525,
 25:1526, 25:1527, 25:1528, 25:1529,
 25:1530, 25:1531, 25:1532, 25:1533,
 25:1534, 25:1535, 25:1536, 25:1537,
 25:1538, 25:1539, 25:1540, 25:1541,
 25:1542, 25:1543, 25:1544, 25:1545,
 25:1546, 25:1547, 25:1548, 25:1549,
 25:1550, 25:1551, 25:1552, 25:1553,
 25:1554, 25:1555, 25:1556, 25:1557,
 25:1558, 25:1559, 25:1560, 25:1561,
 25:1562, 25:1563, 25:1564, 25:1565,
 25:1566, 25:1567, 25:1568, 25:1569,
 25:1570, 25:1571, 25:1572, 25:1573,
 25:1574, 25:1575, 25:1576, 25:1577,
 25:1578, 25:1579, 25:1580, 25:1581,
 25:1582, 25:1583, 25:1584, 25:1585,
 25:1586, 25:1587, 25:1588, 25:1589,
 25:1590, 25:1591, 25:1592, 25:1593,
 25:1594, 25:1595, 25:1596, 25:1597,
 25:1598, 25:1599, 25:1600, 25:1601,
 25:1602, 25:1603, 25:1604, 25:1605,
 25:1606, 25:1607, 25:1608, 25:1609,
 25:1610, 25:1611, 25:1612, 25:1613,
 25:1614, 25:1615, 25:1616, 25:1617,
 25:1618, 25:1619, 25:1620, 25:1621,
 25:1622, 25:1623, 25:1624, 25:1625,
 25:1626, 25:1627, 25:1628, 25:1629,
 25:1630, 25:1631, 25:1632, 25:1633,
 25:1634, 25:1635, 25:1636, 25:1637,
 25:1638, 25:1639, 25:1640, 25:1641,
 25:1642, 25:1643, 25:1644, 25:1645,
 25:1646, 25:1647, 25:1648, 25:1649,
 25:1650, 25:1651, 25:1652, 25:1653,
 25:1654, 25:1655, 25:1656, 25:1657,
 25:1658, 25:1659, 25:1660, 25:1661,
 25:1662, 25:1663, 25:1664, 25:1665,
 25:1666, 25:1667, 25:1668, 25:1669,
 25:1670, 25:1671, 25:1672, 25:1673,
 25:1674, 25:1675, 25:1676, 25:1677,

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
 06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpara.dtx,
 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
 56=ltthyphen.dtx, 57=ltfinal.dtx

25:1678, 25:1679, 25:1680, 25:1681,
 25:1682, 25:1683, 25:1684, 25:1685,
 25:1686, 25:1687, 25:1688, 25:1689,
 25:1690, 25:1691, 25:1692, 25:1693,
 25:1694, 25:1695, 25:1696, 25:1697,
 25:1698, 25:1699, 25:1700, 25:1701,
 25:1702, 25:1703, 25:1704, 25:1705,
 25:1706, 25:1707, 25:1708, 25:1709,
 25:1710, 25:1711, 25:1712, 25:1713,
 25:1714, 25:1715, 25:1716, 25:1717,
 25:1718, 25:1719, 25:1720, 25:1721,
 25:1722, 25:1723, 25:1724, 25:1725,
 25:1726, 25:1727, 25:1728, 25:1729,
 25:1730, 25:1731, 25:1732, 25:1733,
 25:1734, 25:1735, 25:1736, 25:1737,
 25:1738, 25:1739, 25:1740, 25:1741,
 25:1742, 25:1743, 25:1744, 25:1745,
 25:1746, 25:1747, 25:1748, 25:1749,
 25:1750, 25:1751, 25:1752, 25:1753,
 25:1754, 25:1755, 25:1756, 25:1757,
 25:1758, 25:1759, 25:1760, 25:1761,
 25:1762, 25:1763, 25:1764, 25:1765,
 25:1766, 25:1767, 25:1768, 25:1769,
 25:1770, 25:1771, 25:1772, 25:1773,
 25:1774, 25:1775, 25:1776, 25:1777,
 25:1778, 25:1779, 25:1780, 25:1781,
 25:1782, 25:1783, 25:1784, 25:1785,
 25:1786, 25:1787, 25:1788, 25:1789,
 25:1790, 25:1791, 25:1792, 25:1793,
 25:1794, 25:1795, 25:1796, 25:1797,
 25:1798, 25:1799, 25:1800, 25:1801,
 25:1802, 25:1803, 25:1804, 25:1805,
 25:1806, 25:1807, 25:1808, 25:1809,
 25:1810, 25:1811, 25:1812, 25:1813,
 25:1814, 25:1815, 25:1816, 25:1817,
 25:1818, 25:1819, 25:1820, 25:1821,
 25:1822, 25:1823, 25:1824, 25:1825,
 25:1826, 25:1827, 25:1828, 25:1829,
 25:1830, 25:1831, 25:1832, 25:1833,
 25:1834, 25:1835, 25:1836, 25:1837,
 25:1838, 25:1839, 25:1840, 25:1841,
 25:1842, 25:1843, 25:1844, 25:1845,
 25:1846, 25:1847, 25:1848, 25:1849,
 25:1850, 25:1851, 25:1852, 25:1853,
 25:1854, 25:1855, 25:1856, 25:1857,
 25:1858, 25:1859, 25:1860, 25:1861,
 25:1862, 25:1863, 25:1864, 25:1865,
 25:1866, 25:1867, 25:1868, 25:1869,
 25:1870, 25:1871, 25:1872, 25:1873,
 25:1874, 25:1875, 25:1876, 25:1877,
 25:1878, 25:1879, 25:1880, 25:1881,
 25:1882, 25:1883, 25:1884, 25:1885,
 25:1886, 25:1887, 25:1888, 25:1889,
 25:1890, 25:1891, 25:1892, 25:1893,
 25:1894, 25:1895, 25:1896, 25:1897,
 25:1898, 25:1899, 25:1900, 25:1901,
 25:1902, 25:1903, 25:1904, 25:1905,
 25:1906, 25:1907, 25:1908, 25:1909,
 25:1910, 25:1911, 25:1912, 25:1913,
 25:1914, 25:1915, 25:1916, 25:1917,
 25:1918, 25:1919, 25:1920, 25:1921,
 25:1922, 25:1923, 25:1924, 25:1925,
 25:1926, 25:1927, 25:1928, 25:1929,
 25:1930, 25:1931, 25:1932, 25:1933,
 25:1934, 25:1935, 25:1936, 25:1937,
 25:1938, 25:1939, 25:1940, 25:1941,
 25:1942, 25:1943, 25:1944, 25:1945,
 25:1946, 25:1947, 25:1948, 25:1949,
 25:1950, 25:1951, 25:1952, 25:1953,
 25:1954, 25:1955, 25:1956, 25:1957,
 25:1958, 25:1959, 25:1960, 25:1961,
 25:1962, 25:1963, 25:1964, 25:1965,
 25:1966, 25:1967, 25:1968, 25:1969,
 25:1970, 25:1971, 25:1972, 25:1973,
 25:1974, 25:1975, 25:1976, 25:1977,
 25:1978, 25:1979, 25:1980, 25:1981,
 25:1982, 25:1983, 25:1984, 25:1985,
 25:1986, 25:1987, 25:1988, 25:1989,
 25:1990, 25:1991, 25:1992, 25:1993,
 25:1994, 25:1995, 25:1996, 25:1997,
 25:1998, 25:1999, 25:2000, 25:2001,
 25:2002, 25:2003, 25:2004, 25:2005,
 25:2006, 25:2007, 25:2008, 25:2009,
 25:2010, 25:2011, 25:2012, 25:2013,
 25:2014, 25:2015, 25:2016, 25:2017,
 25:2018, 25:2019, 25:2020, 25:2021,
 25:2022, 25:2023, 25:2024, 25:2025,
 25:2026, 25:2027, 25:2028, 25:2029,
 25:2030, 25:2031, 25:2032, 25:2033,
 25:2034, 25:2035, 25:2036, 25:2037,
 25:2038, 25:2039, 25:2040, 25:2041,
 25:2042, 25:2043, 25:2044, 25:2045,
 25:2046, 25:2047, 25:2048, 25:2049,
 25:2050, 25:2051, 25:2052, 25:2053,
 25:2054, 25:2055, 25:2056, 25:2057,
 25:2058, 25:2059, 25:2060, 25:2061,
 25:2062, 25:2063, 25:2064, 25:2065,
 25:2066, 25:2067, 25:2068, 25:2069,
 25:2070, 25:2071, 25:2072, 25:2073,
 25:2074, 25:2075, 25:2076, 25:2077,
 25:2078, 25:2079, 25:2080, 25:2081,
 25:2082, 25:2083, 25:2084, 25:2085,
 25:2086, 25:2087, 25:2088, 25:2089,
 25:2090, 25:2091, 25:2092, 25:2093,
 25:2094, 25:2095, 25:2096, 25:2097,
 25:2098, 25:2099, 25:2100, 25:2101,
 25:2102, 25:2103, 25:2104, 25:2105,
 25:2106, 25:2107, 25:2108, 25:2109,

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
 06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpara.dtx,
 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
 56=ltthyphen.dtx, 57=ltfinal.dtx

25:2110, 25:2111, 25:2112, 25:2113,
 25:2114, 25:2115, 25:2116, 25:2117,
 25:2118, 25:2119, 25:2120, 25:2121,
 25:2122, 25:2123, 25:2124, 25:2125,
 25:2126, 25:2127, 25:2128, 25:2129,
 25:2130, 25:2131, 25:2132, 25:2133,
 25:2134, 25:2135, 25:2136, 25:2137,
 25:2138, 25:2139, 25:2140, 25:2141,
 25:2142, 25:2143, 25:2144, 25:2145,
 25:2146, 25:2147, 25:2148, 25:2149,
 25:2150, 25:2151, 25:2152, 25:2153,
 25:2154, 25:2155, 25:2156, 25:2157,
 25:2158, 25:2159, 25:2160, 25:2161,
 25:2162, 25:2163, 25:2164, 25:2165,
 25:2166, 25:2167, 25:2168, 25:2169,
 25:2170, 25:2171, 25:2172, 25:2173,
 25:2174, 25:2175, 25:2176, 25:2177,
 25:2178, 25:2179, 25:2180, 25:2181,
 25:2182, 25:2183, 25:2184, 25:2185,
 25:2186, 25:2187, 25:2188, 25:2189,
 25:2190, 25:2191, 25:2192, 25:2193,
 25:2194, 25:2195, 25:2196, 25:2197,
 25:2198, 25:2199, 25:2200, 25:2201,
 25:2202, 25:2203, 25:2204, 25:2205,
 25:2206, 25:2207, 25:2208, 25:2209,
 25:2210, 25:2211, 25:2212, 25:2213,
 25:2214, 25:2215, 25:2216, 25:2217,
 25:2218, 25:2219, 25:2220, 25:2221,
 25:2222, 25:2223, 25:2224, 25:2225,
 25:2226, 25:2227, 25:2228, 25:2229,
 25:2230, 25:2231, 25:2232, 25:2233,
 25:2234, 25:2235, 25:2236, 25:2237,
 25:2238, 25:2239, 25:2240, 25:2241,
 25:2242, 25:2243, 25:2244, 25:2245,
 25:2246, 25:2247, 25:2248, 25:2249,
 25:2250, 25:2251, 25:2252, 25:2253,
 25:2254, 25:2255, 25:2256, 25:2257,
 25:2258, 25:2259, 25:2260, 25:2261,
 25:2262, 25:2263, 25:2264, 25:2265,
 25:2266, 25:2267, 25:2268, 25:2269,
 25:2270, 25:2271, 25:2272, 25:2273,
 25:2274, 25:2275, 25:2276, 25:2277,
 25:2278, 25:2279, 25:2280, 25:2281,
 25:2282, 25:2283, 25:2284, 25:2285,
 25:2286, 25:2287, 25:2288, 25:2289,
 25:2290, 25:2291, 25:2292, 25:2293,
 25:2294, 25:2295, 25:2296, 25:2297,
 25:2298, 25:2299, 25:2300, 25:2301,
 25:2302, 25:2303, 25:2304, 25:2305,
 25:2306, 25:2307, 25:2308, 25:2309,
 25:2310, 25:2311, 25:2312, 25:2313,
 25:2314, 25:2315, 25:2316, 25:2317,
 25:2318, 25:2319, 25:2320, 25:2321,
 25:2322, 25:2323, 25:2324, 25:2325,
 25:2326, 25:2327, 25:2328, 25:2329,
 25:2330, 25:2331, 25:2332, 25:2333,
 25:2334, 25:2335, 25:2336, 25:2337,
 25:2338, 25:2339, 25:2340, 25:2341,
 25:2342, 25:2343, 25:2344, 25:2345,
 25:2346, 25:2347, 25:2348, 25:2349,
 25:2350, 25:2351, 25:2352, 25:2353,
 25:2354, 25:2355, 25:2356, 25:2357,
 25:2358, 25:2359, 25:2360, 25:2361,
 25:2362, 25:2363, 25:2364, 25:2365,
 25:2366, 25:2367, 25:2368, 25:2369,
 25:2370, 25:2371, 25:2372, 25:2373,
 25:2374, 25:2375, 25:2376, 25:2377,
 25:2378, 25:2379, 25:2380, 25:2381,
 25:2382, 25:2383, 25:2384, 25:2385,
 25:2386, 25:2387, 25:2388, 25:2389,
 25:2390, 25:2391, 25:2392, 25:2393,
 25:2394, 25:2395, 25:2396, 25:2397,
 25:2398, 25:2399, 25:2400, 25:2401,
 25:2402, 25:2403, 25:2404, 25:2405,
 25:2406, 25:2407, 25:2408, 25:2409,
 25:2410, 25:2411, 25:2412, 25:2413,
 25:2414, 25:2415, 25:2416, 25:2417,
 25:2418, 25:2419, 25:2420, 25:2421,
 25:2422, 25:2423, 25:2424, 25:2425,
 25:2426, 25:2427, 25:2428, 25:2429,
 25:2430, 25:2431, 25:2432, 25:2433,
 25:2434, 25:2435, 25:2436, 25:2437,
 25:2438, 25:2439, 25:2440, 25:2441,
 25:2442, 25:2443, 25:2444, 25:2445,
 25:2446, 25:2447, 25:2448, 25:2449,
 25:2450, 25:2451, 25:2452, 25:2453,
 25:2454, 25:2455, 25:2456, 25:2457,
 25:2458, 25:2459, 25:2460, 25:2461,
 25:2462, 25:2463, 25:2464, 25:2465,
 25:2466, 25:2467, 25:2468, 25:2469,
 25:2470, 25:2471, 25:2472, 25:2473,
 25:2474, 25:2475, 25:2476, 25:2477,
 25:2478, 25:2479, 25:2480, 25:2481,
 25:2482, 25:2483, 25:2484, 25:2485,
 25:2486, 25:2487, 25:2488, 25:2489,
 25:2490, 25:2491, 25:2492, 25:2493,
 25:2494, 25:2495, 25:2496, 25:2497,
 25:2498, 25:2499, 25:2500, 25:2501,
 25:2502, 25:2503, 25:2504, 25:2505,
 25:2506, 25:2507, 25:2508, 25:2509,
 25:2510, 25:2511, 25:2512, 25:2513,
 25:2514, 25:2515, 25:2516, 25:2517,
 25:2518, 25:2519, 25:2520, 25:2521,
 25:2522, 25:2523, 25:2524, 25:2525,
 25:2526, 25:2527, 25:2528, 25:2529,
 25:2530, 25:2531, 25:2532, 25:2533,
 25:2534, 25:2535, 25:2536, 25:2537,
 25:2538, 25:2539, 25:2540, 25:2541,

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
 06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpara.dtx,
 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
 56=ltthyphen.dtx, 57=ltfinal.dtx

- 25:2542, 25:2543, 25:2544, 25:2545,
 25:2546, 25:2547, 25:2548, 25:2549,
 25:2550, 25:2551, 25:2552, 25:2553,
 25:2554, 25:2555, 25:2556, 25:2557,
 25:2558, 25:2559, 25:2560, 25:2561,
 25:2562, 25:2563, 25:2564, 25:2565,
 25:2566, 25:2567, 25:2568, 25:2569,
 25:2570, 25:2571, 25:2572, 25:2573,
 25:2574, 25:2575, 25:2576, 25:2577,
 25:2578, 25:2579, 25:2580, 25:2581,
 25:2582, 25:2583, 25:2584, 25:2585,
 25:2586, 25:2587, 25:2588, 25:2589,
 25:2590, 25:2591, 25:2592, 25:2593,
 25:2594, 25:2595, 25:2596, 25:2597,
 25:2598, 25:2599, 25:2600, 25:2601,
 25:2602, 25:2603, 25:2604, 25:2605,
 25:2606, 25:2607, 25:2608, 25:2609,
 25:2610, 25:2611, 25:2612, 25:2613,
 25:2614, 25:2615, 25:2616, 25:2617,
 25:2618, 25:2619, 25:2620, 25:2621,
 25:2622, 25:2623, 25:2624, 25:2625,
 25:2626, 25:2627, 25:2628, 25:2629,
 25:2630, 25:2631, 25:2632, 25:2633,
 25:2634, 25:2635, 25:2636, 25:2637,
 25:2638, 25:2639, 25:2640, 25:2641,
 25:2642, 25:2643, 25:2644, 25:2645,
 25:2646, 25:2647, 25:2648, 25:2649,
 25:2650, 25:2651, 25:2652, 25:2653,
 25:2654, 25:2655, 25:2656, 25:2657,
 25:2658, 25:2659, 25:2660, 25:2661,
 25:2662, 25:2663, 25:2664, 25:2665,
 25:2666, 25:2667, 25:2668, 25:2669,
 25:2670, 25:2671, 25:2672, 25:2673,
 25:2674, 25:2675, 25:2676, 25:2677,
 25:2678, 25:2679, 25:2680, 25:2681,
 25:2682, 25:2683, 25:2684, 25:2685,
 25:2686, 25:2687, 25:2688, 25:2689,
 25:2690, 25:2691, 25:2692, 25:2693,
 25:2694, 25:2695, 25:2696, 25:2697,
 25:2698, 25:2699, 25:2700, 25:2701,
 25:2702, 25:2703, 25:2704, 25:2705,
 25:2706, 25:2707, 25:2708, 25:2709,
 25:2710, 25:2711, 25:2712, 25:2713,
 25:2714, 25:2715, 25:2716, 25:2717,
 25:2718, 25:2719, 25:2720, 25:2721,
 25:2722, 25:2723, 25:2724, 25:2725,
 25:2726, 25:2727, 25:2728, 25:2729,
 25:2730, 25:2731, 25:2732, 25:2733,
 25:2734, 25:2735, 25:2736, 25:2737,
 25:2738, 25:2739, 25:2740, 25:2741,
 25:2742, 25:2743, 25:2744, 25:2745,
 25:2746, 25:2747, 25:2748, 25:2749,
 25:2750, 25:2751, 25:2752, 25:2753,
 25:2754, 25:2755, 25:2756, 25:2757,
 25:2758, 25:2759, 25:2760, 25:2761,
 25:2762, 25:2763, 25:2764, 25:2765,
 25:2766, 25:2767, 25:2768, 25:2769,
 25:2770, 25:2774, 25:2776, 1358
- `\DeclareFontSeriesDefault` 696
- `\DeclareFontSeriesDefault`
 29:35, 29:36, 29:70, 29:72,
 29:73, 29:83, 29:94, 29:103, 29:105, 707
- `\DeclareFontShape` 24:19,
 24:543, 24:544, 24:545, 24:546,
 24:547, 24:548, 24:549, 24:550,
 24:551, 24:552, 24:553, 24:554,
 24:555, 24:556, 24:557, 24:558,
 24:559, 24:560, 24:561, 24:562,
 24:563, 27:25, 27:27, 27:81, 27:82, 1283
- `\DeclareFontShapeChangeRule` 25:2912,
 25:2934, 25:2947, 25:2949, 25:2950,
 25:2951, 25:2952, 25:2953, 25:2954,
 25:2955, 25:2956, 25:2957, 25:2958,
 25:2959, 25:2960, 25:2961, 25:2962,
 25:2963, 25:2964, 25:2965, 25:2966,
 25:2967, 25:2968, 25:2969, 25:2970,
 25:2971, 25:2972, 25:2973, 25:2974,
 25:2975, 25:2976, 25:2977, 25:2978,
 25:2979, 25:2980, 25:2981, 25:2982,
 25:2983, 25:2984, 25:2985, 25:2986,
 25:2987, 25:2988, 25:2989, 25:2990,
 25:2991, 25:2992, 25:2993, 25:2994,
 25:2995, 25:2996, 25:2997, 25:2998,
 25:2999, 25:3000, 25:3001, 25:3002,
 25:3003, 25:3004, 25:3005, 25:3006,
 25:3007, 25:3008, 25:3009, 25:3010,
 25:3011, 25:3012, 25:3013, 25:3014,
 25:3015, 25:3016, 25:3017, 25:3018,
 25:3019, 25:3020, 25:3021, 25:3022,
 25:3023, 25:3024, 25:3025, 25:3026,
 25:3027, 25:3028, 25:3029, 25:3030,
 25:3031, 25:3032, 25:3033, 25:3034,
 25:3035, 25:3036, 25:3037, 25:3038,
 25:3039, 25:3040, 25:3041, 25:3042,
 25:3043, 25:3044, 25:3045, 25:3049,
 25:3051, 25:3052, 25:3053, 25:3054,
 25:3055, 25:3056, 25:3057, 25:3058,
 25:3059, 25:3060, 25:3061, 25:3062,
 25:3063, 25:3064, 25:3065, 25:3066,
 25:3067, 25:3068, 25:3069, 25:3070,
 25:3071, 25:3072, 25:3073, 25:3074,
 25:3075, 25:3076, 25:3077, 25:3078,
 25:3079, 25:3080, 25:3081, 25:3082,
 25:3083, 25:3084, 25:3085, 25:3086,
 25:3087, 25:3088, 25:3090, 25:3091,
 25:3092, 25:3093, 25:3095, 25:3096,
 25:3097, 25:3099, 25:3100, 25:3101,
 25:3102, 25:3103, 25:3104, 25:3106,

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
 06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
 11=lttemplates.dtx, 12=ltaloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpara.dtx,
 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
 56=ltthyphen.dtx, 57=ltfinal.dtx

25:3107, 25:3108, 25:3109, 25:3110,	30:225, 30:226, 30:227, 30:228,
25:3111, 25:3112, 25:3114, 25:3115,	30:229, 30:230, 30:231, 30:232,
25:3116, 25:3117, 25:3118, 25:3119,	30:233, 30:234, 30:235, 30:236,
25:3121, 25:3122, 25:3123, 25:3124,	30:237, 30:238, 30:239, 30:240,
25:3125, 25:3126, 25:3127, 25:3130	30:241, 30:242, 30:243, 30:244,
<code>\DeclareFontSubstitution</code>	30:245, 30:246, 30:247, 30:248,
21:767, 21:853,	30:249, 30:250, 30:251, 30:252,
24:142, 30:26, 30:34, 30:37, 30:38,	30:253, 30:254, 30:255, 30:256,
30:39, 30:141, 30:142, 30:143, 30:144	30:257, 30:258, 30:259, 30:260,
<code>\DeclareHookRule</code>	30:261, 30:262, 30:272, 30:273,
08:2817, 08:2925,	30:275, 30:279, 30:280, 30:281,
37:50, 37:51, 37:115, 37:116, 208	30:282, 30:283, 30:284, 30:285,
<code>\DeclareInstance</code>	30:286, 30:287, 30:288, 30:289,
11:1163, 358	30:290, 30:291, 30:292, 30:293,
<code>\DeclareInstanceCopy</code>	30:294, 30:295, 30:296, 30:297,
11:1163, 357	30:298, 30:299, 30:300, 30:301,
<code>\DeclareKeys</code>	30:302, 30:303, 30:304, 30:305,
51:187, 1091	30:306, 30:307, 30:308, 30:309,
<code>\DeclareLowercaseMapping</code>	30:310, 30:311, 30:312, 30:313,
57:571, 1353	30:314, 30:315, 30:316, 30:317,
<code>\DeclareMathAccent</code>	30:318, 30:319, 30:320, 30:321,
28:804,	30:322, 30:323, 30:324, 30:325,
30:527, 30:528, 30:529, 30:530,	30:326, 30:327, 30:328, 30:329,
30:531, 30:532, 30:533, 30:534,	30:330, 30:331, 30:332, 30:333,
30:535, 30:536, 30:537, 30:538, 30:539	30:334, 30:335, 30:336, 30:338,
<code>\DeclareMathAlphabet</code>	30:339, 30:340, 30:341, 30:342,
27:119,	30:343, 30:344, 30:345, 30:352,
27:123, 27:125, 27:132, 28:630,	30:353, 30:354, 30:355, 30:356,
28:793, 30:162, 30:163, 30:164, 30:165	30:357, 30:358, 30:360, 30:361,
<code>\DeclareMathAlphabetCharacter</code>	30:362, 30:363, 30:364, 30:365,
28:979	30:367, 30:368, 30:369, 30:370,
<code>\DeclareMathDelimiter</code>	30:371, 30:372, 30:375, 30:376,
28:981, 30:266, 30:267,	30:377, 30:378, 30:381, 30:382,
30:268, 30:269, 30:270, 30:271,	30:383, 30:384, 30:385, 30:386,
30:274, 30:276, 30:277, 30:574,	30:387, 30:388, 30:389, 30:390,
30:576, 30:578, 30:580, 30:582,	30:391, 30:392, 30:393, 30:394,
30:585, 30:587, 30:589, 30:591,	30:395, 30:396, 30:397, 30:398,
30:593, 30:595, 30:597, 30:599,	30:399, 30:400, 30:401, 30:402,
30:601, 30:603, 30:605, 30:607,	30:403, 30:404, 30:405, 30:406,
30:609, 30:611, 30:613, 30:615,	30:407, 30:408, 30:409, 30:410,
30:617, 30:619, 30:621, 30:623, 1321	30:411, 30:412, 30:413, 30:414,
<code>\DeclareMathOperator</code>	30:415, 30:416, 30:417, 30:418,
1309	30:419, 30:420, 30:421, 30:422,
<code>\DeclareMathRadical</code>	30:425, 30:426, 30:429, 30:430,
28:1116, 30:540	30:431, 30:432, 30:433, 30:434,
<code>\DeclareMathSizes</code>	30:435, 30:436, 30:437, 30:438,
24:268, 24:274, 24:296,	30:439, 30:440, 30:441, 30:443,
30:168, 30:169, 30:170, 30:171,	30:444, 30:445, 30:446, 30:447,
30:172, 30:173, 30:174, 30:175,	30:448, 30:449, 30:452, 30:453,
30:176, 30:177, 30:178, 30:179, 1307	30:454, 30:456, 30:457, 30:458,
<code>\DeclareMathSizes*</code>	30:459, 30:460, 30:461, 30:462,
24:268	30:463, 30:464, 30:465, 30:466,
<code>\DeclareMathSymbol</code>	30:488, 30:490, 30:512, 30:513,
28:917, 28:980, 28:997, 30:180,	30:514, 30:564, 30:565, 30:566,
30:181, 30:182, 30:183, 30:184,	
30:185, 30:186, 30:187, 30:188,	
30:189, 30:190, 30:191, 30:192,	
30:193, 30:194, 30:195, 30:196,	
30:197, 30:198, 30:199, 30:200,	
30:201, 30:202, 30:203, 30:204,	
30:205, 30:206, 30:207, 30:208,	
30:209, 30:210, 30:211, 30:212,	
30:213, 30:214, 30:215, 30:216,	
30:217, 30:218, 30:219, 30:220,	
30:221, 30:222, 30:223, 30:224,	

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpara.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=ltthyphen.dtx, 57=ltfinal.dtx

- 30:567, 30:625, 30:626, 30:627, 1321
- `\DeclareMathVersion` . . . 28:412, 29:3, 29:4
- `\DeclareOldFontCommand` . . . 32:125, 32:141
- `\DeclareOption` 1041
- `\DeclareOption*` 1041
- 21:1557, 26:30, 26:38, 26:46,
26:54, 26:57, 26:61, 33:782, 33:783,
33:784, 33:785, 33:786, 33:788,
33:790, 33:792, 33:793, 33:833,
33:834, 33:835, 33:836, 33:837,
33:839, 33:840, 50:496, 50:1193, 1311
- `\DeclareOption*` 1041
- `\DeclareOption*` 50:496
- `\DeclarePreloadSizes`
- 24:248, 27:95, 27:96,
31:19, 31:21, 31:22, 31:23, 31:25,
31:26, 31:27, 31:28, 31:29, 31:30,
31:34, 31:38, 31:43, 31:45, 31:49,
31:50, 31:53, 31:54, 31:57, 31:58, 31:64
- `\DeclareProtectedCommand` 1298
- `\DeclareRelease` . . . 33:775, 33:776, 50:1669
- DeclareRelease commands:
- `\DeclareRelease:` 50:1672
- `\DeclareRobustCommand`
- 14:4, 14:11, 14:30,
14:57, 18:7, 18:8, 18:9, 18:10, 18:11,
18:69, 18:93, 18:377, 18:453, 18:467,
18:512, 18:537, 19:2, 19:3, 19:13,
20:469, 20:616, 21:168, 21:176,
21:327, 21:330, 21:331, 21:332,
21:333, 21:334, 21:336, 21:338,
21:340, 22:242, 23:32, 23:33, 23:34,
24:314, 24:342, 24:343, 24:344,
24:349, 24:361, 24:371, 24:379,
24:381, 24:399, 24:738, 24:747,
25:2782, 25:2786, 25:2795, 25:2797,
25:2807, 25:2914, 25:2919, 25:2924,
25:3135, 25:3138, 25:3146, 25:3147,
25:3153, 25:3233, 26:116, 26:165,
29:5, 29:8, 29:11, 29:14, 29:17,
29:20, 29:23, 29:26, 29:29, 29:255,
29:278, 29:308, 29:329, 29:353,
29:364, 29:381, 29:384, 29:406,
29:411, 29:416, 29:449, 29:454,
29:459, 29:475, 29:478, 29:481,
29:484, 29:487, 29:501, 29:550,
29:551, 29:586, 29:592, 29:614,
29:616, 29:625, 29:627, 29:634,
29:650, 29:658, 29:676, 29:692,
29:709, 30:346, 30:347, 30:348,
30:359, 30:366, 30:423, 30:424,
30:455, 30:467, 30:471, 30:474,
30:479, 30:481, 30:483, 30:486,
30:489, 30:491, 30:492, 30:494,
30:496, 30:498, 30:500, 30:502,
30:504, 30:506, 30:508, 30:510,
30:516, 30:518, 30:520, 30:523,
30:541, 30:544, 30:547, 30:551,
30:555, 30:558, 30:561, 30:568,
30:571, 30:628, 30:629, 30:630,
30:635, 30:637, 30:639, 30:641,
32:3, 32:126, 33:4, 33:11, 33:574,
33:1097, 35:186, 35:197, 37:269,
37:450, 37:455, 37:460, 37:470,
37:474, 37:478, 37:576, 37:580,
38:3, 38:4, 38:5, 38:6, 38:7, 38:8,
38:9, 38:10, 38:11, 38:12, 38:13,
38:14, 38:15, 38:16, 38:17, 38:18,
38:19, 38:20, 38:21, 38:22, 38:23,
38:24, 38:25, 38:26, 38:27, 38:28,
38:29, 38:30, 38:31, 38:32, 38:33,
38:34, 38:35, 38:39, 38:41, 38:42,
38:43, 38:44, 38:45, 38:46, 38:47,
38:48, 38:49, 38:50, 38:51, 38:52,
38:81, 38:82, 38:83, 38:84, 38:126,
38:167, 38:169, 38:173, 38:218,
38:220, 38:222, 38:224, 38:227,
38:228, 38:230, 38:237, 38:239,
38:240, 38:251, 38:274, 38:276,
38:292, 38:303, 38:353, 38:354,
38:355, 38:429, 38:463, 38:487,
06:233, 40:7, 40:24, 40:159, 40:172,
40:195, 40:196, 40:212, 40:223,
40:302, 40:534, 40:552, 40:560,
40:615, 40:616, 40:617, 40:618,
40:619, 40:620, 41:139, 41:142,
41:154, 42:124, 42:127, 42:130,
44:7, 44:8, 44:9, 44:10, 44:14,
44:247, 45:402, 45:423, 47:16, 47:28,
48:506, 48:507, 49:23, 49:28, 49:38,
49:49, 49:64, 49:72, 49:126, 49:128,
49:130, 49:137, 50:1128, 50:1129,
50:1130, 50:1136, 50:1137, 50:1793,
52:148, 52:184, 06:833, 06:834,
06:840, 06:855, 53:448, 54:83, 1298
- `\DeclareSizeFunction` 26:418,
26:491, 26:492, 26:503, 26:504,
26:508, 26:509, 26:515, 26:516,
26:544, 26:558, 26:559, 26:566, 26:567
- `\DeclareSymbolFont` 27:136,
28:471, 30:152, 30:153, 30:154, 30:155
- `\DeclareSymbolFontAlphabet`
- 28:1190, 30:159, 30:160, 30:161
- `\DeclareTemplateCode`
- 11:1001, 11:1042, 11:1068, 11:1163, 352
- `\DeclareTemplateCopy` 11:1163, 355
- `\DeclareTemplateInterface` . . . 11:1163, 352
- `\DeclareTextAccent` 21:82,

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=ltthyphen.dtx, 57=ltfinal.dtx

21:398, 21:399, 21:400, 21:401, 33:208, 33:210, 33:212, 33:214,
 21:402, 21:403, 21:404, 21:405, 33:216, 33:218, 33:220, 33:222,
 21:406, 21:407, 21:408, 21:485, 33:224, 33:226, 33:228, 33:230,
 21:486, 21:487, 21:488, 21:489, 33:232, 33:234, 33:236, 33:238,
 21:490, 21:491, 21:492, 21:493, 33:240, 33:242, 33:244, 33:246,
 21:494, 21:495, 21:763, 21:768, 33:248, 33:250, 33:252, 33:254,
 21:769, 21:770, 21:771, 21:772, 33:256, 33:258, 33:260, 33:262,
 21:773, 21:774, 21:775, 21:776, 33:264, 33:266, 33:268, 33:270,
 21:777, 21:778, 21:860, 21:861, 33:273, 33:275, 33:277, 33:279,
 21:862, 21:863, 21:864, 21:865, 33:281, 33:283, 33:285, 33:287,
 21:866, 21:867, 21:868, 21:869, 33:289, 33:291, 33:293, 33:295,
 21:870, 21:871, 21:872, 21:873, 21:874 33:297, 33:299, 33:301, 33:303,
 \DeclareTextAccentDefault 33:305, 33:307, 33:309, 33:311,
 . . . 21:203, 21:246, 21:247, 21:248, 33:313, 33:315, 33:317, 33:319,
 21:249, 21:250, 21:251, 21:252, 33:321, 33:323, 33:325, 33:327,
 21:253, 21:254, 21:255, 21:256, 33:329, 33:618, 33:626, 33:628,
 21:257, 21:258, 21:259, 21:299, 33:634, 33:642, 33:967, 33:969,
 21:302, 33:651, 33:652, 33:901, 33:970, 33:972, 33:974, 33:976,
 33:902, 33:903, 33:904, 33:905, 33:978, 33:980, 33:982, 33:984,
 33:906, 33:907, 33:908, 33:909, 33:986, 33:988, 33:990, 33:992,
 33:910, 33:911, 33:912, 33:913, 1304 33:994, 33:996, 33:998, 33:1000,
 \DeclareTextCommand 21:3, 21:76, 21:83, 33:1002, 33:1004, 33:1006, 33:1008,
 21:101, 21:409, 21:412, 21:415, 33:1010, 33:1012, 33:1014, 33:1016,
 21:429, 21:430, 21:431, 21:434, 33:1018, 33:1020, 33:1022, 33:1024,
 21:435, 21:442, 21:444, 21:446, 33:1026, 33:1028, 33:1030, 33:1032,
 21:448, 21:454, 21:456, 21:458, 33:1034, 33:1036, 33:1038, 33:1040,
 21:465, 21:496, 21:499, 21:503, 33:1042, 33:1044, 33:1046, 33:1048,
 21:506, 21:508, 21:511, 21:513, 33:1050, 33:1052, 33:1054, 33:1056,
 21:515, 21:531, 21:589, 21:590, 33:1058, 33:1060, 33:1062, 33:1064,
 21:591, 21:752, 21:779, 21:781, 33:1066, 33:1068, 33:1070, 33:1072,
 21:784, 21:787, 21:820, 21:827, 33:1074, 33:1076, 33:1078, 33:1080,
 21:854, 21:857, 21:887, 21:915, 33:1082, 33:1084, 33:1087, 1304
 21:1075, 21:1102, 33:340, 33:341, \DeclareTextComposite . 21:94, 21:472,
 33:342, 33:343, 33:344, 33:345, 21:473, 21:608, 21:609, 21:610,
 33:346, 33:347, 33:348, 33:349, 21:611, 21:612, 21:613, 21:614,
 33:589, 33:596, 33:603, 33:723, 1298 21:615, 21:616, 21:617, 21:618,
 \DeclareTextCommandDefault . . . 21:75, 21:619, 21:620, 21:621, 21:622,
 21:204, 21:206, 21:303, 21:306, 21:623, 21:624, 21:625, 21:626,
 21:307, 21:308, 21:310, 21:312, 21:627, 21:628, 21:629, 21:630,
 21:316, 21:320, 21:321, 21:323, 21:631, 21:632, 21:633, 21:634,
 21:324, 21:325, 21:326, 21:346, 21:635, 21:636, 21:637, 21:638,
 21:375, 33:119, 33:121, 33:124, 21:639, 21:640, 21:641, 21:642,
 33:126, 33:128, 33:130, 33:132, 21:643, 21:644, 21:645, 21:646,
 33:134, 33:136, 33:138, 33:140, 21:647, 21:648, 21:649, 21:650,
 33:142, 33:144, 33:146, 33:148, 21:651, 21:652, 21:653, 21:654,
 33:150, 33:152, 33:154, 33:157, 21:655, 21:656, 21:657, 21:658,
 33:158, 33:159, 33:160, 33:161, 21:659, 21:660, 21:661, 21:662,
 33:162, 33:163, 33:164, 33:165, 21:663, 21:664, 21:665, 21:666,
 33:166, 33:167, 33:168, 33:169, 21:667, 21:668, 21:669, 21:670,
 33:170, 33:171, 33:172, 33:174, 21:671, 21:672, 21:673, 21:674,
 33:176, 33:178, 33:180, 33:182, 21:675, 21:676, 21:677, 21:678,
 33:184, 33:186, 33:188, 33:190, 21:679, 21:680, 21:681, 21:682,
 33:192, 33:194, 33:196, 33:198, 21:683, 21:684, 21:685, 21:686,
 33:200, 33:202, 33:204, 33:206, 21:687, 21:688, 21:689, 21:690,

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpara.dtx,
 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
 56=ltthyphen.dtx, 57=ltfinal.dtx

- 21:691, 21:692, 21:693, 21:694,
21:695, 21:696, 21:697, 21:698,
21:699, 21:700, 21:701, 21:702,
21:703, 21:704, 21:705, 21:706,
21:707, 21:708, 21:709, 21:710,
21:711, 21:712, 21:713, 21:714,
21:715, 21:716, 21:717, 21:718,
21:834, 21:835, 21:836, 21:837,
21:838, 21:839, 21:840, 21:841,
21:842, 21:843, 21:844, 21:845,
21:846, 21:847, 21:848, 21:849, *1313*
- `\DeclareTextCompositeCommand`
. *21:94*, 21:451, 21:474,
21:475, 21:476, 21:477, 21:478,
21:480, 21:719, 21:720, 21:722,
21:725, 21:726, 21:727, 21:728,
21:729, 21:730, 21:731, 21:732,
21:733, 21:735, 21:817, 21:1081, *1305*
- `\DeclareTextFontCommand` *32:1*,
32:15, 32:16, 32:17, 32:18, 32:19,
32:20, 32:21, 32:22, 32:23, 32:24,
32:29, 32:30, 32:31, 32:42, 32:140, *1322*
- `\DeclareTextFoo` *1289*
- `\DeclareTextGlyph` *1296*
- `\DeclareTextSymbol` *21:3*, 21:418,
21:419, 21:420, 21:421, 21:422,
21:423, 21:424, 21:425, 21:426,
21:427, 21:428, 21:432, 21:433,
21:436, 21:437, 21:438, 21:439,
21:440, 21:441, 21:547, 21:548,
21:549, 21:550, 21:551, 21:552,
21:553, 21:554, 21:555, 21:556,
21:557, 21:558, 21:559, 21:560,
21:561, 21:562, 21:564, 21:565,
21:566, 21:567, 21:568, 21:569,
21:570, 21:571, 21:572, 21:573,
21:574, 21:575, 21:576, 21:577,
21:578, 21:579, 21:580, 21:581,
21:582, 21:583, 21:584, 21:585,
21:586, 21:587, 21:588, 21:592,
21:593, 21:594, 21:595, 21:596,
21:597, 21:598, 21:599, 21:600,
21:601, 21:602, 21:603, 21:604,
21:605, 21:606, 21:607, 21:739,
21:740, 21:741, 21:742, 21:743,
21:744, 21:745, 21:746, 21:747,
21:748, 21:749, 21:750, 21:751,
21:761, 21:762, 21:790, 21:791,
21:792, 21:793, 21:794, 21:795,
21:796, 21:798, 21:799, 21:800,
21:801, 21:802, 21:803, 21:804,
21:805, 21:806, 21:807, 21:808,
21:809, 21:810, 21:811, 21:812,
21:813, 21:814, 21:815, 21:816,
21:875, 21:876, 21:877, 21:878,
21:879, 21:880, 21:881, 21:882,
21:883, 21:884, 21:885, 21:886,
21:898, 21:899, 21:900, 21:901,
21:902, 21:903, 21:904, 21:905,
21:906, 21:907, 21:908, 21:909,
21:910, 21:911, 21:912, 21:913,
21:914, 21:921, 21:922, 21:923,
21:924, 21:925, 21:926, 21:927,
21:928, 21:929, 21:930, 21:931,
21:932, 21:933, 21:934, 21:935,
21:936, 21:937, 21:938, 21:939,
21:940, 21:941, 21:942, 21:943,
21:944, 21:945, 21:946, 21:947,
21:948, 21:949, 21:950, 21:951,
21:952, 21:953, 21:954, 21:955,
21:956, 21:957, 21:958, 21:959,
21:960, 21:961, 21:962, 21:963,
21:964, 21:965, 21:966, 21:967,
21:968, 21:969, 21:970, 21:971,
21:972, 21:973, 21:974, 21:975,
21:976, 21:977, 21:978, 21:979,
21:980, 21:981, 21:982, 21:983,
21:984, 21:985, 21:986, 21:987,
21:988, 21:989, 21:990, 21:991,
21:992, 21:993, 21:994, 21:995,
21:996, 21:997, 21:998, 21:999,
21:1000, 21:1101, 33:332, 33:333,
33:334, 33:335, 33:336, 33:337,
33:338, 33:339, 33:350, 33:351,
33:352, 33:353, 33:354, 33:355,
33:356, 33:357, 33:358, 33:359,
33:553, 33:554, 33:555, 33:556,
33:557, 33:558, 33:559, 33:560, *1305*
- `\DeclareTextSymbolDefault`
. . . *21:203*, 21:260, 21:261, 21:262,
21:263, 21:264, 21:265, 21:266,
21:267, 21:268, 21:269, 21:270,
21:271, 21:272, 21:273, 21:274,
21:275, 21:276, 21:277, 21:278,
21:279, 21:280, 21:281, 21:282,
21:283, 21:284, 21:285, 21:286,
21:287, 21:288, 21:289, 21:290,
21:291, 21:292, 21:293, 21:294,
21:295, 21:296, 21:297, 21:298,
21:300, 21:301, 21:311, 33:78,
33:80, 33:82, 33:84, 33:85, 33:86,
33:87, 33:88, 33:89, 33:90, 33:91,
33:92, 33:93, 33:94, 33:95, 33:96,
33:97, 33:98, 33:99, 33:100, 33:101,
33:102, 33:103, 33:104, 33:105,
33:106, 33:107, 33:108, 33:109,
33:110, 33:111, 33:112, 33:113,
33:114, 33:115, 33:116, 33:117,

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpara.dtx,
16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
56=ltthyphen.dtx, 57=ltfinal.dtx

- 33:118, 33:541, 33:542, 33:543,
33:544, 33:545, 33:546, 33:547,
33:548, 33:561, 33:562, 33:563,
33:564, 33:565, 33:566, 33:567,
33:568, 33:587, 33:588, 33:606,
33:607, 33:608, 33:609, 33:610,
33:611, 33:612, 33:614, 33:914,
33:915, 33:916, 33:917, 33:918,
33:919, 33:920, 33:921, 33:922,
33:923, 33:924, 33:925, 33:926,
33:927, 33:928, 33:929, 33:930,
33:931, 33:932, 33:933, 33:934,
33:935, 33:936, 33:937, 33:938,
33:939, 33:940, 33:941, 33:942,
33:943, 33:944, 33:945, 33:946,
33:947, 33:948, 33:949, 33:950,
33:951, 33:952, 33:953, 33:954,
33:955, 33:956, 33:957, 33:958,
33:959, 33:960, 33:961, 33:962,
33:963, 33:964, 33:965, 33:966, 1304
\DeclareTitlecaseMapping . . . 57:571, 1353
\DeclareUnicode 510
\DeclareUnicodeAccent 21:1064,
21:1259, 21:1260, 21:1261, 21:1262,
21:1263, 21:1264, 21:1265, 21:1266,
21:1267, 21:1268, 21:1269, 21:1270,
21:1271, 21:1272, 21:1273, 1333
\DeclareUnicodeCharacter . . . 57:384, 1334
\DeclareUnicodeCommand
. 21:1102, 21:1103,
21:1104, 21:1105, 21:1186, 21:1188,
21:1190, 21:1237, 21:1274, 1347
\DeclareUnicodeComposite
21:1079, 21:1278, 21:1279, 21:1280,
21:1281, 21:1282, 21:1283, 21:1284,
21:1285, 21:1286, 21:1287, 21:1288,
21:1289, 21:1290, 21:1291, 21:1292,
21:1293, 21:1294, 21:1295, 21:1296,
21:1297, 21:1298, 21:1299, 21:1300,
21:1301, 21:1302, 21:1303, 21:1304,
21:1305, 21:1306, 21:1307, 21:1308,
21:1309, 21:1310, 21:1311, 21:1312,
21:1313, 21:1314, 21:1315, 21:1316,
21:1317, 21:1318, 21:1319, 21:1320,
21:1321, 21:1322, 21:1323, 21:1324,
21:1325, 21:1326, 21:1327, 21:1328,
21:1329, 21:1330, 21:1331, 21:1332,
21:1333, 21:1334, 21:1335, 21:1336,
21:1337, 21:1338, 21:1339, 21:1340,
21:1341, 21:1342, 21:1343, 21:1344,
21:1345, 21:1346, 21:1347, 21:1348,
21:1349, 21:1350, 21:1351, 21:1352,
21:1353, 21:1354, 21:1355, 21:1356,
21:1357, 21:1358, 21:1359, 21:1360,
21:1361, 21:1362, 21:1363, 21:1364,
21:1365, 21:1366, 21:1367, 21:1368,
21:1369, 21:1370, 21:1371, 21:1372,
21:1373, 21:1374, 21:1375, 21:1376,
21:1377, 21:1378, 21:1379, 21:1380,
21:1381, 21:1382, 21:1383, 21:1384,
21:1385, 21:1386, 21:1387, 21:1388,
21:1389, 21:1390, 21:1391, 21:1392,
21:1393, 21:1394, 21:1395, 21:1396,
21:1397, 21:1398, 21:1399, 21:1400,
21:1401, 21:1402, 21:1403, 21:1404,
21:1405, 21:1406, 21:1407, 21:1408,
21:1409, 21:1410, 21:1411, 21:1412,
21:1413, 21:1414, 21:1415, 21:1416,
21:1417, 21:1418, 21:1419, 21:1420,
21:1421, 21:1422, 21:1423, 21:1424,
21:1425, 21:1426, 21:1427, 21:1428,
21:1429, 21:1430, 21:1431, 21:1432,
21:1433, 21:1434, 21:1435, 21:1436,
21:1437, 21:1438, 21:1439, 21:1440,
21:1441, 21:1442, 21:1443, 21:1444,
21:1445, 21:1446, 21:1447, 21:1448,
21:1449, 21:1450, 21:1451, 21:1452,
21:1453, 21:1454, 21:1455, 21:1456,
21:1457, 21:1458, 21:1459, 21:1460,
21:1461, 21:1462, 21:1463, 21:1464,
21:1465, 21:1466, 21:1467, 21:1468,
21:1469, 21:1470, 21:1471, 21:1472,
21:1473, 21:1474, 21:1475, 21:1476,
21:1477, 21:1478, 21:1479, 21:1480,
21:1481, 21:1482, 21:1483, 21:1484,
21:1485, 21:1486, 21:1487, 21:1488,
21:1489, 21:1490, 21:1491, 21:1492,
21:1493, 21:1494, 21:1495, 21:1496,
21:1497, 21:1498, 21:1499, 21:1500,
21:1501, 21:1502, 21:1503, 21:1504,
21:1505, 21:1506, 21:1507, 21:1508,
21:1509, 21:1510, 21:1511, 21:1512,
21:1513, 21:1514, 21:1515, 21:1516,
21:1517, 21:1518, 21:1519, 21:1520,
21:1521, 21:1522, 21:1523, 21:1524, 509
\DeclareUnicodeSymbol
. 21:1101, 21:1106, 21:1107,
21:1108, 21:1109, 21:1110, 21:1111,
21:1112, 21:1113, 21:1114, 21:1115,
21:1116, 21:1117, 21:1118, 21:1119,
21:1120, 21:1121, 21:1122, 21:1123,
21:1124, 21:1125, 21:1126, 21:1128,
21:1129, 21:1130, 21:1131, 21:1132,
21:1133, 21:1134, 21:1135, 21:1136,
21:1137, 21:1138, 21:1139, 21:1140,
21:1141, 21:1142, 21:1144, 21:1145,
21:1146, 21:1147, 21:1148, 21:1149,
21:1150, 21:1151, 21:1152, 21:1153,

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpara.dtx,
16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
56=ltthyphen.dtx, 57=ltfinal.dtx

- 21:1154, 21:1155, 21:1156, 21:1157,
 21:1158, 21:1159, 21:1160, 21:1161,
 21:1162, 21:1163, 21:1164, 21:1165,
 21:1166, 21:1167, 21:1168, 21:1169,
 21:1170, 21:1171, 21:1172, 21:1173,
 21:1174, 21:1175, 21:1176, 21:1177,
 21:1178, 21:1179, 21:1180, 21:1181,
 21:1182, 21:1183, 21:1184, 21:1185,
 21:1192, 21:1193, 21:1194, 21:1195,
 21:1196, 21:1197, 21:1198, 21:1199,
 21:1200, 21:1201, 21:1202, 21:1203,
 21:1204, 21:1205, 21:1206, 21:1207,
 21:1208, 21:1209, 21:1210, 21:1211,
 21:1212, 21:1213, 21:1214, 21:1215,
 21:1216, 21:1217, 21:1218, 21:1219,
 21:1220, 21:1221, 21:1222, 21:1223,
 21:1224, 21:1225, 21:1226, 21:1227,
 21:1228, 21:1229, 21:1230, 21:1231,
 21:1232, 21:1233, 21:1235, 21:1236,
 21:1248, 21:1249, 21:1250, 21:1251,
 21:1252, 21:1253, 21:1254, 21:1255,
 21:1256, 21:1257, 21:1258, 1347
- `\DeclareUnknownKeyHandler` .. 51:189, 1092
`\DeclareUnknownKeysHandler` 1348
`\DeclareUppercaseMapping` ... 57:571, 1353
`\def` 728
`default (plug)` 55:37
`\defaultencoding` 1301
`\defaultthyphenchar` . 06:843, 06:858, 02:378
`\defaultscriptratio` . 24:721, 24:728, 1294
`\defaultscriptscriptratio`
 24:722, 24:728, 1294
`\defaultskewchar` 02:379
`\deg` 38:34
`\decode` 28:1114
`\delimiter` 28:1027, 28:1097, 28:1108, 1332
`\delimiterfactor` 02:382
`\delimitershortfall` 02:397
`\Delta` 30:309
`\delta` 30:282
`\depth` 40:32, 40:35
`\det` 38:30
`\Details` 33:1589, 33:1590
`\detokenize` 03:96, 03:97,
 03:182, 14:249, 14:250, 20:254,
 20:505, 21:1020, 21:1062, 24:190,
 25:2822, 29:564, 37:205, 06:574,
 06:592, 50:564, 50:598, 50:638,
 06:656, 50:902, 50:1269, 50:1273,
 50:1401, 50:1402, 50:1406, 1350
`\DH` 21:549, 21:1150, 57:656, 1300
`\dh` 21:559, 21:1156, 57:656, 1300
`\Diamond` 29:728
`\diamond` 30:391
- `\diamondsuit` 30:343
`\dim` 38:28
`dim` commands:
`\dim_eval:n` 11:364, 05:164
`\dim_new:N`
 . 11:33, 53:199, 53:200, 53:201, 53:202
`\dim_set:Nn`
 53:193, 53:194, 53:195, 53:196
`\dim_set_eq:NN` 48:68, 48:70
`\dim_use:N` 53:511, 53:512, 53:513
`\c_max_dim` ... 48:68, 48:70, 48:100,
 53:213, 53:239, 53:264, 53:291, 1018
`\l_tmpa_dim` 354
`\c_zero_dim`
 ... 16:102, 53:219, 53:220, 53:221,
 53:227, 53:245, 53:246, 53:247,
 53:270, 53:271, 53:272, 53:299,
 53:300, 53:301, 53:329, 53:331, 53:332
`\dimen` 1319
`\dimendef`
 04:226, 02:42, 02:43, 02:44, 02:52, 02:91
`\dimenzero` 04:226
`\dimeval` 77
`\dimeval` 05:158, 05:174, 77
`\dimexpr` 21:524, 21:540,
 21:891, 21:894, 21:1241, 21:1244, 42:13
`\directlua` 04:2, 04:14, 04:29,
 04:213, 04:231, 04:256, 04:261,
 04:265, 21:1033, 01:12, 01:15, 06:34,
 01:20, 01:23, 50:1254, 50:1385,
 06:754, 06:839, 01:28, 02:65, 57:98,
 57:100, 57:108, 57:113, 57:115,
 02:81, 02:105, 02:256, 02:353, 02:363
- `disable` commands:
`disable_callback` 04:975
`\disable_callback` 46
`\DisableGenericHook`
 08:2755, 08:2761, 08:2909, 312
`\DisableHook` 08:2860, 200
`\DiscardShipoutBox`
 .. 53:82, 53:411, 53:444, 53:500, 1125
`\discretionary`
 38:251, 06:841, 06:856, 06:866
`\displayindent` 02:403
`\displaylines` 38:203
`displaymath (env.)` 38:345
`\displaymath` 38:347
`\displaystyle` 30:543,
 30:546, 30:550, 30:552, 38:62,
 38:210, 38:372, 38:375, 38:428,
 38:470, 38:482, 38:510, 38:535, 38:538
`\displaywidowpenalty` 02:327
`\displaywidth`
 38:210, 38:371, 38:457, 38:513, 02:402

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=ltparams.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=ltthyphen.dtx, 57=ltfinal.dtx

<code>\div</code>	30:394	33:1315, 33:1316, 33:1317, 33:1318,
<code>\DJ</code>	21:550, 21:1160, 57:656, 1300	33:1319, 33:1320, 33:1321, 33:1322,
<code>\dj</code>	21:560, 21:1161, 57:656, 1300	33:1323, 33:1324, 33:1325, 33:1326,
<code>\do</code>	01:110, 13:3, 13:7, 13:16,	33:1327, 33:1328, 33:1329, 33:1330,
	13:26, 20:66, 20:69, 20:72, 20:135,	33:1331, 33:1332, 33:1333, 33:1334,
	20:138, 20:190, 20:193, 20:233,	33:1337, 33:1341, 33:1342, 33:1343,
	20:305, 20:367, 20:414, 20:587,	33:1344, 33:1345, 33:1346, 33:1347,
	20:604, 20:755, 20:761, 32:90, 06:69,	33:1348, 33:1349, 33:1350, 33:1351,
	37:541, 37:562, 37:722, 37:732,	33:1352, 33:1353, 33:1354, 33:1355,
	37:738, 37:744, 40:78, 40:97, 41:244,	33:1356, 33:1357, 33:1358, 33:1359,
	41:269, 42:104, 42:116, 42:123,	33:1360, 33:1361, 33:1362, 33:1363,
	42:203, 42:337, 42:339, 42:362,	33:1364, 33:1365, 33:1366, 33:1367,
	42:365, 42:394, 42:396, 42:417,	33:1368, 33:1369, 33:1370, 33:1371, 788
	42:422, 42:477, 42:505, 42:534,	<code>\dollar</code>
	42:730, 42:786, 45:65, 45:134,	1301
	47:36, 47:65, 47:82, 50:232, 50:249,	<code>\dospecials</code>
	50:543, 50:562, 50:579, 50:597,	01:110, 37:541,
	50:602, 50:616, 50:621, 50:657,	37:562, 37:722, 37:732, 02:13,
	50:667, 02:13, 02:14, 50:1182,	50:1301, 50:1434, 50:1523, 07:1605,
	50:1219, 50:1301, 50:1354, 50:1434,	07:1815, 07:2182, 07:2195, 01:58, 1319
	50:1523, 50:1808, 07:1604, 07:1814,	<code>\dot</code>
	07:2181, 07:2194, 01:58, 01:59, 1351	30:536
<code>\DocInput</code>	26:8, 30:5, 31:5, 56:4	<code>\doteq</code>
<code>\doclearpage</code>	1326	30:480
<code>\document</code>	451	<code>\dotfill</code>
<code>document (env.)</code>	37:8	06:881, 06:902, 02:546
<code>\document</code>		<code>\dots</code>
	16:143, 20:9, 37:258, 47:64, 47:81, 224	21:340, 21:342, 1305
<code>\documentclass</code>	04:16, 26:2, 30:2,	<code>\doublehyphendemerits</code>
	31:2, 50:674, 50:681, 50:740, 50:743,	02:334
	50:975, 50:1070, 50:1202, 56:2, 303	<code>\doublerulesep</code>
<code>\DocumentMetadata</code>		41:311, 41:338, 41:362
	.. 17:6, 17:11, 17:15, 17:18, 17:61, 427	<code>\Downarrow</code>
<code>\documentstyle</code>	50:679, 50:1202	30:597
<code>\doesglyphexist</code>	33:1182,	<code>\downarrow</code>
	33:1208, 33:1226, 33:1227, 33:1228,	30:591
	33:1229, 33:1230, 33:1231, 33:1232,	<code>\downbracefill</code>
	33:1235, 33:1236, 33:1237, 33:1238,	30:549, 30:568
	33:1241, 33:1242, 33:1243, 33:1244,	<code>\dump</code>
	33:1247, 33:1248, 33:1251, 33:1252,	57:720
	33:1255, 33:1256, 33:1257, 33:1258,	
	33:1259, 33:1260, 33:1261, 33:1262,	
	33:1265, 33:1266, 33:1269, 33:1270,	
	33:1271, 33:1272, 33:1273, 33:1274,	
	33:1275, 33:1276, 33:1277, 33:1278,	
	33:1279, 33:1280, 33:1281, 33:1282,	
	33:1283, 33:1284, 33:1285, 33:1286,	
	33:1287, 33:1288, 33:1289, 33:1290,	
	33:1291, 33:1292, 33:1293, 33:1294,	
	33:1295, 33:1296, 33:1297, 33:1298,	
	33:1299, 33:1300, 33:1301, 33:1302,	
	33:1303, 33:1304, 33:1305, 33:1306,	
	33:1307, 33:1308, 33:1309, 33:1310,	
	33:1311, 33:1312, 33:1313, 33:1314,	
		E
		<code>\E</code>
		50:1308,
		50:1311, 50:1339, 50:1441, 50:1444,
		50:1471, 50:1530, 50:1533, 50:1561
		<code>\edef</code>
		29
		<code>\EditInstance</code>
		11:1163, 358
		<code>\EditTemplateDefaults</code>
		11:1163, 358
		<code>\egroup</code>
		02:450, 1295
		<code>\eject</code>
		02:513
		<code>\ell</code>
		30:322
		<code>\else</code>
		1292
		else commands:
		<code>\else:</code>
		08:2269, 08:2297,
		08:2439, 08:2455, 08:2502, 08:2537,
		08:2552, 09:364, 16:26, 16:32,
		16:64, 06:721, 52:241, 07:1337,
		07:1822, 07:1823, 07:1824, 07:1825,
		08:768, 08:1042, 08:1199, 08:1261,
		08:1402, 08:1409, 08:1417, 08:1733
		<code>\em</code>
		29:551, 29:578, 32:42, 714
		<code>\emergencystretch</code> ..
		49:132, 49:138, 1310
		<code>\emforce</code>
		713
		<code>\emforce</code>
		29:547, 29:574, 29:583, 714
		<code>\emminershape</code> ..
		29:555, 29:561, 29:578, 714
		<code>\emph</code>
		32:42, 1337

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpara.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=ltthyphen.dtx, 57=ltfinal.dtx

<code>\empty</code>	02:448	09:456, 09:508, 09:512, 09:559,
<code>\emptysetset</code>	30:329	09:586, 09:589, 09:601, 03:71 ,
<code>\emreset</code>	714	14:157, 14:161, 14:204, 14:210,
<code>\emreset</code>	29:547, 29:550, 29:581, 714	16:44, 16:75, 17:43, 17:54, 18:21,
<code>\emrest</code>	29:550	18:31, 18:65, 18:78, 18:86, 18:91,
<code>\ENC-cmd</code>	1300	18:104, 18:110, 18:152, 18:166,
<code>\encodingdefault</code>	04:260 ,	18:178, 18:189, 18:206, 18:218,
	04:275, 04:283, 21:1012, 21:1558 ,	18:250, 18:266, 18:283, 18:319,
	21:1595, 21:1596, 28:404, 29:659 ,	18:353, 18:375, 18:412, 18:445,
	29:677, 29:693, 29:710, 30:62, 1301	18:481, 18:485, 18:494, 18:498,
<code>\EncodingSpecific</code>	1296	18:506, 18:510, 18:520, 18:526,
<code>\EncodingSpecificAccent</code>	1289	18:540, 18:547, 20:83, 20:140,
<code>\EncodingSpecificAccentedLetter</code> ..	1289	20:195, 20:256, 20:277, 20:289,
<code>\EncodingSpecificCommand</code>	1289	20:352, 20:356, 20:435, 20:473,
<code>\end</code> 14:250, 26:9, 30:6, 31:6, 06:23, 37:268 ,		20:496, 20:531, 20:552, 20:571,
	37:296, 37:310, 37:346, 37:362 ,	20:578, 20:597, 20:614, 20:627,
	37:376, 37:384, 37:520, 37:521 ,	20:631, 20:649, 20:660, 20:670,
	38:491, 38:500, 39:112, 44:15 ,	20:709, 20:736, 21:121, 21:141,
	44:17, 50:1316, 50:1320, 50:1327 ,	21:186, 21:193, 21:350, 21:372,
	06:719, 50:1449, 50:1453, 50:1459 ,	21:387, 21:395, 04:234, 04:257,
	50:1538, 50:1542, 50:1548, 06:776 ,	04:280, 04:284, 22:30, 22:35, 22:59,
	53:396, 56:5, 07:1450, 07:1570 ,	22:77, 22:88, 22:93, 22:121, 22:126,
	07:1585, 07:1746, 01:53, 07:2850, 222	22:151, 22:178, 22:187, 22:229,
<code>\end\verbvisiblespace</code>	37:343	22:235, 22:255, 22:258, 23:10,
<code>\endaligned</code>	1250	23:14, 23:25, 23:30, 24:53, 24:74,
<code>\endarray</code>	41:171, 1250	24:227, 24:246, 24:294, 24:311,
<code>\endcenter</code>	37:445	24:357, 24:367, 24:377, 24:438,
<code>\endcsname</code>	205	24:447, 24:527, 24:532, 24:567,
<code>\enddisplaymath</code>	38:348	24:572, 24:589, 24:607, 24:646,
<code>\enddocument</code>	37:8, 37:77 ,	24:679, 24:796, 24:808, 25:1431,
	37:78, 37:141, 37:143, 53:352, 1122	25:2772, 25:2778, 25:2791, 25:2803,
<code>\endenumerate</code>	39:271	25:2810, 25:2893, 25:2908, 25:2930,
<code>\endeqnarray</code>	38:380, 38:427	25:2942, 25:3047, 25:3128, 25:3131,
<code>\endequation</code>	38:351	25:3142, 25:3149, 25:3156, 25:3229,
<code>\endfilecontents</code>	50:1206	25:3247, 25:3254, 25:3258, 25:3261,
<code>\endflushleft</code>	37:495	26:158, 26:161, 26:177, 26:550,
<code>\endflushright</code>	37:497	26:556, 27:21, 27:143, 28:78, 28:106,
<code>\endgraf</code>	16:140 ,	28:180, 28:210, 28:240, 28:272,
	16:178, 40:138, 40:144, 02:445, 1319	28:322, 28:365, 28:426, 28:432,
<code>\endgroup</code>	1284	28:437, 28:520, 28:527, 28:572,
<code>\EndIncludeInRelease</code> 08:1949, 02:579 ,		28:579, 28:853, 28:895, 28:908,
	08:2053, 08:2058, 08:2063, 08:2153 ,	28:915, 28:1103, 28:1111, 29:68,
	08:2160, 08:2184, 08:2200, 08:2207 ,	29:101, 29:123, 29:228, 29:249,
	08:2212, 08:2234, 08:2254, 02:613 ,	05:10, 29:301, 05:18, 29:348, 29:377,
	08:2284, 08:2302, 08:2307, 08:2317 ,	29:388, 05:27, 29:435, 29:469,
	08:2321, 08:2324, 08:2351, 02:621 ,	29:495, 29:531, 29:540, 29:577,
	08:2366, 08:2377, 08:2395, 08:2407 ,	29:589, 29:596, 29:630, 29:636,
	08:2419, 08:2429, 08:2459, 08:2478 ,	29:671, 29:687, 29:704, 29:719,
	08:2541, 08:2545, 02:646, 08:2627 ,	30:92, 30:102, 30:120, 30:129,
	08:2630, 08:2744, 08:2750, 08:2757 ,	05:110, 05:131, 30:644, 30:651,
	08:2762, 08:2769, 08:2773, 08:2780 ,	05:147, 32:33, 32:40, 05:156, 05:168,
	08:2784, 08:2807, 08:2812, 02:686 ,	05:176, 05:191, 05:197, 05:203,
	02:691, 09:37, 09:48, 02:701, 09:62 ,	05:208, 06:12, 06:18, 35:28, 35:45,
	02:706, 09:245, 09:312, 09:437 ,	35:61, 35:75, 35:82, 35:110, 35:115,

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpara.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

35:145, 35:154, 35:166, 35:176,
 35:188, 06:60, 35:199, 35:210,
 06:64, 37:75, 37:139, 37:178, 37:186,
 37:192, 37:225, 37:238, 37:266,
 37:277, 37:308, 37:333, 37:341,
 37:359, 37:374, 37:382, 37:387,
 37:401, 37:410, 37:420, 37:427,
 37:437, 37:442, 37:466, 37:482,
 37:491, 37:510, 37:516, 37:546,
 37:567, 37:589, 37:599, 37:616,
 37:628, 37:632, 37:640, 37:644,
 37:665, 37:678, 37:693, 37:705,
 37:727, 37:735, 38:86, 38:95, 38:117,
 38:124, 38:143, 38:148, 38:156,
 38:160, 38:175, 38:183, 38:232,
 38:249, 38:279, 38:287, 38:316,
 38:343, 38:407, 38:416, 38:448,
 38:453, 38:473, 38:485, 38:494,
 38:503, 06:224, 06:229, 39:134,
 39:139, 39:161, 39:169, 40:13, 40:22,
 40:57, 40:65, 40:90, 40:107, 40:121,
 40:132, 40:141, 40:146, 40:152,
 40:163, 40:170, 40:227, 40:234,
 40:260, 40:271, 40:307, 40:315,
 40:354, 40:375, 40:399, 40:417,
 40:494, 40:510, 40:527, 40:536,
 40:541, 40:564, 01:290, 40:571,
 40:605, 40:612, 41:64, 41:69, 41:156,
 41:164, 06:324, 41:226, 41:231,
 06:352, 42:15, 42:19, 42:38, 42:47,
 42:68, 42:76, 42:88, 42:96, 42:111,
 42:121, 42:150, 42:155, 42:171,
 42:182, 42:249, 42:263, 06:380,
 06:385, 42:368, 42:425, 01:300,
 42:462, 42:468, 42:499, 06:404,
 42:529, 42:554, 42:570, 42:581,
 42:594, 42:602, 42:623, 42:640,
 42:650, 06:418, 42:654, 42:744,
 42:799, 42:818, 42:835, 43:33,
 43:39, 43:56, 43:63, 44:19, 44:29,
 44:167, 44:173, 44:178, 44:195,
 44:207, 44:217, 44:223, 44:249,
 44:271, 06:459, 45:104, 06:469,
 45:172, 45:231, 45:246, 45:293,
 45:306, 45:351, 45:368, 45:411,
 45:417, 45:426, 45:430, 45:439,
 45:445, 45:449, 45:481, 45:498,
 45:516, 06:501, 45:558, 45:564,
 06:510, 47:24, 47:32, 47:76, 47:92,
 06:537, 06:543, 06:557, 06:562,
 49:34, 49:60, 49:81, 49:104, 49:115,
 49:124, 01:25, 50:21, 50:26, 50:49,
 50:61, 50:84, 50:93, 50:99, 50:111,
 50:142, 50:149, 50:174, 50:181,
 50:199, 50:210, 50:244, 50:261,
 50:272, 50:281, 50:312, 50:342,
 50:359, 50:371, 50:392, 50:405,
 50:418, 50:428, 50:464, 50:480,
 50:489, 50:523, 50:533, 50:573,
 50:589, 50:611, 50:626, 50:640,
 50:647, 50:662, 50:671, 50:705,
 50:712, 50:729, 50:736, 06:663,
 50:789, 50:796, 50:834, 50:861,
 50:889, 06:678, 50:1045, 50:1102,
 50:1132, 50:1139, 50:1157, 50:1167,
 50:1342, 50:1473, 06:729, 06:733,
 50:1563, 06:765, 06:774, 52:14,
 52:23, 52:88, 52:140, 52:179, 52:191,
 52:200, 52:245, 52:256, 52:263,
 52:298, 52:321, 52:342, 52:356,
 06:824, 52:361, 52:384, 52:404,
 06:831, 52:429, 52:474, 52:494,
 52:501, 52:531, 52:536, 06:853,
 06:864, 06:867, 06:895, 53:429,
 06:916, 53:466, 53:483, 53:487,
 54:49, 54:58, 54:176, 54:194, 54:363,
 54:368, 54:416, 54:462, 54:686,
 54:765, 54:869, 54:928, 54:986,
 54:1085, 54:1104, 54:1167, 54:1185,
 54:1227, 54:1248, 54:1491, 54:1635,
 54:1804, 54:1887, 54:1967, 54:2061,
 54:2183, 54:2310, 54:2420, 54:2428,
 54:2462, 54:2491, 54:2529, 54:2533,
 07:240, 54:2697, 54:2746, 54:2776,
 54:2794, 54:2825, 07:263, 54:2866,
 54:2910, 57:15, 57:19, 57:29, 57:33,
 57:51, 57:69, 57:79, 57:86, 57:94,
 57:141, 57:146, 57:198, 57:222,
 57:292, 57:297, 57:338, 57:344,
 57:440, 57:487, 02:87, 01:34, 02:101,
 02:118, 02:123, 02:133, 02:137,
 02:147, 02:150, 07:1176, 07:1179,
 07:1194, 02:167, 07:1207, 07:1210,
 02:181, 07:1340, 07:1343, 02:185,
 07:1372, 07:1375, 07:1384, 07:1392,
 07:1395, 07:1555, 07:1559, 02:219,
 02:224, 02:232, 02:240, 07:2189,
 07:2200, 07:2232, 07:2248, 07:2259,
 02:288, 02:300, 02:359, 02:367,
 07:3159, 07:3189, 07:3259, 07:3266,
 08:85, 08:101, 08:129, 08:145,
 08:157, 08:169, 08:185, 08:198,
 08:218, 08:228, 08:248, 08:263,
 08:281, 08:287, 08:307, 08:325,
 08:329, 08:579, 08:630, 08:647,
 08:651, 08:674, 08:695, 08:729,
 02:471, 08:787, 08:817, 08:849,
 08:876, 08:879, 08:899, 08:902,

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpara.dtx,
 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
 56=ltthyphen.dtx, 57=ltfinal.dtx

08:915, 02:488, 08:933, 08:938,	flushleft	37:494
08:941, 08:950, 08:955, 08:958,	flushright	37:496
08:995, 08:1032, 08:1082, 08:1089,	itemize	39:273
08:1105, 08:1112, 08:1132, 08:1136,	list	39:34
08:1188, 08:1192, 08:1208, 08:1212,	lrbbox	878
08:1240, 08:1243, 08:1266, 08:1270,	math	38:345
08:1287, 08:1292, 08:1299, 08:1303,	minipage	879
08:1339, 08:1363, 08:1446, 08:1473,	picture	42:21
08:1507, 08:1546, 08:1635, 08:1718, 28	sloppypar	49:135
\EndIncludeRelease	tabbing	41:71
\endinput	tabular	41:174
\enditemize	thebibliography	998
\endline	trivlist	39:89
\endlinechar	verbatim	37:569
20:718, 01:188, 06:37, 06:39, 06:44,	verbatim*	37:584
50:433, 50:434, 50:435, 57:329,	\epsilon	30:283
02:380, 01:76, 01:77, 01:78, 1326	eqnarray (env.)	38:357, 38:516
\endlist	\eqnarray	38:362, 38:426
\endlrbox	eqnarray* (env.)	38:425
\endmath	\eqno	38:351, 38:439, 1304
\endminipage	equation (env.)	38:349, 38:504
\EndModuleRelease	\equation	38:350
08:2931,	\equiv	30:459
09:625, 10:224, 03:111, 03:112,	\errhelp	03:31, 14:39,
03:152, 11:1222, 16:183, 17:63,	14:66, 01:201, 56:12, 57:307, 57:709	
33:773, 36:280, 48:517, 55:310, 07:3279	\errmessage	03:32, 10:77,
\endpicture	14:47, 14:72, 04:63, 01:7, 24:618,	
\endscname	24:653, 01:206, 26:426, 26:526,	
\endsloppypar	27:65, 05:79, 05:94, 56:16, 57:63,	
\endsmallmatrix	57:309, 01:42, 02:164, 02:178, 02:304	
\endtabbing	\ERROR	07:1865, 07:1875, 07:1884,
\endtabular	07:2284, 07:2297, 07:2317, 07:2336,	
\endtabular*	07:2412, 07:2439, 08:1719, 08:1720, 175	
\endtrivlist	\errorcontextlines	02:569,
... 37:445, 37:495, 37:497, 37:570,	02:588, 02:604, 02:619, 02:633,	
38:515, 39:100, 39:101, 41:85, 43:66	02:657, 02:674, 14:212, 02:389, 1282	
\endverbatim	\ERRORmissingcells	55:206
\enlargethispage	\errorstopmode	57:719, 02:554, 1313
\enlargethispage*	\ERRORusetaggingsocket	55:21, 55:31
\enskip	\escapchar	464
\enspace	\escapechar	20:454, 04:212,
\ensuremath	24:464, 24:688, 26:230, 28:59, 28:87,	
45:409, 45:416, 45:437, 45:444, 1318	28:159, 28:190, 28:220, 28:251,	
enumerate (env.)	28:388, 06:126, 06:169, 06:173,	
\enumerate	06:181, 06:287, 06:288, 06:312,	
environments:	06:449, 06:572, 06:590, 52:280,	
array	52:303, 52:326, 52:347, 02:377, 102	
center	\eta	30:285
displaymath	\etatcatcode	04:1026
document	\TeXversion	01:41
enumerate	\evensidemargin	
eqnarray	54:69, 54:821, 54:891, 54:950	
eqnarray*	\everycr	38:205,
equation	38:208, 38:371, 38:532, 02:534, 1285	
filecontents		

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- `\everydisplay` 24:419, [24:420](#), 24:431, 24:443
`\everyeof` 57:329
`\everyjob` 03:37, 04:216, 04:262, 04:263,
28:408, 53:29, 53:30, 57:112, 57:331,
57:416, 57:679, 57:680, 57:682, *1334*
`\everymath` . 24:418, [24:420](#), 24:436, 24:445
`\everypar` 408
`\everypar`
16:35, 16:41, 16:72, [16:77](#), 16:120,
16:130, 16:143, 16:181, 20:43,
20:113, 20:171, 24:741, 24:755,
24:802, 37:241, 37:543, 37:565,
39:130, 39:131, 39:133, 39:137,
39:138, 39:211, 39:228, 40:389,
40:410, 41:81, 44:48, 44:96, 44:107,
44:127, 44:136, 45:187, 54:163,
54:190, 54:1426, 54:1575, 54:1736, *421*
`\EveryShipout` *1131*
`\ExecuteOptions` 26:58, 26:71, 33:841, [50:650](#)
`\exhyphenpenalty` 02:322, 02:507
`\exists` 30:335
`\exp` 38:31
exp commands:
`\exp:w` ... 07:2654, 08:758, 08:812, *251*
`\exp_after:wN` 08:1841,
08:1874, 08:1959, 08:2103, 08:2248,
08:2432, 08:2435, 08:2499, 08:2961,
09:30, 09:32, 09:187, 09:262, 09:316,
09:317, 09:370, 09:488, 11:285,
11:747, 11:880, 11:887, 16:41, 16:72,
28:335, 28:338, 51:46, 52:73, 52:236,
52:237, 07:325, 07:341, 07:395,
07:401, 07:415, 07:1168, 07:1188,
07:1192, 07:1203, 07:1205, 07:1254,
07:1303, 07:1333, 07:1364, 07:1371,
07:1447, 07:1581, 07:1834, 07:1858,
07:1874, 07:1892, 07:1898, 07:1902,
07:1933, 07:1967, 07:1974, 07:1987,
07:1993, 07:1999, 07:2005, 07:2011,
07:2017, 07:2023, 07:2029, 07:2060,
07:2204, 07:2657, 07:2658, 07:2704,
07:2705, 07:2706, 07:2707, 07:2708,
07:2709, 07:2710, 07:2741, 08:56,
08:61, 08:380, 08:381, 08:426,
08:757, 08:811, 08:1041, 08:1043,
08:1184, 08:1349, 08:1522, 08:1732, *251*
`\exp_args:Nc` 08:2189, 09:46,
09:59, 09:91, 09:149, 48:22, 07:266,
07:1099, 07:1110, 07:1410, 07:1451,
07:1470, 07:1574, 08:1205, 08:1218
`\exp_args:Ncc` 07:141, 07:183
`\exp_args:NcV` 08:570
`\exp_args:Ne`
..... 08:2337, 08:2338, 09:522,
10:139, 11:852, 51:113, 51:144,
52:46, 52:97, 07:2730, 08:715, 08:717
`\exp_args:Nee` 52:129
`\exp_args:Nf` 09:178,
09:253, 52:32, 52:34, 52:394, 07:420,
07:2416, 07:2443, 07:2504, 07:2732
`\exp_args:NNc` 09:163, 09:168
`\exp_args:NNe` 07:1380, 08:1624
`\exp_args:Nne` 08:1851
`\exp_args:NNNo` 07:401, 07:1737, 07:2102
`\exp_args:Nnrv` 08:257
`\exp_args:NNo` ... 08:2962, 09:186,
09:261, 08:1065, 08:1122, 08:1713
`\exp_args:NNV`
.... 09:205, 09:280, 08:1597, 08:1684
`\exp_args:NNx` 07:1469, 08:1503, 08:1541
`\exp_args:No`
..... 08:2062, 08:2121, 08:2951,
08:2962, 09:186, 09:205, 09:261,
09:280, 36:41, 36:46, 07:93, 07:322,
07:412, 07:431, 07:440, 07:528,
07:932, 07:1339, 07:2289, 07:2300,
07:2341, 07:2422, 07:2449, 07:2672,
07:2698, 08:1065, 08:1122, 08:1183
`\exp_args:Nof` 07:2292
`\exp_args:Noo` 07:1312
`\exp_args:NV` 09:206,
09:281, 51:195, 52:53, 07:259,
07:364, 07:1536, 07:1743, 07:1746
`\exp_args:Nv`
.. 08:1991, 08:1999, 08:2101, 08:1250
`\exp_args:Nx` 36:10, 53:29, 07:2671,
07:2674, 07:2697, 08:432, 08:474, 08:546
`\exp_args_generate:n` 57:597
`\exp_end:` 07:2657, 08:758, 08:812
`\exp_last_unbraced:Ne`
..... 08:2953, 07:2743,
08:1067, 08:1100, 08:1124, 08:1217
`\exp_last_unbraced:Nf`
..... 08:1898, 09:127, 08:1230
`\exp_last_unbraced:NNf` 08:1158
`\exp_last_unbraced:NNNo`
..... 08:2056, 09:133, 09:491, 08:383
`\exp_last_unbraced:NnNo` 07:2516
`\exp_last_unbraced:NNo` 09:381
`\exp_not:N` .. 09:125, 09:127, 09:128,
09:129, 09:201, 09:216, 09:223,
09:224, 09:231, 09:236, 09:276,
09:291, 09:298, 09:299, 09:303,
09:307, 09:324, 09:369, 09:375,
09:393, 09:402, 09:476, 09:477,
09:497, 09:502, 09:525, 09:552,
09:553, 09:580, 09:581, 11:277,
11:278, 11:279, 11:280, 11:290,

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx,
16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
56=ltthyphen.dtx, 57=ltfinal.dtx

- 11:296, 11:299, 11:301, 11:302,
 11:306, 11:308, 11:311, 11:316,
 11:467, 11:468, 11:483, 11:485,
 11:501, 11:674, 11:676, 11:779,
 11:787, 11:827, 11:842, 11:867,
 11:874, 16:39, 16:40, 16:70, 16:71,
 16:79, 16:80, 36:72, 48:102, 48:105,
 48:107, 48:112, 48:115, 51:195,
 51:196, 51:197, 53:31, 53:95, 53:125,
 07:123, 07:138, 07:139, 07:167,
 07:168, 07:200, 07:202, 07:203,
 07:209, 07:211, 07:270, 07:282,
 07:299, 07:300, 07:301, 07:303,
 07:371, 57:604, 07:941, 07:951,
 07:1032, 07:1045, 07:1069, 07:1087,
 07:1094, 07:1132, 07:1140, 07:1141,
 07:1173, 07:1219, 07:1220, 07:1227,
 07:1228, 07:1229, 07:1233, 07:1236,
 07:1237, 07:1239, 07:1258, 07:1261,
 07:1262, 07:1263, 07:1271, 07:1318,
 07:1337, 07:1351, 07:1358, 07:1359,
 07:1460, 07:1461, 07:1623, 07:1630,
 07:1646, 07:1650, 07:1652, 07:1679,
 07:1692, 07:1696, 07:1697, 07:1701,
 07:1702, 07:1706, 07:1707, 07:1711,
 07:1712, 07:1772, 07:1778, 07:1779,
 07:1781, 07:1782, 07:1784, 07:1785,
 07:2034, 07:2035, 07:2143, 07:2145,
 07:2146, 07:2228, 07:2229, 07:2240,
 07:2243, 07:2256, 07:2733, 07:2736,
 07:2737, 07:2738, 07:2739, 07:2741,
 08:391, 08:1152, 08:1156, 08:1175,
 08:1178, 08:1179, 08:1181, 08:1225,
 08:1487, 08:1489, 08:1625, 08:1627, 322
- `\exp_not:n` 08:2951,
 09:188, 09:201, 09:204, 09:206,
 09:241, 09:263, 09:276, 09:279,
 09:281, 09:309, 09:373, 09:403,
 09:405, 09:406, 09:408, 09:409,
 11:282, 11:470, 11:595, 11:675,
 11:780, 11:781, 11:788, 11:826,
 11:828, 11:829, 11:842, 11:845,
 11:846, 11:867, 11:868, 11:874,
 11:875, 36:238, 48:172, 48:179,
 48:251, 48:252, 48:253, 48:283,
 48:360, 51:179, 52:211, 52:225,
 53:30, 07:162, 07:166, 07:170,
 07:173, 07:177, 07:194, 07:201,
 07:213, 07:283, 07:302, 07:371,
 07:374, 07:375, 07:628, 07:770,
 07:888, 07:899, 07:901, 07:902,
 07:981, 07:1033, 07:1046, 07:1057,
 07:1069, 07:1086, 07:1172, 07:1174,
 07:1227, 07:1240, 07:1272, 07:1307,
 07:1565, 07:1567, 07:1580, 07:1616,
 07:1617, 07:1678, 07:1720, 07:1734,
 07:1735, 07:1742, 07:1759, 07:1843,
 07:1844, 07:1887, 07:1947, 07:2144,
 07:2307, 07:2308, 07:2348, 07:2349,
 07:2531, 07:2540, 07:2557, 07:2571,
 07:2583, 07:2733, 08:114, 08:241,
 08:549, 08:1049, 08:1073, 08:1172,
 08:1176, 08:1182, 08:1183, 08:1631, 326
- `\exp_stop_f:` 07:1492,
 07:2294, 08:1251, 08:1399, 08:1407
- expandable commands:
`\expandable_grab_{type}:w` 148
- `\expandafter` 1318
- expandafter commands:
`\expandafter:` 32:88, 41:269
- `\ExpandArgs` 78
- `\ExpandArgs` 05:177,
 05:193, 05:196, 06:534, 06:535, 57:601
- `\expanded` 50:902, 57:111, 241
- `\ExplLoaderFileDate` 05:201, 05:207
- `\ExplSyntaxOff`
 . . 08:2968, 09:398, 09:626, 10:225,
 11:1223, 16:184, 17:42, 17:64,
 20:529, 20:749, 20:799, 24:417,
 28:320, 28:363, 05:146, 05:166,
 05:189, 36:281, 48:518, 49:32, 49:59,
 49:113, 50:515, 51:260, 52:12, 52:86,
 52:138, 52:243, 52:254, 52:402,
 52:427, 52:472, 52:529, 52:584,
 53:427, 53:474, 53:514, 55:59,
 55:300, 57:570, 57:654, 07:3280, 1045
- `\ExplSyntaxOn` . 09:3, 09:398, 10:2, 11:6,
 16:3, 17:2, 17:22, 20:518, 20:743,
 20:796, 24:412, 28:278, 28:326,
 05:135, 05:161, 05:180, 36:2, 48:3,
 49:22, 49:37, 49:110, 50:513, 51:3,
 52:7, 52:29, 52:92, 52:207, 52:251,
 52:391, 52:410, 52:435, 52:508,
 52:577, 53:5, 53:472, 53:509, 07:8,
 55:2, 55:160, 57:560, 57:571, 08:3, 310
- `\extracolsep` 41:167
- `\extrafloats` 02:152, 02:189, 02:274
- F**
- `\fam` 04:22, 04:26, 04:38, 24:16, 02:98, 02:375
- `\family` 1283
- `\familydefault`
 . . 21:1596, 28:405, 29:445, 29:660,
 29:678, 29:694, 29:711, 30:131, 702
- `\fbox` 878
- `\fbox` 40:212, 40:225, 40:232, 1298
- `\fboxrule` 40:210, 40:257, 40:269, 40:274,
 40:280, 40:282, 40:289, 40:290, 57:151

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx,
 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
 56=lthyphen.dtx, 57=ltfinal.dtx

- \fboxsep [40:210](#), [40:216](#), [40:249](#), [40:256](#),
[40:268](#), [40:275](#), [40:285](#), [40:287](#), [57:150](#)
- \fi [1281](#)
- fi commands:
 - \fi: [08:2104](#), [08:2224](#), [08:2232](#),
[08:2249](#), [08:2271](#), [08:2299](#), [08:2436](#),
[08:2441](#), [08:2457](#), [08:2504](#), [08:2539](#),
[08:2554](#), [09:359](#), [09:361](#), [09:362](#),
[09:366](#), [09:371](#), [09:378](#), [16:26](#),
[16:33](#), [16:65](#), [16:103](#), [52:241](#), [07:899](#),
[07:902](#), [07:980](#), [07:1337](#), [07:1494](#),
[07:1826](#), [07:2654](#), [07:2658](#), [08:770](#),
[08:1044](#), [08:1201](#), [08:1224](#), [08:1264](#),
[08:1403](#), [08:1411](#), [08:1417](#), [08:1735](#), [326](#)
- \filbreak [02:511](#)
- file commands:
 - \g_file_curr_name_str [08:2798](#)
 - \file_full_name:n [20:525](#), [52:35](#), [1109](#)
 - \file_md5five_hash:n [20:797](#)
 - \file_parse_full_name_apply:nN [52:32](#), [52:47](#), [52:90](#), [52:93](#), [52:95](#)
 - \l_file_search_path_seq [1116](#)
 - \file_size:n [20:798](#)
- file internal commands:
 - __file_parse_full_name_area:nw [52:102](#), [52:105](#), [52:109](#)
 - __file_parse_full_name_auxi:nN [52:97](#), [52:100](#)
 - __file_parse_full_name_base:nw [52:108](#), [52:112](#), [52:124](#)
 - __file_parse_full_name_tidy:nnnN [52:119](#), [52:120](#), [52:122](#), [52:127](#)
- file/.../after [1100](#)
- file/.../before [1100](#)
- file/after [1100](#)
- file/before [1100](#)
- filecontents (env.) [50:1206](#), [1039](#)
- \filecontents [50:1206](#)
- filehook internal commands:
 - __filehook_clear_replacement_
flag: [52:417](#), [52:420](#), [52:520](#)
 - __filehook_drop_extension:N [52:44](#), [52:49](#), [52:522](#)
 - __filehook_drop_extension_
aux:nnn [52:54](#), [52:57](#)
 - __filehook_file_name_compose:nnn
[52:218](#), [52:406](#), [52:414](#), [52:415](#), [52:425](#)
 - __filehook_file_parse_full_
name:nN [52:28](#), [52:30](#), [52:30](#), [52:53](#),
[52:238](#), [52:394](#), [52:412](#), [52:414](#), [1106](#)
 - __filehook_file_pop: [52:59](#), [52:70](#), [52:526](#)
 - __filehook_file_pop_assign:nnnn
[52:59](#), [52:73](#), [52:79](#), [52:528](#)
- __filehook_file_push: [52:59](#), [52:62](#), [52:524](#)
- __filehook_file_replaced [52:417](#)
- __filehook_file_subst_begin:nnn
[52:412](#), [52:422](#), [52:422](#), [1116](#)
- __filehook_file_subst_cycle_
error:NN [52:457](#),
[52:462](#), [52:462](#), [52:467](#), [52:469](#), [1118](#)
- __filehook_file_subst_loop:NN [52:431](#), [52:444](#), [52:452](#), [52:461](#), [1117](#)
- __filehook_file_subst_tortoise_
hare:nn [52:424](#),
[52:431](#), [52:434](#), [52:436](#), [52:459](#), [1118](#)
- __filehook_full_name:nn [52:30](#), [52:34](#), [52:38](#)
- __filehook_if_file_replaced:TF [52:417](#), [52:418](#), [52:518](#), [1117](#)
- __filehook_if_no_extension:nTF [52:44](#), [52:44](#), [52:510](#)
- \g_filehook_input_file_seq [52:59](#), [1343](#)
- \l__filehook_internal_tl [52:59](#)
- __filehook_log_file_record:n [52:547](#), [52:547](#), [52:561](#), [52:563](#)
- \g__filehook_nesting_level_int [52:544](#), [52:549](#), [52:552](#), [52:553](#), [52:559](#)
- __filehook_normalize_file_
name:w [52:406](#), [52:413](#), [52:516](#)
- __filehook_resolve_file_subst:w
[52:406](#), [52:409](#), [52:411](#), [52:514](#)
- __filehook_set_curr_file:nNN [52:387](#), [52:390](#), [52:392](#), [52:512](#)
- __filehook_set_curr_file_
assign:nnnNN [52:387](#), [52:395](#), [52:397](#)
- __filehook_subst_add:nn [52:206](#), [52:208](#), [52:208](#), [52:252](#), [1111](#)
- __filehook_subst_empty_name_
chk:NN [52:208](#), [52:236](#), [52:240](#)
- __filehook_subst_file_normalize:Nn
[52:208](#), [52:216](#), [52:218](#), [52:230](#), [52:234](#)
- __filehook_subst_remove:n [52:208](#), [52:222](#), [52:253](#)
- \filename@parse [6](#), [1](#)
- \filesize [05:72](#), [05:118](#)
- \fill [18:530](#)
- \finalhyphdemerits [37:453](#), [37:457](#), [37:463](#), [02:335](#), [1339](#)
- \finalstrut [1307](#)
- \FirstMark [48:388](#), [48:508](#), [1009](#)
- \firstmark [49:120](#), [54:925](#), [54:984](#), [54:2844](#), [1003](#)
- flag internal commands:
 - \flag__filehook_file_replaced [1117](#)

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscomp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- flag commands:
- `\flag_clear:n` 52:421
 - `\flag_if_raised:nTF`
..... 36:109, 36:201, 52:419, 52:440
 - `\flag_new:n` 36:88, 52:417
 - `\flag_raise:n` 36:110, 52:441
 - `\flat` 30:339
 - `\floatingpenalty`
..... 45:469, 45:488, 45:506, 02:332
 - `\floatpagefraction` 45:278, 54:2922
 - `floats:footnotes (plug)` 54:665
 - `floats:space:footnotes (plug)` ... 54:649
 - `\floatsep` .. 54:1000, 54:1018, 54:1025,
54:2653, 54:2703, 54:2753, 54:2927
 - `\flushbottom` 49:128, 1183
 - `flushleft (env.)` 37:494
 - `\flushleft` 37:494
 - `flushright (env.)` 37:496
 - `\flushright` 37:496
 - `\fmtname` 03:1, 03:41, 03:43, 03:46, 03:48,
03:51, 03:60, 50:749, 50:753, 1286
 - `\fmtversion`
.. 03:1, 03:19, 03:41, 03:43, 03:46,
03:48, 03:51, 03:60, 03:105, 03:118,
03:166, 14:2, 04:276, 21:1560,
29:226, 41:1, 42:1, 50:169, 50:177,
50:766, 50:769, 57:663, 57:689, 1286
 - `\fnsymbol` 521
 - `\fnsymbol` 22:194
 - `\font` 21:317, 21:318,
21:319, 21:459, 21:466, 21:821,
21:828, 21:888, 21:1025, 21:1026,
21:1082, 21:1187, 21:1189, 21:1191,
21:1238, 24:82, 24:88, 24:90, 26:85,
29:554, 29:587, 29:593, 29:619, 31:8,
31:9, 31:10, 32:85, 33:6, 33:576,
33:590, 33:597, 37:564, 06:842,
06:845, 06:857, 06:860, 02:539, 02:544
 - `\fontdimen` 21:317, 21:318,
21:319, 21:459, 21:466, 21:821,
21:828, 29:554, 29:587, 29:593,
32:85, 33:6, 33:576, 33:590, 33:597,
42:126, 42:129, 42:678, 02:539, 02:544
 - `\fontencoding` .. 04:259, 04:260, 21:889,
04:282, 04:283, 21:1595, 24:314,
24:349, 24:361, 24:371, 28:404,
29:659, 29:677, 29:693, 30:16, 30:24,
30:79, 30:86, 30:95, 30:97, 33:36, 1298
 - `\fontfamily` . 24:342, 29:7, 29:10, 29:13,
29:146, 29:157, 29:483, 29:486,
29:489, 29:739, 30:70, 30:81, 30:88,
30:99, 33:28, 33:30, 33:32, 33:34,
33:49, 33:51, 33:53, 33:55, 33:877, 618
 - `\FontFamilyToCheck`
..... 33:1584, 33:1585, 33:1598
 - `\fontname` 21:1026, 24:90
 - `\fontseries` 24:342, 25:2781,
25:2782, 25:2793, 25:2795, 25:2805,
25:2807, 29:16, 29:19, 29:259,
29:260, 29:261, 29:262, 29:282,
29:283, 29:284, 29:285, 29:320,
29:321, 29:322, 29:323, 29:340,
29:341, 29:342, 29:343, 29:356,
29:357, 29:358, 29:359, 29:367,
29:368, 29:369, 29:370, 29:383,
29:386, 29:477, 29:480, 29:740, 637
 - `\fontseriesforce`
..... 25:2786, 25:2797, 25:2808, 618
 - `\fontshape` 21:469, 21:831,
24:342, 25:2916, 25:2921, 25:2926,
25:3134, 25:3135, 25:3144, 25:3146,
25:3151, 25:3153, 25:3204, 25:3208,
25:3211, 25:3214, 25:3217, 25:3220,
25:3223, 25:3226, 25:3233, 29:22,
29:25, 29:28, 29:31, 29:741, 33:600, 637
 - `\fontshapeforce`
.. 25:3138, 25:3147, 25:3154, 25:3234
 - `\fontsize` 19:6, 21:322,
21:348, 21:380, 21:890, 21:1240,
21:1276, 24:80, 24:381, 29:642,
29:742, 33:68, 33:636, 33:894,
45:409, 45:416, 45:437, 45:444, 1299
 - `\fontsubfuzz` 26:442,
26:476, 37:53, 37:118, 37:157, 1295
 - `\F00` 81
 - `\foo` 320
 - `\footins` 45:390, 45:465,
45:484, 45:502, 54:312, 54:313,
54:314, 54:315, 54:373, 54:420,
54:584, 54:588, 54:594, 54:618,
54:691, 54:697, 54:701, 54:768, 1186
 - `\footnote` 45:452, 1282
 - `\footnotemark` 44:10, 45:518, 1282
 - `\footnoterule`
. 40:466, 45:394, 54:592, 54:700, 1312
 - `footnotes:floats (plug)` 54:670
 - `footnotes:floats:legacy (plug)` .. 54:675
 - `footnotes:space:floats (plug)` ... 54:642
 - `\footnotesep` 40:490,
40:507, 40:524, 45:451, 45:468,
45:477, 45:487, 45:495, 45:505, 45:513
 - `\footnotesize` 40:482,
40:500, 40:517, 45:466, 45:485, 45:503
 - `\footnotetext` 44:12, 45:535, 1282
 - `\footref` 45:547, 1343
 - `\footskip` 54:73, 54:852, 54:915, 54:974, 1181
 - `\forall` 30:334

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpara.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- fp commands:
- `\fp_eval:n` 11:368, 05:162, 49:40, 49:51, 49:66, 49:74, 49:89, 49:97, 54:829, 54:899, 54:958, 1359
 - `\l_tmpa_fp` 354
 - `\fpeval` 77
 - `\fpeval` 05:158, 05:170, 05:172, 77
 - `\frac` 38:354, 1288
 - `\frame` 40:196, 40:297
 - `\framebox` 878
 - `\framebox` 40:219, 1303
 - `\frenchspacing` 20:46, 20:116, 20:174, 37:569, 37:586, 37:737, 06:882, 06:903, 02:431
 - `\frown` 30:462
 - `\fussy` 49:137
 - `\futurelet` 18:457, 18:465, 05:49, 32:83, 38:256, 41:359, 06:782, 06:796
- G**
- `\g_hook_\meta_{hook}_code_prop` ... 230
 - `\g_hook_\meta_{hook}_declared_tl` . 230
 - `\g_hook_\meta_{hook}_parameter_tl` 230
 - `\g_hook_\meta_{hook}_reversed_tl` . 230
 - `\Gamma` 30:308
 - `\gamma` 30:281
 - `\gathered` 1250
 - `\gcd` 38:33
 - `\gdef` 1336
- gdef commands:
- `\gdef` 38:269
 - `\ge` 30:428, 30:430
 - `\Generic*` 1306
 - `\GenericError` 14:18, 14:85, 14:111, 14:164, 26:63, 1302
 - `\GenericInfo` ... 03:106, 03:108, 03:119, 03:122, 03:127, 03:162, 03:163, 03:168, 03:172, 03:176, 03:197, 14:4, 14:104, 14:130, 14:182, 21:8, 26:32, 26:35, 26:40, 26:76, 05:63, 50:66, 50:74, 50:104, 50:107, 50:1734, 1302
 - `\GenericWarning` 14:11, 14:94, 14:120, 14:141, 14:149, 14:173, 14:195, 26:43, 26:48, 26:51, 26:79, 1302
 - `\geq` 30:426, 30:428
 - `\getanddefinefonts` 1280
 - `\GetFileInfo` 30:3
 - `\gets` 30:450, 30:452
 - `\gg` 30:444
 - `\global` 1319
 - `\globaldefs` 24:690, 26:232, 28:61, 28:90, 28:161, 28:192, 28:222, 28:254, 02:372
 - `\glossary` 995
 - `\glossary` 44:192, 44:200, 46:23, 46:35, 48:268,
 - `\glossaryentry` 46:32
 - `\gluestretchorder` 54:563
 - `\glyphmissingdetails` 33:1183, 33:1208, 788
 - `\goodbreak` 06:883, 06:904, 02:511
 - `\grave` 30:528
 - `\group` 1280
- group commands:
- `\group_align_safe_begin/end:` .. 169
 - `\group_align_safe_begin:` 07:289, 07:301, 07:406, 125
 - `\group_align_safe_end:` 07:323, 07:429, 07:2090, 169
 - `\group_begin:` 09:29, 09:218, 09:293, 16:17, 16:23, 16:50, 16:56, 28:298, 28:348, 48:67, 48:236, 48:359, 51:4, 52:210, 52:224, 53:92, 07:1600, 07:1609, 07:1642, 07:1688, 07:1727, 07:1791, 07:1798, 07:2082, 07:2486, 07:2491, 07:2663, 07:2667, 07:2695, 08:387, 08:1151, 08:1164
 - `\c_group_begin_token` 07:690, 07:701, 07:1665, 07:2088, 07:2169, 07:2775
 - `\group_end:` . 09:31, 09:222, 09:297, 16:19, 16:28, 16:52, 16:60, 28:301, 28:351, 48:72, 48:255, 48:372, 51:21, 52:220, 52:232, 53:94, 07:1627, 07:1673, 07:1715, 07:1737, 07:1748, 07:1803, 07:1807, 07:1811, 07:2103, 07:2495, 07:2500, 07:2677, 07:2678, 07:2680, 07:2682, 07:2692, 07:2699, 07:2700, 08:390, 08:1155, 08:1174, 316
 - `\c_group_end_token` 07:693, 07:1704, 07:2158
 - `\group_insert_after:N` 28:334, 28:335, 28:338, 28:360, 28:361
 - `\guillemetleft` 21:561, 21:795, 21:1126, 1334
 - `\guillemetright` 21:562, 21:796, 21:1142, 1334
 - `\guillemotleft` ... 21:564, 21:798, 21:1128
 - `\guillemotright` .. 21:565, 21:799, 21:1144
 - `\guilsinglleft` 21:566, 21:1205
 - `\guilsinglright` 21:567, 21:1206
- H**
- `\H` 14:24, 21:250, 21:405, 21:490, 21:622, 21:630, 21:649, 21:657, 21:775, 21:1268, 21:1407, 21:1408, 21:1435, 21:1436, 33:141, 33:165
 - `\halign` 38:127, 38:210, 38:371, 38:532, 02:534, 1285
 - `\hang` 1311
 - `\hangafter` 02:374

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- `\hangindent` 44:139, 02:405
`\hat` 30:534
`\hbadness` 24:744, 24:751, 24:787, 24:806,
53:238, 53:240, 53:290, 53:292, 02:318
`\hbar` 30:346
`\hbox` 1126
hook commands:
`\hbox:n` 53:382
`\hbox_set:Nn`
53:170, 53:217, 53:243, 53:268, 53:297
`\hbox_set_to_wd:Nnn`
..... 53:241, 53:293, 53:329
`\hbox_unpack:N`
..... 48:411, 48:443, 53:251, 53:295
`\hbox_unpack_drop:N` 53:180
`\hbox_to_` 1296
`\headheight` .. 54:71, 54:836, 54:904, 54:963
`\headsep` 54:72, 54:850, 54:913, 54:972
`\heartsuit` 30:344
`\height` . 21:894, 21:1244, 40:31, 40:34, 1287
`\hfil` 862
`\hfill` 1283
`\hfuzz` 24:752,
49:133, 49:134, 49:140, 49:141,
53:237, 53:239, 53:289, 53:291, 02:390
`\hglue` 02:501
`\hideoutput` 02:692
`\hideskip` 02:309, 02:525
`\hidewidth` 21:347,
21:349, 21:378, 21:382, 21:410,
21:411, 21:414, 21:417, 21:497,
21:498, 21:502, 21:505, 21:507,
21:510, 21:522, 21:527, 21:543,
21:782, 21:783, 21:786, 21:789,
21:856, 21:859, 21:1275, 21:1277, 02:525
`\hline` 41:358, 41:361, 1284
`\hoffset` 02:406
`\holdinginserts` 02:339
`\hom` 38:29
hook commands:
`\hook_activate_generic:n`
..... 08:2754, 08:2840,
08:2859, 08:2872, 08:288, 08:288,
08:290, 08:308, 08:326, 08:328, 212
`\hook_debug_off:`
..... 08:2816, 08:7, 08:13, 214
`\hook_debug_on:` 08:2815, 08:7, 08:8, 214
`\hook_disable:n` 08:2828, 08:2828
`\hook_disable_generic:n`
08:2756, 08:2833, 08:2865, 08:264,
08:264, 08:266, 08:282, 08:285, 212
`\hook_gclear_next_code:n`
..... 08:2201, 08:2201, 08:2786, 213
`\hook_gput_code:nnn`
.. 08:2766, 08:486, 08:486, 08:488,
08:580, 08:582, 08:682, 08:703, 213
`\hook_gput_code_with_args:nnn` ...
. 08:2768, 08:486, 08:494, 08:629, 213
`\hook_gput_next_code:nn`
08:2123, 08:2123, 08:2125, 08:2154,
08:2156, 08:2777, 08:691, 08:709, 213
`\hook_gput_next_code_with_`
`args:nn` 08:2139, 08:2159, 08:2779, 213
`\hook_gremove_code:nn` .. 08:2788,
08:959, 08:959, 08:961, 08:996, 213
`\hook_gset_rule:nnnn` ... 08:2818,
08:2820, 08:2823, 08:1304, 08:1304, 265
`\hook_if_empty:n`
.. 08:2443, 08:2445, 08:2460, 08:2462
`\hook_if_empty:nTF`
08:1863, 08:1969, 08:2443, 08:2824,
08:2825, 08:2826, 53:63, 53:164, 206
`\hook_if_empty_p:n` 08:1918, 08:2018,
08:2443, 53:67, 53:113, 53:377, 214
`\hook_log:n`
..... 08:1819, 08:1819, 08:2814, 214
`\hook_new:n` 08:2731,
48:274, 53:153, 53:154, 53:155,
53:156, 53:157, 53:158, 53:159,
08:67, 08:69, 08:88, 08:223, 08:862, 212
`\hook_new_pair:nn` 08:2735,
16:6, 16:7, 08:199, 08:201, 08:221, 211
`\hook_new_pair_with_args:nn` ... 212
`\hook_new_pair_with_args:nnn` ...
..... 08:2743, 08:199,
08:199, 08:203, 08:219, 08:226, 212
`\hook_new_reversed:n` ... 08:2733,
08:170, 08:172, 08:188, 08:224, 211
`\hook_new_reversed_with_args:nn` .
..... 08:2741, 08:170,
08:170, 08:174, 08:186, 08:197, 1352
`\hook_new_with_args:nn` .. 08:2739,
08:67, 08:67, 08:71, 08:86, 08:100, 212
`\g_hook_patch_action_list_tl` ...
..... 09:6, 09:108, 09:141
`\hook_provide:n` 08:2828, 08:2835
`\hook_provide_pair:nn` 08:2828, 08:2849
`\hook_provide_reversed:n`
..... 08:2828, 08:2842
`\hook_show:n`
..... 08:1819, 08:1824, 08:2813, 214
`\hook_use:n`
08:2213, 08:2213, 08:2215, 08:2235,
08:2237, 08:2255, 08:2257, 08:2312,
08:2803, 16:22, 16:31, 16:55, 16:63,
16:98, 16:105, 48:238, 53:62, 53:65,
53:71, 53:75, 53:123, 08:1466, 212

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx,
16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
26=ltfssstrc.dtx, 27=ltfsscscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
56=lthyphen.dtx, 57=ltfinal.dtx

- \hook_use:nnw 08:2253,
08:2283, 08:2285, 08:2285, 08:2287,
08:2303, 08:2305, 08:2313, 08:2805, 213
- \hook_use_once:n
08:2352, 08:2354, 08:2369, 08:2804, 293
- \hook_use_once:nnw 08:2352, 08:2352,
08:2360, 08:2367, 08:2375, 08:2806, 212
- hook internal commands:
- \c__hook_ 08:1296, 08:1300
- \g__hook_??_code_prop 08:1293
- \c__hook_??_parameter_tl 08:1293
- \g__hook_??_reversed_tl 08:1293
- \g__hook_{hook}_code_prop 230
- \g__hook_{hook}_labels_clist 234
- \g__hook_{hook}_reversed_tl 230
- __hook_print_args:nn 1354
- __hook_activate_generic:n 08:288
- __hook_activate_generic:nn
. 08:2857, 08:291, 08:292, 08:310
- __hook_activate_generic_pair:nn
. 08:2828, 08:2854, 08:2858, 08:2886
- __hook_activate_generic_
reversed:n 08:2828,
08:2847, 08:2856, 08:2859, 08:2879
- \g__hook_all_seq 08:2959,
08:28, 08:108, 08:137, 08:1427, 08:1454
- __hook_apply_rule_>:nnn 08:1794
- __hook_apply_rule_<:nnn 08:1794
- __hook_apply_rule_<:nnn 08:1794
- __hook_apply_rule_>:nnn 08:1794
- __hook_apply_rule_xE:nnn 08:1794
- __hook_apply_rule_xW:nnn 08:1794
- __hook_apply_label_pair:nnn
. 08:1574, 08:1575,
08:1656, 08:1657, 08:1721, 08:1721, 273
- __hook_apply_rule:nnn
. 08:1731, 08:1737, 08:1737, 277
- __hook_apply_rule:nnnN 277
- __hook_apply_rule_>:nnn 08:1772
- __hook_apply_rule_<:nnn 08:1772
- __hook_apply_rule_<:nnn 08:1744
- __hook_apply_rule_>:nnn 08:1744
- __hook_apply_rule_xE:nnn 08:1758
- __hook_apply_rule_xW:nnn 08:1758
- __hook_braced_cs_parameter:n
. 08:2954,
08:1068, 08:1125, 08:1150, 08:1161,
08:1213, 08:1213, 08:1215, 08:1241
- __hook_braced_hidden_loop:w
. 08:1213, 08:1217, 08:1220, 08:1226
- __hook_braced_parameter:n
. 09:238, 09:522, 08:1244,
08:1244, 08:1246, 08:1267, 08:1269,
08:1488, 08:1490, 08:1625, 08:1627
- __hook_braced_real_loop:w 08:1244
- __hook_chk_args_allowed:nn
. 08:2177, 08:502, 08:631,
08:631, 08:633, 08:648, 08:650, 287
- __hook_clear_next:n
08:2181, 08:2197, 08:2202, 08:2203,
08:2203, 08:2205, 08:2208, 08:2210, 285
- __hook_clist_gput:Nn
08:1496, 08:1498, 08:1530, 08:1534,
08:1598, 08:1685, 08:1719, 08:1720
- __hook_cmd_begindocument_code:
09:63, 09:71, 09:76, 09:79, 09:623, 333
- __hook_cmd_if_scanable:Nn 09:467
- __hook_cmd_if_scanable:NnTF
. 09:425, 09:444, 09:467, 328
- __hook_cmd_patch_xparse:Nnn
. 09:145, 09:166, 09:166
- __hook_cmd_try_patch:nn
. 09:69, 09:80, 09:80
- __hook_code_gset:nn 08:2414,
08:118, 08:1090, 08:1090, 08:1092,
08:1104, 08:1106, 08:1108, 08:1485
- __hook_code_gset_aux:nnn
. 08:1053, 08:1090, 08:1093,
08:1095, 08:1097, 08:1098, 08:1111
- __hook_code_gset_auxi:nnnn
. 08:1033, 08:1054,
08:1079, 08:1081, 08:1088, 08:1119, 258
- __hook_cs_end:w 08:1213,
08:1232, 08:1233, 08:1234, 08:1239
- __hook_cs_gput_right:nnn
08:2182, 08:2385, 08:541, 08:1033,
08:1033, 08:1035, 08:1083, 08:1085
- __hook_cs_gput_right_fast:nnn
. 08:1033, 08:1041, 08:1047, 08:1086
- __hook_cs_gput_right_slow:nnn
. 08:1033, 08:1043, 08:1050, 08:1087
- __hook_cs_if_empty:N
. 08:1193, 08:1195, 08:1209, 08:1211
- __hook_cs_if_empty:NTF
. 08:1888, 08:1895, 08:2178,
08:2180, 08:2451, 08:2452, 08:1193
- __hook_cs_if_empty_p:N 08:1193
- __hook_cs_parameter_count:N
. 08:1213, 08:1218, 08:1228
- __hook_cs_parameter_count:w
. 08:1213, 08:1230, 08:1237, 08:1238
- \l__hook_cur_hook_tl 08:29,
08:1560, 08:1642, 08:1778, 08:1789, 279
- __hook_curr_name_pop:
. 08:2794, 08:2900, 08:417, 08:449, 243
- __hook_curr_name_push:n
. 08:2792, 08:2797, 08:417, 08:431, 303

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- `__hook_curr_name_push_aux:n` . . .
 [08:417](#), [08:432](#), [08:433](#)
- `__hook_cururname_or_default:` . . .
 [08:2642](#), [08:2690](#),
 [08:333](#), [08:343](#), [08:347](#), [08:363](#),
 [08:364](#), [08:364](#), [08:538](#), [08:612](#), [240](#)
- `__hook_debug:n`
 [08:2129](#), [08:2143](#), [09:19](#),
 [09:28](#), [09:45](#), [09:56](#), [09:65](#), [09:66](#),
 [09:82](#), [09:86](#), [08:7](#), [08:7](#), [08:20](#),
 [08:531](#), [08:605](#), [08:1426](#), [08:1435](#),
 [08:1453](#), [08:1456](#), [08:1478](#), [08:1502](#),
 [08:1512](#), [08:1540](#), [08:1579](#), [08:1599](#),
 [08:1661](#), [08:1686](#), [08:1746](#), [08:1753](#),
 [08:1760](#), [08:1768](#), [08:1774](#), [08:1785](#), [227](#)
- `\g__hook_debug_bool`
 [08:6](#), [08:10](#), [08:15](#), [08:21](#)
- `__hook_debug_gset:`
 [08:7](#), [08:11](#), [08:16](#), [08:18](#)
- `__hook_debug_label_data:N`
 [08:1807](#), [08:1807](#),
 [08:1579](#), [08:1621](#), [08:1662](#), [08:1710](#)
- `__hook_debug_print_rules:n`
 [08:2107](#), [08:2107](#)
- `__hook_declare_deprecated_-
 generic:NNn` . [08:757](#), [08:778](#), [08:811](#)
- `__hook_declare_deprecated_-
 generic:NNw` . [08:773](#), [08:779](#), [08:780](#)
- `__hook_def_cmd:w`
 [09:13](#), [09:14](#), [09:209](#),
 [09:215](#), [09:284](#), [09:290](#), [09:316](#), [324](#)
- `\g__hook_delayed_patches_prop`
 [09:17](#), [09:68](#), [09:74](#), [09:75](#)
- `__hook_deprecated_generic_-
 warn:n` [08:1839](#),
 [08:1957](#), [08:756](#), [08:763](#), [08:810](#),
 [08:988](#), [08:1024](#), [08:1315](#), [08:1346](#), [251](#)
- `__hook_deprecated_generic_-
 warn:Nn` [08:763](#)
- `__hook_deprecated_generic_-
 warn:Nw` [08:763](#)
- `__hook_deprecated_generic_-
 warn:w` [08:764](#), [08:765](#)
- `__hook_deprecated_warn:nn`
 [08:2830](#), [08:2837](#),
 [08:2844](#), [08:2851](#), [08:2862](#), [08:2869](#),
 [08:2876](#), [08:2883](#), [08:2888](#), [08:2888](#)
- `__hook_disable:n`
 [09:226](#), [09:518](#), [08:264](#), [08:267](#), [08:268](#)
- `__hook_do_deprecated_generic:Nn`
 . [08:1840](#), [08:1958](#), [08:773](#), [08:773](#),
 [08:989](#), [08:1025](#), [08:1316](#), [08:1347](#)
- `__hook_do_deprecated_generic:Nw`
 [08:773](#), [08:774](#), [08:775](#)
- `__hook_double_hashes:n`
 [09:205](#), [09:280](#),
 [09:342](#), [09:342](#), [09:380](#), [08:550](#),
 [08:1065](#), [08:1074](#), [08:1122](#), [08:1179](#), [263](#)
- `__hook_double_hashes:w`
 [09:342](#), [09:343](#),
 [09:344](#), [09:376](#), [09:380](#), [09:383](#), [325](#)
- `__hook_double_hashes_group:n`
 [09:342](#), [09:350](#), [09:379](#)
- `__hook_double_hashes_output:N`
 [09:342](#), [09:347](#), [09:355](#), [326](#)
- `__hook_double_hashes_space:w`
 [09:342](#), [09:351](#), [09:382](#)
- `__hook_double_hashes_stop:w`
 [09:342](#), [09:358](#), [09:378](#)
- `\c__hook_empty_tl` [08:35](#), [08:1285](#)
- `__hook_end_document_label_-
 check:` [08:417](#), [08:456](#), [08:457](#), [08:464](#)
- `__hook_exp_not:n`
 [09:188](#), [09:189](#), [09:193](#), [09:204](#),
 [09:263](#), [09:264](#), [09:268](#), [09:279](#), [322](#)
- `__hook_exp_not:NN` [09:13](#),
 [09:13](#), [09:210](#), [09:216](#), [09:236](#),
 [09:241](#), [09:285](#), [09:291](#), [09:307](#), [09:309](#)
- `__hook_file_hook_normalize:n`
 [08:2338](#), [08:717](#),
 [08:900](#), [08:900](#), [08:903](#), [08:905](#), [254](#)
- `\l__hook_front_tl` [08:1549](#),
 [08:1590](#), [08:1593](#), [08:1596](#), [08:1598](#),
 [08:1599](#), [08:1600](#), [08:1614](#), [08:1615](#),
 [08:1675](#), [08:1679](#), [08:1683](#), [08:1685](#),
 [08:1687](#), [08:1689](#), [08:1703](#), [08:1704](#)
- `\c__hook_generic_⟨type⟩/./⟨place⟩_tl`
 [251](#)
- `\c__hook_generic_class/./after_-
 tl` [08:916](#)
- `\c__hook_generic_class/./before_-
 tl` [08:916](#)
- `\c__hook_generic_cmd/./after_tl`
 [08:916](#)
- `\c__hook_generic_cmd/./before_tl`
 [08:916](#)
- `\c__hook_generic_env/./after_tl`
 [08:916](#)
- `\c__hook_generic_env/./before_tl`
 [08:916](#)
- `\c__hook_generic_env/./begin_tl`
 [08:916](#)
- `\c__hook_generic_env/./end_tl` [08:916](#)
- `\c__hook_generic_file/./after_tl`
 [08:916](#)
- `\c__hook_generic_file/./before_-
 tl` [08:916](#)

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- `\c_hook_generic_include/./after_tl` [08:916](#)
- `\c_hook_generic_include/./before_tl` [08:916](#)
- `\c_hook_generic_include/./end_tl` [08:916](#)
- `\c_hook_generic_package/./after_tl` [08:916](#)
- `\c_hook_generic_package/./before_tl` [08:916](#)
- `__hook_generic_parameter:n` [08:1060](#), [08:1280](#), [08:1291](#)
- `__hook_generic_parameter:w` [08:1281](#), [08:1282](#)
- `\c_hook_generics_file_prop` [08:892](#), [08:939](#)
- `\c_hook_generics_prop` [08:828](#), [08:860](#), [08:916](#), [08:934](#), [08:936](#)
- `\c_hook_generics_reversed_ii_prop` [08:837](#), [08:864](#), [08:939](#)
- `\c_hook_generics_reversed_iii_prop` [08:841](#), [08:868](#), [08:939](#)
- `__hook_gput_code:nnn` [08:486](#), [08:491](#), [08:497](#), [08:500](#), [08:584](#), [08:585](#), [08:663](#)
- `__hook_gput_code_store:nnn` [08:486](#), [08:513](#), [08:516](#)
- `__hook_gput_next_code:nn` [08:2128](#), [08:2142](#), [08:2158](#), [08:2161](#), [08:2161](#), [08:670](#)
- `__hook_gput_next_do:nn` . [08:2167](#), [08:2172](#), [08:2172](#), [08:2174](#), [08:2185](#), [08:2187](#), [08:671](#), [08:692](#), [08:709](#), [248](#)
- `__hook_gput_next_do:Nnn` [08:2189](#), [08:2192](#)
- `__hook_gput_undeclared_hook:nnn` [08:652](#), [08:652](#), [08:664](#), [08:683](#), [08:703](#), [248](#)
- `__hook_gremove_code:nn` .. [08:959](#), [08:962](#), [08:963](#), [08:990](#), [08:998](#), [08:1026](#)
- `__hook_gset_rule:nnnn` [08:1304](#), [08:1306](#), [08:1309](#), [08:1311](#), [08:1316](#), [08:1340](#), [08:1342](#), [08:1347](#)
- `__hook_guess_arg_count:NN` [09:484](#), [09:484](#), [09:486](#), [09:509](#), [09:511](#), [09:517](#)
- `__hook_guess_arg_count:nw` [09:484](#), [09:497](#), [09:502](#), [09:505](#)
- `__hook_guess_arg_count:wN` [09:484](#), [09:488](#), [09:492](#)
- `\c_hook_hash_tl` ... [09:11](#), [09:271](#), [09:272](#), [09:274](#), [09:361](#), [08:1225](#), [262](#)
- `\c_hook_hashes_tl` . [09:11](#), [09:196](#), [09:197](#), [09:199](#), [09:362](#), [08:1152](#), [325](#)
- `\g_hook_hook_curr_name_tl` [08:32](#), [08:366](#), [08:376](#), [08:417](#), [08:429](#), [08:444](#), [08:445](#), [08:452](#), [08:462](#), [08:463](#), [08:484](#), [243](#)
- `__hook_hook_gput_code_do:nnn` ... [08:240](#), [08:257](#), [08:486](#), [08:520](#), [08:529](#), [08:592](#), [08:603](#), [08:655](#)
- `__hook_if_cmd_hook:n` [08:2526](#), [08:2528](#), [08:2542](#)
- `__hook_if_cmd_hook:nTF` [08:2069](#), [08:2526](#), [08:2544](#)
- `__hook_if_cmd_hook:w` [08:2529](#), [08:2530](#)
- `__hook_if_cmd_hook:wTF` [08:2526](#)
- `__hook_if_cmd_hook_p:n` [08:2526](#)
- `__hook_if_cmd_hook_p:w` [08:2526](#)
- `__hook_if_declared:n` [08:2491](#)
- `__hook_if_declared:nTF` . [08:2068](#), [08:2491](#), [09:84](#), [08:75](#), [08:93](#), [08:178](#), [08:207](#), [08:210](#), [08:297](#), [08:315](#), [08:637](#), [08:862](#), [08:1038](#), [08:1056](#), [232](#)
- `__hook_if_declared_p:n` [08:2491](#)
- `__hook_if_deprecated_generic:n` . [08:2514](#)
- `__hook_if_deprecated_generic:nTF` [08:1837](#), [08:1955](#), [08:2506](#), [08:754](#), [08:808](#), [08:986](#), [08:1022](#), [08:1313](#), [08:1344](#)
- `__hook_if_deprecated_generic:w` . [08:2515](#), [08:2516](#)
- `__hook_if_deprecated_generic_p:n` [08:2506](#)
- `__hook_if_disabled:n` [08:273](#)
- `__hook_if_disabled:nTF` [08:1849](#), [08:1861](#), [08:1944](#), [08:1967](#), [08:2047](#), [08:2163](#), [08:264](#), [08:294](#), [08:312](#), [08:524](#), [08:596](#), [232](#)
- `__hook_if_disabled_p:n` [08:264](#)
- `__hook_if_execute_immediately:n` [08:2430](#)
- `__hook_if_execute_immediately:nTF` [08:2356](#), [08:2362](#), [08:2371](#), [08:2430](#), [08:503](#), [08:587](#), [08:1320](#)
- `__hook_if_execute_immediately_p:n` [08:2430](#)
- `__hook_if_file_hook:w` [08:877](#), [08:880](#), [08:882](#)
- `__hook_if_file_hook:wTF` [08:2335](#), [08:713](#), [08:877](#), [249](#)
- `__hook_if_file_hook_p:w` [08:877](#)
- `__hook_if_generic:n` [08:2506](#)
- `__hook_if_generic:nTF` [08:1847](#), [08:1963](#), [08:2506](#), [08:737](#), [08:795](#), [08:1059](#)
- `__hook_if_generic:w` [08:2507](#), [08:2508](#)

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

`__hook_if_generic_p:n` [08:2506](#)
`__hook_if_generic_reversed:n` [08:2546](#)
`__hook_if_generic_reversed:nTF` .
[08:2546](#), [08:302](#), [08:320](#), [08:749](#), [08:803](#)
`__hook_if_generic_reversed:w`
. [08:2547](#), [08:2548](#)
`__hook_if_generic_reversed_p:n` .
. [08:2546](#)
`__hook_if_has_hash:n` [09:328](#)
`__hook_if_has_hash:nTF`
. [09:186](#), [09:261](#), [09:328](#), [325](#)
`__hook_if_has_hash:w`
[09:328](#), [09:329](#), [09:330](#), [09:336](#), [09:338](#)
`__hook_if_has_hash_check:w`
. [09:328](#), [09:335](#), [09:340](#)
`__hook_if_has_hash_p:n` [09:328](#)
`__hook_if_label_case:nmnn`
. [08:2084](#),
[08:1413](#), [08:1413](#), [08:1572](#), [08:1654](#)
`__hook_if_public_command:N` . [09:125](#)
`__hook_if_public_command:NTF`
. [09:94](#), [09:104](#), [318](#)
`__hook_if_public_command:w`
. [09:94](#), [09:128](#), [09:134](#)
`__hook_if_replacing_args:` . [08:2560](#)
`__hook_if_replacing_args:TF`
. [08:2556](#), [08:2562](#), [08:2567](#),
[08:2568](#), [08:2573](#), [08:2574](#), [08:2579](#),
[08:505](#), [08:548](#), [08:559](#), [08:635](#), [08:1072](#)
`__hook_if_reversed:n` [08:2497](#)
`__hook_if_reversed:nTF` . [08:1884](#),
[08:1924](#), [08:1926](#), [08:1985](#), [08:2024](#),
[08:2027](#), [08:2497](#), [08:1494](#), [08:1527](#)
`__hook_if_reversed_p:n` [08:2497](#)
`__hook_if_structure_exist:n` [08:2485](#)
`__hook_if_structure_exist:nTF`
[08:2166](#), [08:2464](#), [08:2485](#), [08:2721](#),
[08:150](#), [08:162](#), [08:965](#), [08:1000](#), [232](#)
`__hook_if_structure_exist_p:n`
. [08:2485](#)
`__hook_if_usable:n` [08:2479](#)
`__hook_if_usable:nTF`
[08:1859](#), [08:1883](#), [08:1942](#), [08:1965](#),
[08:1983](#), [08:2045](#), [08:2130](#), [08:2144](#),
[08:2388](#), [08:2479](#), [08:2827](#), [08:106](#),
[08:518](#), [08:532](#), [08:561](#), [08:590](#),
[08:606](#), [08:739](#), [08:797](#), [08:830](#),
[08:982](#), [08:1018](#), [08:1481](#), [08:1516](#), [256](#)
`__hook_if_usable_p:n`
. [08:1917](#), [08:2017](#), [08:2479](#)
`__hook_if_usable_use:n`
[08:2322](#), [08:2337](#), [08:2340](#), [08:2343](#), [291](#)
`__hook_include_legacy_code_-`
`chunk:n` [08:126](#),

[08:142](#), [08:229](#), [08:229](#), [08:231](#),
[08:249](#), [08:251](#), [08:1480](#), [08:1515](#), [1352](#)
`__hook_init_structure:n` [08:2176](#),
[08:119](#), [08:139](#), [08:146](#), [08:146](#),
[08:148](#), [08:158](#), [08:160](#), [08:540](#),
[08:614](#), [08:654](#), [08:1326](#), [08:1351](#), [231](#)
`__hook_initialize_all:`
. [08:2898](#), [08:1421](#), [08:1421](#),
[08:1423](#), [08:1447](#), [08:1449](#), [08:1471](#)
`__hook_initialize_hook_code:n`
. [08:2231](#), [08:2252](#),
[08:2281](#), [08:1425](#), [08:1452](#), [08:1474](#),
[08:1474](#), [08:1476](#), [08:1508](#), [08:1510](#), [277](#)
`__hook_initialize_single:NNn`
[08:1500](#), [08:1538](#), [08:1554](#), [08:1554](#),
[08:1556](#), [08:1634](#), [08:1636](#), [08:1638](#), [273](#)
`\l__hook_label_0_tl` [08:1549](#)
`__hook_label_if_exist_apply:mnnTF`
[08:1721](#), [08:1723](#), [08:1725](#), [08:1728](#), [277](#)
`__hook_label_ordered:nn` [08:1405](#), [268](#)
`__hook_label_ordered:nmTF`
[08:1368](#), [08:1375](#), [08:1382](#), [08:1405](#), [267](#)
`__hook_label_ordered_p:mn` . [08:1405](#)
`__hook_label_pair:mn`
. [08:1803](#), [08:1804](#),
[08:1367](#), [08:1374](#), [08:1381](#), [08:1386](#),
[08:1391](#), [08:1396](#), [08:1397](#), [08:1397](#), [268](#)
`\l__hook_labels_int` [08:1549](#),
[08:1559](#), [08:1563](#), [08:1595](#), [08:1617](#),
[08:1641](#), [08:1645](#), [08:1681](#), [08:1706](#), [275](#)
`\l__hook_labels_seq`
[08:1809](#), [08:1549](#), [08:1558](#), [08:1564](#),
[08:1582](#), [08:1640](#), [08:1646](#), [08:1665](#)
`__hook_list_if_rule_exists:mnnTF`
. [08:2075](#), [08:2095](#), [08:2096](#), [08:2098](#)
`__hook_list_one_rule:mnn`
. [08:2075](#), [08:2086](#), [08:2087](#), [08:2093](#)
`__hook_list_rules:mn` [08:1904](#),
[08:2004](#), [08:2075](#), [08:2075](#), [08:2112](#), [285](#)
`__hook_log:nN` [08:1819](#),
[08:1822](#), [08:1827](#), [08:1833](#), [08:1835](#),
[08:1840](#), [08:1951](#), [08:1953](#), [08:1958](#)
`__hook_log_cmd:n` [08:1821](#), [08:1826](#),
[08:1830](#), [08:1832](#), [08:1844](#), [08:1962](#)
`__hook_log_line:n` [08:1819](#), [08:1829](#),
[08:1860](#), [08:1862](#), [08:1866](#), [08:1880](#),
[08:1892](#), [08:1902](#), [08:1920](#), [08:1939](#),
[08:1966](#), [08:1968](#), [08:1972](#), [08:1980](#),
[08:1994](#), [08:2002](#), [08:2020](#), [08:2041](#)
`__hook_log_line_indent:n`
. [08:1819](#), [08:1831](#),
[08:1870](#), [08:1876](#), [08:1886](#), [08:1893](#),
[08:1907](#), [08:1915](#), [08:1974](#), [08:1977](#),
[08:1987](#), [08:1995](#), [08:2007](#), [08:2015](#)

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx,
16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
56=ltthyphen.dtx, 57=ltfinal.dtx

`__hook_log_next_code:n` . 08:1999,
[08:2054](#), [08:2054](#), [08:2059](#), [08:2061](#)
`__hook_log_next_code:w`
. [08:1898](#), [08:2057](#)
`__hook_make_name:n`
. [08:357](#), [08:363](#), [08:372](#),
[08:378](#), [08:378](#), [08:432](#), [08:475](#), [230](#)
`__hook_make_name:w`
. [08:378](#), [08:380](#), [08:384](#)
`__hook_make_prefixes:w`
. [09:171](#), [09:224](#), [09:299](#), [09:320](#), [09:325](#)
`__hook_make_usable:n`
. [08:97](#), [08:133](#), [08:318](#), [08:801](#), [08:834](#)
`__hook_make_usable:nn`
. [09:228](#), [09:520](#), [08:81](#),
[08:102](#), [08:102](#), [08:104](#), [08:130](#),
[08:132](#), [08:300](#), [08:744](#), [08:747](#), [250](#)
`__hook_misused_if_replacing_`
`args:nn` . . [08:2556](#), [08:2557](#), [08:2563](#)
`__hook_msg_pair_found:nnn`
. [08:1800](#), [08:1800](#), [08:1746](#), [08:1753](#),
[08:1760](#), [08:1768](#), [08:1776](#), [08:1787](#)
`\g__hook_name_stack_seq`
. [08:32](#), [08:418](#), [08:419](#), [08:423](#),
[08:430](#), [08:444](#), [08:451](#), [08:459](#), [08:469](#)
`__hook_new:n` [08:89](#), [08:91](#), [08:193](#)
`__hook_new:nn` [08:67](#), [08:70](#),
[08:72](#), [08:73](#), [08:90](#), [08:181](#), [08:213](#)
`__hook_new_pair:nnn`
. [08:202](#), [08:204](#), [08:205](#)
`__hook_new_reversed:n` [08:189](#), [08:191](#)
`__hook_new_reversed:nn`
. [08:170](#), [08:173](#),
[08:175](#), [08:176](#), [08:190](#), [08:196](#), [08:214](#)
`__hook_next_□{hook}` [295](#)
`__hook_next_gset:nn`
. [08:2181](#), [08:2206](#), [08:2415](#), [08:154](#),
[08:971](#), [08:1090](#), [08:1096](#), [08:1110](#)
`\c__hook_nine_parameters_tl`
. [08:1875](#),
[08:35](#), [08:114](#), [08:1061](#), [08:1176](#)
`__hook_normalise_code_pool:n`
. [08:120](#), [08:1137](#),
[08:1137](#), [08:1139](#), [08:1189](#), [08:1191](#), [234](#)
`__hook_normalise_cs_args:nn`
. [08:116](#), [08:117](#), [08:1113](#),
[08:1113](#), [08:1115](#), [08:1133](#), [08:1135](#), [234](#)
`__hook_normalise_fn:nn`
. [08:565](#), [08:1143](#), [08:1162](#), [08:1187](#), [260](#)
`__hook_normalize_hook_args:Nn`
. [08:1822](#), [08:1827](#),
[08:2128](#), [08:2142](#), [08:2157](#), [08:2202](#),
[08:2357](#), [08:2363](#), [08:2372](#), [08:2857](#),
[08:70](#), [08:72](#), [08:89](#), [08:173](#), [08:175](#),
[08:189](#), [08:267](#), [08:291](#), [08:385](#), [08:394](#)
`__hook_normalize_hook_args:Nnn`
. [08:202](#), [08:204](#), [08:385](#),
[08:399](#), [08:491](#), [08:497](#), [08:583](#), [08:962](#)
`__hook_normalize_hook_args_`
`aux:Nn`
. [08:385](#), [08:385](#), [08:396](#), [08:401](#), [08:409](#)
`__hook_normalize_hook_rule_`
`args:Nnnnn` . [08:385](#), [08:407](#), [08:1306](#)
`\l__hook_param_text_tl`
. [09:8](#), [09:192](#),
[09:213](#), [09:267](#), [09:288](#), [09:317](#), [321](#)
`__hook_parameter:n`
. [08:1854](#), [08:1101](#), [08:1159](#), [08:1271](#),
[08:1271](#), [08:1273](#), [08:1288](#), [08:1290](#)
`\c__hook_parameter_cmd` [08:951](#), [08:956](#)
`\c__hook_parameter_cmd/./after_`
`tl` [08:951](#)
`\c__hook_parameter_cmd/./before_`
`tl` [08:951](#)
`__hook_parse_dot_label:nN`
. [08:334](#), [08:336](#), [08:336](#)
`__hook_parse_dot_label:w`
. [08:336](#), [08:348](#), [08:351](#)
`__hook_parse_dot_label_aux:w`
. [08:336](#), [08:354](#), [08:362](#)
`__hook_parse_dot_label_cleanup:w`
. [08:336](#), [08:358](#), [08:361](#)
`__hook_parse_label_default:nN`
. [08:330](#), [08:330](#), [08:397](#),
[08:403](#), [08:404](#), [08:411](#), [08:412](#), [08:414](#)
`__hook_patch_check:NNnn`
. [09:94](#), [09:98](#), [09:101](#), [09:104](#), [09:114](#)
`__hook_patch_cmd_or_delay:Nnn`
. [09:33](#),
[09:46](#), [09:59](#), [09:63](#), [09:63](#), [09:73](#), [316](#)
`__hook_patch_command:Nnn`
. [09:73](#), [09:91](#), [09:94](#), [09:94](#), [317](#)
`__hook_patch_debug:n`
. [09:18](#), [09:18](#), [09:96](#), [09:97](#), [09:100](#),
[09:103](#), [09:106](#), [09:175](#), [09:250](#),
[09:389](#), [09:424](#), [09:427](#), [09:428](#),
[09:433](#), [09:443](#), [09:446](#), [09:447](#), [09:452](#)
`__hook_patch_DeclareRobustCommand:Nnn`
. [09:143](#), [09:147](#), [09:147](#), [319](#)
`__hook_patch_DeclareRobustCommand_`
`aux:Nnn` [09:149](#), [09:152](#)
`__hook_patch_expand_redefine:NNnn`
. [09:158](#),
[09:163](#), [09:168](#), [09:171](#), [09:171](#),
[09:173](#), [09:246](#), [09:248](#), [09:387](#), [327](#)
`__hook_patch_newcommand:Nnn`
. [09:144](#), [09:157](#), [09:161](#), [09:161](#), [319](#)

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx,
16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
56=lthyphen.dtx, 57=ltfinal.dtx

- \l__hook_patch_num_args_int
 [09:7](#), [09:176](#),
 [09:181](#), [09:184](#), [09:198](#), [09:228](#),
 [09:237](#), [09:251](#), [09:256](#), [09:259](#),
 [09:273](#), [09:517](#), [09:520](#), [09:540](#), [09:547](#)
- \l__hook_patch_prefixes_tl
 [09:8](#), [09:223](#), [09:298](#), [09:315](#), [324](#)
- __hook_patch_required_catcodes:
 [09:458](#),
 [09:458](#), [09:479](#), [09:555](#), [09:583](#), [332](#)
- __hook_patch_retokenize:Nnnn
 [09:429](#), [09:448](#), [09:513](#),
 [09:513](#), [09:515](#), [09:560](#), [09:562](#), [328](#)
- __hook_post_initialization_
 defs: [08:2308](#), [08:2308](#), [08:2310](#),
 [08:2315](#), [08:2318](#), [08:2320](#), [08:1444](#)
- __hook_preamble_hook:n
 [08:1843](#), [08:1961](#), [08:2213](#),
 [08:2217](#), [08:2228](#), [08:2241](#), [08:2251](#),
 [08:2261](#), [08:2280](#), [08:2289](#), [08:2314](#),
 [08:2347](#), [08:2382](#), [08:2401](#), [08:1467](#), [291](#)
- __hook_print_args:n [08:2064](#)
- __hook_print_args:nn [08:1851](#), [08:2064](#)
- __hook_prop_gput_labeled_
 cleanup:nnn [08:486](#), [08:546](#), [08:556](#)
- __hook_prop_gput_labeled_do:Nnn
 [08:570](#), [08:573](#)
- __hook_prop_gput_labeled_
 do:Nnnn [08:486](#)
- \l__hook_rear_tl
 [08:1549](#), [08:1580](#), [08:1586](#),
 [08:1587](#), [08:1610](#), [08:1611](#), [08:1663](#),
 [08:1671](#), [08:1672](#), [08:1699](#), [08:1700](#)
- __hook_redefine_with_hooks:Nnnn
 [09:171](#), [09:231](#), [09:303](#), [09:313](#), [322](#)
- \l__hook_replace_text_tl
 [09:8](#), [09:193](#), [09:200](#),
 [09:201](#), [09:202](#), [09:207](#), [09:214](#),
 [09:241](#), [09:268](#), [09:275](#), [09:276](#),
 [09:277](#), [09:282](#), [09:289](#), [09:309](#), [321](#)
- __hook_replacement_spec:N
 [08:1197](#), [08:1203](#)
- __hook_replacing_args_false:
 [08:2127](#), [08:2384](#), [08:2556](#), [08:2570](#),
 [08:237](#), [08:490](#), [08:642](#), [08:1430](#), [237](#)
- __hook_replacing_args_reset:
 [08:2137](#), [08:2151](#), [08:2387](#), [08:2556](#),
 [08:2576](#), [08:243](#), [08:492](#), [08:498](#), [08:1433](#)
- __hook_replacing_args_true:
 [08:2141](#),
 [08:2556](#), [08:2564](#), [08:496](#), [08:1431](#)
- \g__hook_replacing_stack_seq [08:2556](#)
- __hook_retokenize_patch:Nnn
 [09:109](#), [09:384](#), [09:384](#)
- \l__hook_return_tl
 [08:2578](#), [08:2579](#), [08:25](#), [08:451](#),
 [08:452](#), [08:459](#), [08:463](#), [08:558](#),
 [08:566](#), [08:571](#), [08:575](#), [08:576](#),
 [08:621](#), [08:624](#), [08:978](#), [08:1013](#),
 [08:1596](#), [08:1597](#), [08:1683](#), [08:1684](#)
- __hook_rollback_tidying: [08:2938](#)
- __hook_rule_<_gset:nnn [08:1364](#)
- __hook_rule_>_gset:nnn [08:1364](#)
- __hook_rule_after_gset:nnn
 [08:1364](#), [08:1371](#), [08:1377](#)
- __hook_rule_before_gset:nnn
 [08:1364](#), [08:1364](#), [08:1370](#), [273](#)
- __hook_rule_gclear:nnn
 [08:1327](#), [08:1352](#), [08:1394](#), [08:1395](#), [304](#)
- __hook_rule_incompatible-error_
 gset:nnn [08:1384](#)
- __hook_rule_incompatible-warning_
 gset:nnn [08:1384](#)
- __hook_rule_unrelated_gset:nnn
 [08:1394](#), [08:1394](#), [268](#)
- __hook_rule_voids_gset:nnn
 [08:1378](#), [08:1378](#)
- __hook_seq_csname:n [08:1814](#),
 [08:1547](#), [08:1548](#), [08:1566](#), [08:1600](#),
 [08:1648](#), [08:1689](#), [08:1749](#), [08:1756](#)
- __hook_set_default_hook_label:n
 [08:2790](#), [08:467](#), [08:467](#)
- __hook_set_default_label:n
 [08:467](#), [08:475](#), [08:477](#)
- __hook_set_normalise_fn:nn
 [08:563](#), [08:1137](#), [08:1141](#), [08:1146](#), [260](#)
- __hook_str_compare:nn
 [08:23](#), [08:23](#), [08:1399](#), [08:1407](#), [08:1416](#)
- __hook_strip_double_slash:n
 [08:900](#), [08:906](#), [08:907](#)
- __hook_strip_double_slash:w
 [08:900](#), [08:908](#), [08:909](#), [08:913](#)
- __hook_tl_csname:n
 [08:1813](#), [08:1547](#), [08:1547](#), [08:1553](#),
 [08:1565](#), [08:1581](#), [08:1584](#), [08:1586](#),
 [08:1590](#), [08:1602](#), [08:1604](#), [08:1607](#),
 [08:1610](#), [08:1615](#), [08:1647](#), [08:1664](#),
 [08:1668](#), [08:1670](#), [08:1675](#), [08:1691](#),
 [08:1693](#), [08:1696](#), [08:1698](#), [08:1704](#),
 [08:1747](#), [08:1748](#), [08:1754](#), [08:1755](#)
- __hook_tl_gclear:N [08:2424](#),
 [08:2425](#), [08:2426](#), [08:64](#), [08:64](#),
 [08:66](#), [08:244](#), [08:259](#), [08:1005](#),
 [08:1006](#), [08:1010](#), [08:1591](#), [08:1676](#)
- __hook_tl_gput:Nn [08:1495](#), [08:1497](#),
 [08:1528](#), [08:1532](#), [08:1597](#), [08:1624](#),
 [08:1684](#), [08:1713](#), [08:1719](#), [08:1719](#), [275](#)

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpara.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

`__hook_tl_gput_left:Nn` 08:58, 08:58, 08:1495, 08:1529
`__hook_tl_gput_right:Nn` 08:2198, 08:55, 08:55, 08:57, 08:615, 08:1497, 08:1533, 08:1626, 08:1715, 08:1717
`__hook_tl_gset:Nn` 08:2197, 08:2411, 08:2949, 08:49, 08:49, 08:51, 08:53, 08:54, 08:56, 08:60, 08:1366, 08:1373, 08:1380, 08:1385, 08:1390, 08:1520
`__hook_tl_gset_eq:NN` 08:63, 08:63, 08:65
`__hook_tl_set:Nn` 08:47, 08:47, 08:1152, 08:1565, 08:1647, 229
`__hook_tmp:w` 08:1874, 08:1877, 08:2079, 08:2101, 08:2110, 08:2121, 08:2947, 08:2964, 08:2965, 09:190, 09:196, 09:197, 09:199, 09:265, 09:271, 09:272, 09:274, 09:470, 09:473, 09:477, 09:529, 09:532, 09:553, 09:565, 09:568, 09:581, 08:34, 08:34, 08:420, 08:424, 08:426, 08:1148, 08:1159, 08:1175, 08:1181, 08:1184, 331
`\l__hook_tmpa_bool` 08:1903, 08:1906, 08:1914, 08:1923, 08:2003, 08:2006, 08:2014, 08:2023, 08:24, 282
`\l__hook_tmpa_tl` .. 09:183, 09:185, 09:187, 09:258, 09:260, 09:262, 09:390, 09:405, 09:408, 09:476, 09:479, 09:521, 09:525, 09:541, 09:548, 09:552, 09:555, 09:580, 09:583, 08:25, 08:430, 08:1160, 08:1182
`\l__hook_tmpb_tl` .. 09:395, 09:406, 09:409, 08:25, 08:1149, 08:1156, 08:1184
`__hook_toplevel_\hook` 231
`__hook_toplevel_gset:nn` 08:2416, 08:153, 08:970, 08:975, 08:1090, 08:1094, 08:1109
`__hook_try_declaring_generic_-hook:nnn` 08:526, 08:598, 08:657, 08:658, 08:660, 08:676, 08:678, 08:697, 08:700, 248
`__hook_try_declaring_generic_-hook:nNnn` 08:702, 08:708, 08:711, 08:711, 248
`__hook_try_declaring_generic_-hook:wn` . 08:731, 08:734, 08:789, 08:792, 08:819, 08:822, 08:851, 08:854
`__hook_try_declaring_generic_-hook:wnTF` 08:662, 08:669, 08:680, 08:689, 08:724, 08:730, 08:783, 251
`__hook_try_declaring_generic_-hook_split:nNnn` 08:711, 08:716, 08:719, 08:722
`__hook_try_declaring_generic_-next_hook:nn` 08:2168, 08:657, 08:667, 08:687, 08:706, 248
`__hook_try_file_hook:n` 08:2322, 08:2330, 08:2333, 291
`__hook_try_patch_with_catcodes:Nnnw` ... 09:402, 09:419, 09:419, 09:421, 09:434, 09:438, 09:440, 09:453, 327
`__hook_try_put_cmd_hook:n` 09:21, 09:21, 09:23, 09:38, 09:40, 09:49, 09:51, 08:743, 08:800, 08:833
`__hook_try_put_cmd_hook:w` 09:21, 09:24, 09:25, 09:41, 09:42, 09:52, 09:53
`__hook_unpatchable_cases:n` 09:608, 09:610
`__hook_update_hook_code:n` 08:2179, 08:2195, 09:243, 09:557, 08:82, 08:303, 08:321, 08:521, 08:593, 08:983, 08:1019, 08:1331, 08:1356, 08:1420, 08:1420, 08:1425, 08:1432, 08:1451, 08:1455, 233
`__hook_use:wn` .. 08:2264, 08:2278, 08:2322, 08:2322, 08:2325, 08:2327, 291
`__hook_use_end:` 08:2272, 08:2275, 08:2282
`__hook_use_i_delimit_by_s_-mark:nw` 08:45, 08:46, 08:1223
`__hook_use_initialized:n` 08:2213, 08:2218, 08:2220, 08:2245, 08:2266, 08:2312, 08:2389, 08:2403, 08:1466, 289
`__hook_use_initialized:nnw` 08:2285, 08:2290, 08:2292, 08:2313
`__hook_use_none_delimit_by_s_-mark:w` 08:2432, 08:2499, 09:506, 08:45, 08:45, 08:1318, 08:1324
`__hook_use_once:n` .. 08:2372, 08:2399
`__hook_use_once:nn` 08:2357, 08:2363, 08:2378, 08:2378, 08:2380, 08:2397, 08:2406
`__hook_use_once_clear:n` 08:2386, 08:2404, 08:2408, 08:2408, 08:2412, 08:2420, 08:2422, 293
`__hook_use_once_set:n` 08:2383, 08:2402, 08:2408, 08:2410, 293
`__hook_use_undefined:w` 08:2270, 08:2274
`\g__hook_used_prop` 08:31, 08:1426, 08:1438, 08:1453, 08:1459, 08:1503, 08:1541
`\l__hook_work_prop` 08:30, 08:566, 08:1142, 08:1144, 08:1178, 08:1499, 08:1536, 08:1561, 08:1568, 08:1570, 08:1579, 08:1596, 08:1621,

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=ltemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=ltthyphen.dtx, 57=ltfinal.dtx

- 08:1643, 08:1650, 08:1652, 08:1662,
08:1682, 08:1710, 08:1781, 08:1792, 273
- hook_□?? internal commands:
 __hook_□?? 265
- hook_□(hook) internal commands:
 __hook_□(hook) 230
- \hookleftarrow 30:491
- \hookrightarrow 30:489
- Hooks:
 build/column/after 54:684
 build/column/before 54:684
 build/page/after 54:682
 build/page/before 54:682
 build/page/reset 54:683
- hook ?? internal commands:
 __hook~?? 08:1293
- \hphantom 38:75
- \hrule 18:397, 18:405, 18:431,
 18:439, 21:309, 21:314, 30:351,
 30:629, 40:202, 40:207, 40:280,
 40:290, 41:359, 41:376, 42:589,
 42:599, 45:395, 02:502, 02:546, 1355
- \hrulefill 06:884, 06:905, 02:546
- \hspace 1324
- \hskip 1283
- \hskip... 1320
- \hspace 18:512, 18:516, 18:522, 448
- \hss 1144
- \Hwithstroke 21:515, 21:1235, 1335
- \hwithstroke 21:531, 21:1236, 1335
- \hyphenation 21:225, 1307
- \hyphenchar 37:564, 06:842, 06:845,
 06:854, 06:857, 06:860, 06:865, 1333
- \hyphenpenalty 24:759, 24:791, 02:321
- I**
- \I 50:1335, 50:1467,
 50:1557, 50:1577, 57:255, 57:550, 02:437
- \i 21:267, 21:422,
 21:472, 21:473, 21:474, 21:475,
 21:476, 21:477, 21:478, 21:568,
 21:608, 21:609, 21:701, 21:703,
 21:705, 21:707, 21:735, 21:800,
 21:1162, 21:1317, 21:1319, 21:1321,
 21:1323, 21:1374, 21:1377, 21:1380,
 21:1383, 21:1453, 57:259, 57:554, 1326
- \ialign 30:348,
 30:470, 30:541, 30:544, 30:548,
 30:551, 38:168, 38:170, 38:189,
 41:191, 42:141, 02:534, 02:536, 1285
- \IeC 57:357, 57:361, 57:468
- \if 826
- if commands:
 \if:w
 08:2433, 08:2447, 08:2500, 08:2532,
 09:360, 08:1037, 08:1197, 08:1222, 294
- \if_case:w
 .. 07:1492, 08:1248, 08:1399, 08:1416
- \if_catcode:w 09:369,
 07:1822, 07:1823, 07:1824, 07:1825, 326
- \if_charcode:w 08:2550, 07:980
- \if_cs_exist:w 08:2100,
 08:2222, 08:2230, 08:2247, 08:2268,
 08:2295, 08:2434, 08:768, 08:1730
- \if_false:
 07:899, 07:902, 07:2654, 07:2658
- \if_int_compare:w 16:101, 08:1407
- \if_meaning:w
 09:357, 09:361, 09:362, 52:241, 07:1337
- \if_mode... 422
- \if_mode_horizontal: ... 16:32, 16:64
- \if... 1091
- \IfBlankF 07:3254
- \IfBlankT 07:3254
- \IfBlankTF 47:19, 07:3254, 1348
- \IfBold 711
- \IfBooleanF 22:135, 22:143, 07:3229
- \IfBooleanT 07:3229
- \IfBooleanTF 17:32,
 35:67, 35:69, 35:164, 07:3229, 196
- \IfClassAtLeastF 50:283
- \IfClassAtLeastT 50:283
- \IfClassAtLeastTF 1041
- \IfClassAtLeastTF
 50:165, 50:293, 50:294, 50:322, 50:323
- \IfClassLoadedF 50:283
- \IfClassLoadedT 50:283
- \IfClassLoadedTF 1041
- \IfClassLoadedTF
 50:263, 50:289, 50:290, 50:318, 50:319
- \IfClassLoadedWithOptionsF 50:283
- \IfClassLoadedWithOptionsT 50:301, 50:330
- \IfClassLoadedWithOptionsTF 1041
- \IfClassLoadedWithOptionsTF
 50:263, 50:283
- \ifcoremisses 33:1186, 33:1377
- \ifcsname 21:11, 21:210, 24:403,
 25:2828, 25:2831, 25:3173, 25:3176,
 26:129, 28:154, 29:43, 29:54, 29:76,
 29:87, 33:1212, 50:1220, 50:1355,
 06:740, 06:757, 53:433, 54:2403, 1357
- \ifdefined 02:570, 02:634,
 05:22, 29:624, 05:71, 05:72, 05:73,
 05:74, 05:117, 05:118, 05:119, 57:319
- \IfDocumentMetadataTF 17:5, 17:17
- \IfExplAtLeastTF 78

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

40:490, 40:507, 40:524, 41:57, 41:62, 41:68, 41:83, 41:92, 41:105, 41:109, 41:116, 41:123, 41:125, 41:134, 41:154, 41:237, 41:301, 41:303, 41:305, 41:332, 42:36, 42:46, 42:66, 42:75, 42:109, 42:120, 42:131, 42:143, 42:148, 42:154, 43:43, 43:45, 44:110, 45:17, 45:24, 45:477, 45:495, 45:513, 47:7, 47:9, 53:330, 675	18:267, 18:286, 18:320, 18:354, 18:380, 18:413, 18:471, 18:482, 18:489, 18:495, 18:501, 18:507, 18:515, 18:521, 18:535, 18:541, 20:10, 20:84, 20:142, 20:220, 20:257, 20:278, 20:293, 20:354, 20:403, 20:464, 20:474, 20:500, 20:532, 20:554, 20:572, 20:582, 20:598, 20:623, 20:628, 20:636, 20:650, 20:661, 20:677, 20:710, 21:95, 21:122, 21:166, 21:187, 21:344, 21:352, 21:373, 21:389, 04:235, 04:258, 04:281, 22:26, 22:32, 22:40, 22:61, 22:80, 22:90, 22:96, 22:122, 22:130, 22:152, 22:179, 22:214, 22:230, 22:238, 22:256, 23:5, 23:11, 23:19, 23:26, 24:25, 24:54, 24:206, 24:228, 24:273, 24:295, 24:347, 24:358, 24:368, 24:424, 24:439, 24:517, 24:528, 24:536, 24:568, 24:576, 24:590, 24:611, 24:647, 24:734, 24:797, 25:3, 25:1432, 25:2773, 25:2780, 25:2792, 25:2804, 25:2812, 25:2894, 25:2910, 25:2931, 25:2946, 25:3048, 25:3129, 25:3133, 25:3143, 25:3150, 25:3158, 25:3230, 25:3249, 25:3255, 25:3259, 26:114, 26:159, 26:162, 26:542, 26:551, 27:2, 27:22, 28:50, 28:79, 28:139, 28:181, 28:211, 28:242, 28:276, 28:323, 28:414, 28:427, 28:433, 28:473, 28:521, 28:548, 28:573, 28:806, 28:854, 28:900, 28:909, 28:1093, 28:1104, 29:34, 29:69, 29:102, 29:127, 05:2, 29:229, 29:253, 05:11, 29:302, 29:349, 05:20, 29:378, 29:396, 29:436, 29:470, 29:499, 29:532, 29:544, 29:578, 29:591, 29:622, 29:631, 05:58, 29:656, 29:673, 29:689, 29:706, 30:75, 30:93, 30:112, 30:121, 05:112, 05:133, 30:633, 30:645, 05:148, 32:27, 32:34, 05:159, 05:169, 05:178, 05:192, 33:571, 05:199, 05:204, 06:5, 06:13, 35:12, 35:29, 35:46, 35:62, 35:76, 35:96, 35:112, 35:116, 35:146, 06:56, 35:155, 35:167, 35:177, 35:189, 35:200, 06:62, 36:264, 37:10, 37:76, 37:140, 37:182, 37:187, 37:199, 37:226, 37:245, 37:267, 37:295, 37:312, 37:334, 37:345, 37:361, 37:375, 37:383, 37:394, 37:402, 37:415, 37:421, 37:433, 37:438, 37:448, 37:467, 37:484, 37:500,
<code>\ignorespacesafterend</code> 37:7	
<code>\IJ</code> 21:270, 21:456, 21:571, 21:1163, 57:656, 1346	
<code>\ij</code> 21:269, 21:454, 21:570, 21:1164, 57:656, 1328	
<code>\Im</code> 30:325	
<code>\imath</code> 30:320	
<code>\immediate</code> 225	
<code>\in</code> 30:440, 30:472	
in commands:	
<code>in_callback</code> 04:959	
<code>\in_callback</code> 46	
<code>\include</code> 451	
<code>\include</code> 20:218, 20:266, 20:268, 20:284, 20:286, 20:345, 20:397, 1102	
<code>include/.../after</code> 1103	
<code>include/.../before</code> 1103	
<code>include/.../end</code> 1103	
<code>include/after</code> 1103	
<code>include/before</code> 1103	
<code>include/end</code> 1103	
<code>\IncludeInRelease</code> 08:1833, 08:1951, 02:581, 08:2054, 08:2059, 08:2123, 08:2154, 08:2172, 08:2185, 08:2203, 08:2208, 08:2213, 08:2235, 08:2255, 08:2285, 02:615, 08:2303, 08:2308, 08:2318, 08:2322, 08:2325, 08:2352, 08:2367, 02:622, 08:2378, 08:2397, 08:2408, 08:2420, 08:2443, 08:2460, 08:2526, 08:2542, 08:2599, 02:648, 08:2628, 08:2736, 08:2745, 08:2751, 08:2758, 08:2763, 08:2770, 08:2774, 08:2781, 08:2801, 08:2808, 08:2902, 02:688, 02:693, 09:21, 09:38, 09:49, 02:703, 09:171, 09:246, 09:419, 09:438, 09:484, 09:509, 09:513, 09:560, 09:587, 09:590, 09:620, 03:71, 10:200, 03:180, 03:184, 03:188, 03:192, 11:1219, 14:138, 14:158, 14:192, 14:205, 04:3, 16:12, 16:45, 16:167, 17:20, 17:44, 17:58, 18:5, 18:22, 18:55, 18:66, 18:82, 18:87, 18:99, 18:105, 18:133, 18:153, 18:167, 18:179, 18:192, 18:207, 18:236, 18:251,	

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpara.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=ltthyphen.dtx, 57=ltfinal.dtx

37:511, 37:525, 37:547, 37:574,
 37:590, 37:603, 37:617, 37:629,
 37:635, 37:641, 37:648, 37:666,
 37:679, 37:695, 37:718, 37:728,
 01:274, 38:79, 38:87, 38:109, 38:118,
 38:137, 38:144, 38:152, 38:157,
 38:165, 38:176, 38:216, 38:233,
 38:272, 38:280, 38:290, 38:317,
 38:399, 38:408, 38:441, 38:449,
 38:462, 38:474, 38:486, 38:495,
 06:218, 06:225, 39:125, 39:135,
 39:143, 39:162, 40:4, 40:14, 40:42,
 40:58, 40:71, 40:91, 40:112, 40:122,
 40:136, 40:142, 40:147, 40:156,
 40:164, 01:21, 40:220, 40:228,
 40:241, 40:261, 40:299, 40:308,
 40:327, 06:284, 40:355, 40:378,
 40:400, 40:477, 40:495, 40:511,
 40:531, 40:537, 40:557, 40:565,
 40:595, 40:606, 01:291, 41:60, 41:65,
 41:137, 41:157, 06:326, 41:222,
 41:227, 42:10, 42:16, 42:26, 06:354,
 42:39, 42:55, 42:69, 42:80, 42:89,
 42:100, 42:112, 42:146, 42:151,
 42:160, 42:172, 42:235, 42:250,
 06:382, 42:311, 06:389, 42:369,
 42:455, 42:463, 42:472, 42:500,
 42:530, 06:406, 42:557, 42:571,
 42:584, 42:595, 42:607, 42:624,
 42:644, 42:651, 06:422, 42:685,
 42:745, 42:803, 42:819, 43:28, 43:34,
 43:50, 43:57, 44:5, 44:20, 44:161,
 44:168, 44:174, 44:182, 44:196,
 44:211, 44:218, 44:227, 44:250,
 06:460, 45:35, 45:105, 06:472,
 45:206, 45:232, 45:280, 45:294,
 45:335, 45:352, 45:406, 45:412,
 45:420, 45:427, 45:434, 45:440,
 45:446, 45:463, 45:482, 45:499,
 06:502, 45:549, 45:559, 06:513,
 47:14, 47:25, 47:61, 47:77, 06:538,
 06:546, 06:558, 06:565, 48:503,
 49:20, 49:35, 49:61, 49:82, 49:108,
 49:116, 50:18, 50:23, 50:36, 50:51,
 50:63, 50:87, 50:95, 50:101, 50:126,
 50:144, 50:167, 50:175, 50:187,
 50:200, 50:226, 50:245, 50:265,
 50:273, 50:285, 50:313, 50:345,
 50:361, 50:375, 50:393, 50:406,
 50:422, 50:442, 50:465, 50:481,
 50:510, 50:524, 50:556, 50:574,
 50:594, 50:612, 50:631, 50:641,
 50:651, 50:663, 50:695, 50:706,
 50:720, 50:730, 50:776, 06:664,
 50:790, 50:805, 50:835, 50:862,
 50:894, 06:681, 01:26, 50:1046,
 50:1126, 50:1133, 50:1144, 50:1158,
 50:1208, 50:1343, 50:1474, 06:730,
 06:736, 06:766, 52:5, 52:15, 52:27,
 52:89, 52:144, 52:181, 52:192,
 52:205, 06:807, 52:249, 52:257,
 52:268, 52:299, 52:322, 52:343,
 52:357, 52:365, 06:826, 52:389,
 52:408, 52:433, 52:479, 52:495,
 06:837, 52:506, 52:533, 06:854,
 53:3, 06:865, 06:876, 06:896, 53:430,
 53:470, 53:484, 54:20, 54:50, 54:149,
 54:177, 54:343, 54:364, 54:369,
 54:417, 54:481, 54:687, 54:790,
 54:870, 54:929, 54:1068, 54:1086,
 54:1147, 54:1168, 02:49, 54:1204,
 54:1228, 54:1340, 54:1492, 54:1636,
 54:1805, 54:1888, 54:1968, 54:2062,
 54:2184, 54:2398, 54:2421, 54:2435,
 54:2463, 07:227, 54:2525, 54:2530,
 07:241, 54:2648, 54:2698, 54:2759,
 54:2777, 54:2796, 54:2826, 54:2867,
 55:305, 57:8, 57:16, 57:23, 57:30,
 57:37, 57:52, 57:71, 57:80, 57:87,
 57:106, 57:143, 57:166, 57:199,
 57:288, 57:293, 57:314, 57:339,
 57:348, 57:441, 02:88, 02:103,
 02:119, 02:125, 02:134, 02:139,
 02:148, 02:154, 07:1125, 07:1177,
 07:1180, 07:1195, 07:1208, 02:168,
 07:1223, 02:182, 07:1341, 07:1344,
 02:186, 07:1373, 07:1376, 07:1385,
 07:1393, 07:1396, 07:1557, 02:220,
 02:228, 02:233, 02:244, 07:2177,
 07:2190, 07:2221, 07:2233, 07:2249,
 02:289, 02:351, 02:361, 07:3107,
 07:3160, 07:3254, 07:3260, 07:3276,
 08:67, 08:86, 08:102, 08:130, 08:146,
 08:158, 08:170, 08:186, 08:199,
 08:219, 08:229, 08:249, 08:264,
 08:282, 08:288, 08:308, 08:326,
 02:454, 08:486, 08:580, 08:631,
 08:648, 08:657, 08:675, 08:696,
 08:730, 02:472, 08:788, 08:818,
 08:850, 08:877, 08:880, 08:900,
 08:903, 08:916, 08:934, 08:939,
 08:942, 08:951, 08:956, 08:959,
 08:996, 08:1033, 08:1083, 08:1090,
 08:1106, 08:1113, 08:1133, 08:1137,
 08:1189, 08:1193, 08:1209, 08:1213,
 08:1241, 08:1244, 08:1267, 08:1271,
 08:1288, 08:1296, 08:1300, 08:1309,
 08:1340, 08:1421, 08:1447, 08:1474,

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
 06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpara.dtx,
 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
 56=ltthyphen.dtx, 57=ltfinal.dtx

- 08:1508, 08:1554, 08:1636, 02:557, 76
- `\includeonly` 451
- `\includeonly` 20:218,
20:258, 20:260, 20:279, 20:280, 459
- `\indent` 18:538, 39:192, 41:81, 1321
- `\IndentBox` 16:91, 16:171, 415
- `\index` 995
- `\index` 44:191, 44:200, 46:6, 46:18, 48:267,
49:40, 49:51, 49:66, 49:74, 49:89,
49:97, 54:828, 54:898, 54:957, 1004
- `\indexentry` 46:15
- `\inf` 38:25
- `\infty` 30:327
- `\initcatcodetable` 04:91
- `\input` 451, 1042
- `\input` 04:18, 20:633, 01:158,
01:161, 26:16, 27:106, 01:218,
29:752, 29:762, 29:772, 30:10,
30:11, 30:12, 30:13, 30:14, 30:23,
30:42, 30:43, 30:44, 30:48, 30:49,
30:50, 30:51, 30:52, 30:53, 30:58,
30:147, 30:148, 30:149, 30:150,
05:108, 05:130, 30:665, 30:666,
30:667, 06:22, 33:1090, 50:680,
57:164, 57:178, 57:203, 57:281,
57:325, 57:405, 57:662, 01:52, 1100
- `\input@path` 6, 1
- `input@path` commands:
- `\input@path:` 01:223
- `\inputencodingname`
..... 57:382, 57:404, 57:486, 1334
- `\InputIfFileExists` 451, 1042
- `\InputIfFileExists` 17:7, 20:616, 20:639,
20:654, 20:664, 20:674, 20:728,
21:1568, 24:488, 29:744, 29:754,
29:764, 33:806, 33:1173, 50:1011,
50:1077, 52:143, 56:8, 57:275, 1107
- `\inputlineno` 14:214, 01:311, 1294
- `\insert` 45:465,
45:484, 45:502, 54:768, 54:769,
54:2383, 02:206, 02:254, 02:279,
02:281, 02:284, 02:299, 02:332, 411
- `\InsertMark` 48:385, 48:507, 227
- `insertmark` 48:264, 1004
- `\int` 30:359
- `int` commands:
- `\int_add:Nn` 07:610, 07:1286
- `\int_case:nnTF` 55:14, 55:24
- `\int_compare:nNnTF` 08:2066, 08:2072,
08:2391, 08:2588, 09:181, 09:256,
09:392, 09:504, 10:39, 11:45, 28:288,
28:290, 28:342, 48:80, 48:269, 53:55,
53:111, 53:142, 53:353, 53:359,
53:367, 55:197, 55:206, 55:229,
55:248, 55:251, 55:276, 55:283,
55:294, 07:460, 07:605, 07:896,
07:938, 07:2161, 07:2227, 07:2242,
07:2510, 07:2512, 08:1584, 08:1606,
08:1617, 08:1667, 08:1695, 08:1706
- `\int_compare:nTF` 11:191
- `\int_compare_p:nNn`
..... 08:2935, 08:2936, 07:965
- `\int_const:Nn` 10:37
- `\int_decr:N` 07:539,
07:585, 07:2160, 08:1595, 08:1681
- `\int_div_truncate:nn` 07:1435
- `\int_eval:n` 08:1853, 11:362,
05:163, 53:355, 07:2524, 08:1249,
08:1603, 08:1692, 08:1748, 08:1755
- `\int_gadd:Nn` 55:186, 55:289
- `\int_gdecr:N` 52:559, 55:260
- `\int_gincr:N` 17:37, 48:270, 52:549,
53:87, 53:102, 55:240, 55:257, 07:1109
- `\int_gset:Nn`
... 24:415, 28:295, 28:347, 52:545,
55:199, 55:244, 55:266, 55:267, 55:286
- `\int_gzero:N` 48:271,
55:235, 55:236, 55:241, 55:254, 55:263
- `\int_if_odd:nTF` 48:425, 48:485
- `\int_if_zero:nTF` 10:33
- `\int_incr:N` 07:363, 07:483,
07:910, 07:991, 07:998, 07:1253,
07:1553, 07:2170, 08:1563, 08:1645, 149
- `\int_new:N` 09:7, 11:34, 17:23,
48:264, 52:544, 53:346, 53:348,
07:16, 07:17, 07:29, 07:38, 55:35,
55:161, 55:162, 55:177, 07:2140, 08:1550
- `\int_set:Nn` . 09:176, 09:219, 09:251,
09:294, 09:464, 09:465, 09:495,
11:190, 52:212, 52:226, 07:1131,
07:1142, 07:1350, 07:1360, 07:1478,
07:2144, 07:2670, 07:2696, 08:1165
- `\int_set_eq:NN` 48:69, 07:793
- `\int_step_function:nN` 07:1433
- `\int_step_inline:nn` 48:353
- `\int_step_inline:nnn`
..... 09:184, 09:198, 09:259, 09:273
- `\int_use:N` 09:237,
09:540, 09:547, 10:63, 10:89, 10:106,
17:38, 28:282, 28:283, 28:285,
28:290, 28:330, 28:331, 28:333,
28:342, 36:233, 36:243, 36:244,
48:105, 48:107, 48:241, 52:553,
53:104, 53:116, 53:363, 53:371,
07:27, 55:15, 55:25, 55:210, 55:220,
55:221, 55:272, 55:273, 07:941,
07:1113, 07:1142, 07:1271, 07:1275,
07:1276, 07:1360, 07:1544, 1129

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- `\int_value:w` 53:48, 53:137
`\int_zero:N` 55:42,
 07:354, 07:448, 07:794, 07:798,
 07:945, 07:1245, 08:1559, 08:1641
`\c_max_int`
 48:69, 53:214, 53:240, 53:265, 53:292
`\l_tmpa_int` 354
`\c_zero_int` 09:181, 09:256, 53:91
int internal commands:
`\g_mark_int`
 48:241, 48:264, 48:269, 48:270, 48:271
`\interdisplaylinepenalty`
 18:14, 38:55, 38:207, 38:393
`\interfootlinepenalty` 02:427
`\interfootnotelinepenalty`
 18:19, 45:467, 45:486, 45:504, 02:427
`\interlinepenalty` 18:11,
 24:761, 37:536, 37:539, 37:557,
 37:560, 44:67, 44:118, 44:234,
 44:257, 45:467, 45:486, 45:504,
 54:336, 54:1428, 54:1432, 54:1577,
 54:1581, 54:1738, 54:1742, 02:331, 1282
`\inteval` 77
`\inteval` 05:158, 05:173, 77
`\intextsep`
 54:1411, 54:1415, 54:1430, 54:1433,
 54:1440, 54:1560, 54:1564, 54:1579,
 54:1582, 54:1589, 54:1717, 54:1723,
 54:1740, 54:1743, 54:1752, 54:2927
`\intop` 30:358, 30:359
`\iota` 30:287
iow commands:
`\iow_char:N`
 08:2611, 08:2613, 08:2615,
 08:2620, 08:2623, 08:2625, 08:2635,
 08:2636, 08:2676, 08:2684, 08:2696,
 08:2701, 08:2706, 08:2892, 08:2894,
 09:598, 16:153, 16:160, 16:161,
 16:162, 16:163, 07:471, 07:535,
 07:566, 07:638, 07:712, 07:724,
 07:755, 07:1152, 07:1158, 07:2145,
 07:2989, 07:3067, 08:1778, 08:1789
`\iow_log:n` 08:1821
`\iow_newline:`
 52:557, 07:1403, 07:1423,
 07:1432, 07:1450, 07:1466, 07:1551
`\iow_now:Nn` 53:362
`\iow_term:n` 08:1802,
 08:1808, 08:1809, 08:1810, 08:1813,
 08:1817, 08:1826, 08:2109, 08:2114,
 08:2129, 08:2143, 09:19, 09:28,
 09:45, 09:56, 09:65, 09:67, 09:83,
 09:87, 10:25, 48:20, 48:91, 48:165,
 48:171, 48:177, 48:244, 52:550, 52:583, 07:1466, 08:531, 08:605,
 08:1437, 08:1440, 08:1458, 08:1461,
 08:1479, 08:1513, 08:1599, 08:1619,
 08:1620, 08:1622, 08:1686, 08:1708,
 08:1709, 08:1711, 08:1777, 08:1788
`\ishortstack` 42:132
`\itdefault` 25:3217, 29:31, 30:106
`\item` ... 14:282, 37:444, 37:494, 37:496,
 37:527, 37:549, 38:469, 38:481,
 38:508, 39:172, 39:250, 41:78, 43:53,
 43:55, 43:60, 43:62, 47:4, 47:8, 355
`\itemindent`
 39:9, 39:42, 39:95, 39:218, 39:239, 861
`\itemitem` 1282
`itemize (env.)` 39:273
`\itemize` 39:273
`\itemsep` 39:1, 39:207, 862
`\iterate` 01:65, 01:66, 02:490
`\itshape` 21:467, 21:829, 25:3215,
 25:3216, 29:29, 29:30, 29:555,
 29:588, 29:594, 32:21, 33:598, 43:53,
 43:55, 43:60, 43:62, 45:399, 713

J

`\J` 57:257, 57:552
`\j` 21:268, 21:423, 21:569,
 21:801, 21:1172, 21:1387, 21:1473
`\jmath` 30:321
`\jobname` 1110
`\Join` 29:726
`\joinrel` 30:482, 30:489,
 30:491, 30:493, 30:495, 30:497,
 30:499, 30:501, 30:503, 30:507, 30:509
`\jot` 38:53, 38:204, 38:404, 38:414

K

`\k` 21:506, 21:611, 21:616,
 21:638, 21:643, 21:719, 21:720,
 21:779, 21:780, 21:834, 21:836,
 21:841, 21:843, 21:1273, 21:1341,
 21:1342, 21:1359, 21:1360, 21:1382,
 21:1383, 21:1384, 21:1437, 21:1438,
 21:1471, 21:1472, 33:145, 33:168, 1327
`\kanjiskip` 05:74
`\kappa` 30:288
`\ker` 38:27
`\kern` 1297
`\kern...` 1320
kernel (refstepcounter/target) (plug)
 35:120
kernel (tagssupport/recordtarget)
 (plug) 55:47
kernel internal commands:
`_kernel_chk_if_free_cs:N`
 07:3273, 198

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx,
16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
56=lthyphen.dtx, 57=ltfinal.dtx

- __kernel_cmd_if_xparse:NTF 09:145, 07:1162,
07:1473, 07:1475, 07:2728, 07:2749, 154
 - __kernel_cs_parameter_spec:N 09:178,
09:253, 09:386, 05:142, 07:2732, 1354
 - __kernel_cs_parm_from_arg_-
count:nnTF 08:109
 - __kernel_exp_not:w 08:48, 08:50, 08:56, 08:61
 - \l__kernel_expl_bool 09:397
 - __kernel_file_name_sanitiz:e:n 52:98
 - __kernel_msg... 1344
 - __kernel_msg... 1346
 - __kernel_quark_new_conditional:Nn
. 11:41
 - __kernel_quark_new_test:N . . . 07:51
 - \g__kernel_target_int 17:23, 17:37, 17:38
 - \kerneltmpDoNotUse . . . 09:457, 09:469,
09:475, 09:480, 09:528, 09:535,
09:556, 09:564, 09:571, 09:584, 329
 - keys commands:
 - \keys_define:nn . . 11:512, 11:533,
11:594, 20:744, 51:73, 51:82, 51:180,
51:188, 51:192, 51:209, 51:244, 57:572
 - \keys_if_exist:nnTF 51:71, 51:121, 51:127, 51:151
 - \l_keys_key_str 51:48, 51:49, 51:78, 51:197
 - \l_keys_key_tl 51:48
 - \l_keys_path_str 51:47
 - \keys_set:nn 11:1215, 51:86, 51:259, 57:594
 - \keys_set_known:nnN 11:1208
 - \l_keys_usage_load_prop 51:205
 - \l_keys_usage_preamble_prop . 51:240
 - keys internal commands:
 - \l__keys_class_only_clist 51:54, 51:124, 51:153
 - __keys_find_key_module:wNN . . 51:46
 - \l__keys_forced_global_clist 51:49, 51:55, 51:155
 - \l__keys_no_value_bool 51:24
 - __keys_option_end: 51:62, 51:81, 51:89
 - __keys_options:n 51:58, 51:58, 51:201
 - __keys_options_aux:n 51:58, 51:59, 51:60
 - __keys_options_class:n 51:103, 51:107, 51:107
 - __keys_options_class:nn 51:107, 51:123, 51:128, 51:135
 - __keys_options_class:nnn 51:107, 51:113, 51:119
 - \l__keys_options_clist 51:56, 51:63,
51:85, 51:138, 51:163, 51:172, 1094
 - __keys_options_end: 51:58
 - __keys_options_expand_module:Nn
. 51:59,
51:181, 51:181, 51:188, 51:192, 51:259
 - __keys_options_expand_module:nN
. 51:181, 51:185
 - __keys_options_global:n 51:66, 51:98, 51:98
 - __keys_options_loaded:n 51:90, 51:203, 51:203
 - __keys_options_loaded:nn 51:203, 51:212, 51:217
 - \l__keys_options_loading_bool 51:57, 51:84, 51:87, 51:219, 1093
 - __keys_options_local: 51:70, 51:166, 51:166
 - __keys_options_package:n 51:104, 51:140, 51:140
 - __keys_options_package:nn 51:140, 51:156, 51:158, 51:161
 - __keys_options_package:nnn 51:140, 51:144, 51:149
 - \c__keys_props_root_str 51:9, 51:10, 51:22
 - __keys_remove_equals:n 51:114, 51:145, 51:177, 51:177
 - __keys_remove_equals:w 51:177, 51:178, 51:179
 - __keys_scope:N 51:22, 51:33, 51:35, 51:44
 - __keys_scope:n 51:22, 51:25, 51:26, 51:28
 - __keys_tmp:nn 51:5, 51:11, 51:13
 - \l__keys_tmpa_tl 51:205, 51:207
- keyval commands:
 - \keyval_parse:NNn 11:212, 11:617
 - \keyval_parse:nnn 11:413, 11:534
 - \KeyValue 11:79, 11:881, 11:1086, 11:1199, 1360
 - \kill 41:154, 41:162

L

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=ltthyphen.dtx, 57=ltfinal.dtx

- `label` [36:236](#), [809](#) [57:266](#), [57:536](#), [57:544](#), [57:551](#),
`\labelenumi` [875](#) [57:553](#), [57:554](#), [57:556](#), [57:558](#), [1327](#)
`\labelenumiv` [875](#) [57:553](#), [57:554](#), [57:556](#), [57:558](#), [1327](#)
`\labelformat` [798](#) [57:553](#), [57:554](#), [57:556](#), [57:558](#), [1327](#)
`\labelitemi` [35:157](#),
[35:185](#), [35:190](#), [35:196](#), [35:201](#), [35:207](#)
`\labelitemii` [875](#)
`\labelitemiii` [875](#)
`\labelitemiv` [875](#)
`\labelsep` [39:9](#), [39:241](#),
[39:247](#), [43:53](#), [43:55](#), [43:60](#), [43:62](#), [875](#)
`\labelwidth`
[39:9](#), [39:93](#), [39:240](#), [39:242](#), [39:245](#), [862](#)
`\Lambda` [30:311](#)
`\lambda` [30:289](#)
`\land` [30:379](#), [30:381](#)
`\angle` [30:605](#)
`\language` [20:52](#), [20:122](#),
[37:532](#), [37:724](#), [02:35](#), [54:799](#),
[54:875](#), [56:10](#), [02:82](#), [02:84](#), [02:99](#), [1333](#)
`\lastbox` [24:779](#),
[38:193](#), [38:194](#), [39:132](#), [39:138](#),
[39:216](#), [44:99](#), [44:132](#), [54:303](#), [418](#)
`\LastDeclaredEncoding`
..... [24:138](#), [24:141](#), [57:482](#), [1324](#)
`\LastMark` [48:388](#), [48:509](#), [1005](#)
`\lastnamedcs` [06:758](#)
`\lastnodetype` [24:772](#), [24:773](#), [24:774](#), [24:778](#)
`\lastpenalty` [24:775](#), [32:112](#), [32:115](#)
`\lastskip` [18:45](#), [18:129](#), [18:141](#),
[18:160](#), [18:221](#), [18:222](#), [18:226](#),
[18:228](#), [18:229](#), [18:241](#), [18:256](#),
[18:273](#), [18:295](#), [18:298](#), [18:328](#),
[18:331](#), [18:362](#), [18:365](#), [18:366](#),
[32:102](#), [32:105](#), [39:115](#), [39:116](#),
[39:181](#), [39:182](#), [42:123](#), [54:562](#),
[02:514](#), [02:515](#), [02:517](#), [02:519](#), [1283](#)
`\LaTeX` ... [19:3](#), [19:15](#), [50:1294](#), [50:1427](#),
[50:1516](#), [06:837](#), [53:397](#), [53:403](#), [02:381](#)
`\LaTeXe` [19:13](#)
`\latexrelease` [1121](#)
`\LaTeXReleaseInfo` [03:36](#), [03:37](#),
[03:40](#), [03:45](#), [03:50](#), [37:47](#), [37:112](#), [38](#)
`\latexreleaseversion` [03:1](#)
`\lbrace` [21:328](#), [30:609](#)
`\lbrack` [02:441](#)
`\lccode` [14:19](#), [14:20](#),
[14:21](#), [14:22](#), [14:23](#), [14:24](#), [21:158](#),
[21:1087](#), [37:653](#), [37:670](#), [37:684](#),
[37:699](#), [37:742](#), [57:224](#), [57:241](#),
[57:249](#), [57:256](#), [57:258](#), [57:259](#),
[57:261](#), [57:263](#), [57:264](#), [57:265](#),
[57:266](#), [57:536](#), [57:544](#), [57:551](#),
[57:553](#), [57:554](#), [57:556](#), [57:558](#), [1327](#)
`\lceil` [30:613](#)
`\lcode` [1321](#)
`\ldotp` [30:512](#), [30:515](#), [30:630](#)
`\ldots` [21:342](#), [30:516](#), [1309](#)
`\le` [30:427](#), [30:429](#)
`\leaders` [30:351](#), [30:569](#), [30:570](#), [30:572](#),
[30:573](#), [41:376](#), [42:564](#), [42:577](#),
[42:589](#), [42:599](#), [44:239](#), [44:262](#), [02:546](#)
`\leadsto` [29:729](#)
`\leavevmode` [18:454](#), [18:468](#),
[21:93](#), [21:202](#), [21:309](#), [21:310](#),
[21:413](#), [21:445](#), [21:449](#), [21:452](#),
[21:500](#), [21:785](#), [21:818](#), [32:123](#),
[33:62](#), [33:888](#), [37:536](#), [37:557](#),
[37:570](#), [37:581](#), [37:611](#), [37:625](#),
[37:720](#), [37:730](#), [37:743](#), [38:469](#),
[38:481](#), [38:508](#), [39:58](#), [39:103](#),
[40:8](#), [40:17](#), [40:24](#), [40:195](#), [40:197](#),
[40:213](#), [40:245](#), [40:264](#), [40:331](#),
[40:358](#), [40:437](#), [40:544](#), [40:561](#),
[40:568](#), [41:178](#), [42:134](#), [42:314](#),
[42:373](#), [44:40](#), [44:235](#), [44:247](#),
[44:258](#), [45:530](#), [47:34](#), [54:155](#),
[54:160](#), [54:182](#), [54:187](#), [02:505](#),
[02:532](#), [02:535](#), [02:546](#), [02:548](#), [1311](#)
`\left` [30:636](#), [30:638](#),
[30:640](#), [30:642](#), [30:647](#), [30:648](#),
[30:649](#), [30:650](#), [38:167](#), [38:173](#), [38:195](#)
`\Leftarrow` [30:421](#), [30:503](#), [30:509](#)
`\leftarrow` [30:448](#),
[30:450](#), [30:491](#), [30:501](#), [30:507](#), [30:561](#)
`\leftarrowfill` [30:545](#), [30:561](#)
`\lefteqn` [38:428](#)
`\leftharpoondown` [30:464](#), [30:478](#)
`\leftharpoonup` [30:463](#)
`\lefthyphenmin` [56:11](#), [02:371](#)
`\leftline` [40:614](#)
`\leftmargin` [39:9](#),
[39:52](#), [39:53](#), [39:94](#), [39:177](#), [39:179](#), [860](#)
`\leftmargini` [38:461](#), [39:17](#), [862](#)
`\leftmarginii` [39:17](#)
`\leftmarginiii` [39:17](#)
`\leftmarginiv` [39:17](#)
`\leftmarginv` [39:17](#)
`\leftmarginvi` [39:17](#), [862](#)
`\leftmark` [49:106](#), [1009](#)
`\Leftrightarrow` [30:420](#)
`\leftrightharrow` [30:447](#)
`\leftskip` [24:756](#),
[37:452](#), [37:458](#), [37:462](#), [37:472](#),
[37:476](#), [37:480](#), [37:529](#), [37:551](#),

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
11=ltparams.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx,
16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
56=ltthyphen.dtx, 57=ltfinal.dtx

- 39:74, 40:392, 40:413, 44:232,
44:237, 44:255, 44:260, 02:416, 02:527
- legacy commands:
- \legacy_if:nTF 48:423, 48:445, 48:483, 48:490
 - \legacyoldstylenums 33:4, 33:581
 - \leq 30:425, 30:427
 - \leqno 38:439
 - \let 1290
 - \LetLtxMacro 101
 - \lfloor 30:617
 - \lg 38:4
 - \lgroup 30:619
 - \lhd 29:732
 - \hook 30:488, 30:489
 - \lim 38:6
 - \liminf 38:8
 - \limits 30:550, 30:554, 38:162, 38:353
 - \limsup 38:7
 - \line . 14:269, 42:158, 42:450, 42:808, 42:825
 - \linebreak 430
 - \linebreak 18:9, 18:27
 - \linepenalty 02:320
 - \lineskip 30:469,
38:200, 40:394, 40:414, 41:71,
41:198, 42:136, 42:315, 42:374,
53:223, 53:274, 54:831, 54:900,
54:959, 02:410, 02:435, 02:500, 02:535
 - \lineskiplimit 30:469,
30:521, 38:202, 38:206, 40:380,
40:395, 40:402, 53:224, 53:275,
54:832, 54:900, 54:959, 02:396,
02:436, 02:500, 02:537, 02:538, 1333
 - \linespread 24:379
 - \linethickness 42:130, 42:809, 42:826
 - \linewidth 20:30, 20:100,
20:158, 38:298, 38:324, 38:470,
38:482, 38:509, 38:513, 38:532, 39:9,
39:51, 39:52, 39:54, 40:390, 40:411,
41:36, 45:266, 54:144, 54:203, 861
 - \LinkTargetOff 17:20
 - \LinkTargetOn 17:20
 - list (env.) 39:34
 - \list 39:34, 39:267, 39:278
 - \listfiles 1042
 - \listfiles 20:750, 225
 - \listparindent 39:9, 39:41, 39:50, 862
 - \literal 1303
 - \ll 30:445
 - \llap 39:269, 39:280, 40:618
 - \lmoustache 30:574
 - \ln 38:5
 - \lnot 30:337, 30:338
 - \LoadClass 1040
 - \LoadClass 50:686, 50:716, 50:959, 50:1094,
50:1189, 50:1197, 50:1198, 1285
 - \LoadClassWithOptions 1040
 - \LoadClassWithOptions 50:715
 - \LoadFontDefinitionFile
..... 24:210, 24:515, 24:541,
24:542, 30:21, 30:27, 30:28, 30:29, 30:33
 - \LoadPackageWithOptions 1101
 - \loccount 04:17
 - \log 38:3
 - \loggingall 02:557
 - \loggingoutput 02:571,
02:589, 02:605, 02:619, 02:553, 1326
 - \LogHook 08:2813, 210
 - \LogSocket 10:182, 10:205, 342
 - \long 1325
 - \Longleftarrow 30:503
 - \longleftarrow 30:500
 - \Longlefttrightharrow 30:509, 30:511
 - \longlefttrightharrow 30:507
 - \longmapsto 30:505
 - \Longrightarrow 30:497
 - \longrightarrow 30:498, 30:505
 - \loop 04:150, 04:159, 24:770,
41:382, 50:1242, 50:1303, 50:1373,
50:1436, 50:1487, 50:1525, 57:365,
57:376, 57:386, 57:397, 57:427,
57:453, 57:463, 01:65, 02:490, 1307
 - \looseness 02:337
 - \lor 30:380, 30:382
 - \lower 19:2, 30:469,
40:278, 42:35, 42:45, 42:196, 42:305,
42:306, 42:353, 42:354, 42:409, 42:410
 - \lowercase
.. 14:26, 21:159, 21:1088, 21:1566,
22:111, 24:395, 24:487, 37:657,
37:674, 37:688, 37:703, 37:743, 1319
 - \lq 02:439
 - lrbox (env.) 878
 - \lrbox 40:183
 - \ltfilehookdate 52:542
 - \ltfilehookversion 52:542
 - lua commands:
 - \lua_now:n 53:31
 - \luabytecode 04:202
 - \luachunk 04:210
 - \luadef 04:182, 04:186, 42
 - \luafunction ... 04:178, 04:182, 04:186, 42
 - luamml commands:
 - \luamml_save:... 1249
 - luatexbase 04:288
 - \luatexluafunction 01:21, 01:26

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpara.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- \luatexversion 48:199, 48:413, 48:414, 48:415,
 ... 04:5, 21:1016, 01:14, 05:73, 05:119
 48:416, 48:447, 48:451, 48:452, 1011
- M
- \M 02:437
- \Macro 1282
- \mag 02:376
- \magstep 02:428
- \magstephalf 02:428
- \makeat... 1301
- \makeatletter
 ... 09:393, 20:32, 20:102, 20:160,
 24:493, 37:27, 37:92, 37:154, 44:151,
 50:680, 50:923, 50:1055, 06:833, 1298
- \makeatother
 . 09:393, 50:680, 06:833, 57:718, 1298
- \makebox 878
- \makebox 38:298, 38:324, 40:3, 1251
- \makecol 1185
- \makeglossary 995
- \makeglossary 20:205, 46:20, 1306
- \makeindex 995
- \makeindex 20:204, 46:3, 1306
- \makelabel 39:45, 39:97,
 39:236, 39:249, 39:269, 39:280, 1283
- \MakeLinkTarget .. 17:20, 43:53, 43:55, 429
- \MakeLowercase 57:571, 1313
- \MakeRobust ... 28:904, 28:1099, 06:283,
 42:805, 42:806, 42:807, 42:808,
 42:809, 42:810, 42:811, 42:812,
 42:813, 42:814, 42:815, 42:816,
 06:878, 06:879, 06:880, 06:881,
 06:882, 06:883, 06:884, 06:885,
 06:886, 06:887, 06:888, 06:889,
 06:890, 06:891, 06:892, 06:893, 1344
- \maketitle 958
- \MakeTitlecase 57:571, 1349
- \MakeUppercase 35:160,
 35:162, 35:187, 35:198, 57:571, 1318
- \mapsto 30:455
- \mapstochar 30:454, 30:455, 30:505
- \marginpar 45:308, 413
- \marginparpush 54:81, 54:2334
- \marginparsep 54:80, 54:2345, 54:2347
- \marginparwidth
 45:341, 45:359, 54:79, 54:2347
- \mark 49:46, 49:56, 49:69,
 49:77, 49:92, 49:100, 49:121, 1289
- mark commands:
- \mark_clear_structure:n
 48:210, 48:210, 1012
- \mark_copy_structure:nn
 48:124, 48:199,
- \mark_debug_off:
 48:326, 48:332, 48:343, 1008
- \mark_debug_on:
 48:326, 48:327, 48:342, 1008
- \mark_get_marks_for_reinsertion:nnn
 48:148, 48:148, 1012
- \mark_if_eq:nnnn 48:285
- \mark_if_eq:nnnnnn 48:292
- \mark_if_eq:nnnnnnTF ... 48:285, 1005
- \mark_if_eq:nnnnTF
 . 48:285, 48:395, 48:398, 48:401, 1005
- \mark_insert:nn 48:175,
 48:186, 48:232, 48:232, 48:387,
 49:24, 49:25, 49:26, 49:29, 49:30,
 49:43, 49:44, 49:45, 49:54, 49:55, 1020
- \mark_new_class:n
 48:7, 48:7, 48:16, 48:385, 1004
- \mark_set_structure_to_err:n ...
 48:221, 48:221, 48:448, 1011
- \mark_update_structure_from_
 material:nn 48:118, 48:118,
 48:406, 48:410, 48:438, 48:442, 1015
- \mark_use_first:nn
 . 48:275, 48:275, 48:389, 49:112, 1005
- \mark_use_last:nn
 . 48:275, 48:276, 48:391, 49:111, 1005
- \mark_use_top:nn
 48:275, 48:277, 48:393, 1005
- mark internal commands:
- __mark__update_dblcol_structures:
 1011
- __mark_class_status:nnn
 48:344, 48:345, 48:377
- __mark_debug:n
 48:20, 48:91, 48:165, 48:171,
 48:177, 48:244, 48:326, 48:326,
 48:339, 48:382, 48:419, 48:479, 1026
- __mark_debug_gset:
 48:326, 48:330, 48:335, 48:337
- __mark_drop_id:n
 48:175, 48:188, 48:281, 48:283, 48:284
- __mark_error:n 48:221
- __mark_error:nn
 48:224, 48:225, 48:226, 48:229, 48:280
- __mark_extract_and_handle_
 marks:nn
 ... 48:66, 48:66, 48:119, 48:151, 1016
- __mark_get_from_splitmarks: ...
 48:152, 48:157, 48:157
- __mark_init_region:nn
 48:24, 48:25, 48:26, 48:27,
 48:28, 48:29, 48:30, 48:31, 48:32,

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
 06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx,
 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
 56=lthyphen.dtx, 57=ltfinal.dtx

48:33, 48:34, 48:35, 48:36, 48:37,
 48:38, 48:39, 48:40, 48:41, 48:42,
 48:43, 48:44, 48:45, 48:46, 48:47,
 48:48, 48:49, 48:50, 48:51, 48:53, 48:53
 _mark_new_class:nn 48:7, 48:14, 48:17
 _mark_prepare_and_extract:nn ..
 48:71, 48:74, 48:74, 48:95, 1015
 _mark_region_status:nnn

... 48:347, 48:348, 48:349, 48:350,
 48:351, 48:352, 48:355, 48:358, 48:358
 _mark_status:nn

48:374, 48:374, 48:382, 48:421, 48:481
 _mark_update_dblcol_structures:
 48:435, 48:435, 48:502, 1012
 _mark_update_singlecol_-
 structures:

 48:403, 48:403, 48:500, 1012
 _mark_update_structure_from_-
 splitmarks:n

 48:120, 48:123, 48:123, 1018
 _mark_use_check:nnn

 48:275, 48:276, 48:277, 48:278
 _mark_value:mn

 48:61, 48:224, 48:225,
 48:226, 48:241, 48:263, 48:263, 48:360
 _mark_vbox_set_split_to_-
 maxdimen:NN

 48:86, 48:100, 48:101, 1017
 \markboth .. 49:21, 49:22, 49:36, 49:38,
 49:62, 49:64, 49:83, 49:85, 49:87, 1009
 \markright

 49:22, 49:49, 49:72, 49:86, 49:95, 1009
 \marks

 04:37, 57:10, 57:12
 math (env.)

 38:345
 \math

 38:345
 \mathaccent

 1332
 \mathaccent . 28:821, 28:869, 28:903, 28:913
 \mathalpha

 .. 28:991, 28:1170, 30:180, 30:181,
 30:182, 30:183, 30:184, 30:185,
 30:186, 30:187, 30:188, 30:189,
 30:190, 30:191, 30:192, 30:193,
 30:194, 30:195, 30:196, 30:197,
 30:198, 30:199, 30:200, 30:201,
 30:202, 30:203, 30:204, 30:205,
 30:206, 30:207, 30:208, 30:209,
 30:210, 30:211, 30:212, 30:213,
 30:214, 30:215, 30:216, 30:217,
 30:218, 30:219, 30:220, 30:221,
 30:222, 30:223, 30:224, 30:225,
 30:226, 30:227, 30:228, 30:229,
 30:230, 30:231, 30:232, 30:233,
 30:234, 30:235, 30:236, 30:237,
 30:238, 30:239, 30:240, 30:241,
 30:308, 30:309, 30:310, 30:311,
 30:312, 30:313, 30:314, 30:315,
 30:316, 30:317, 30:318, 30:527,
 30:528, 30:529, 30:530, 30:531,
 30:532, 30:533, 30:534, 30:536, 30:539
 \mathbf

 29:15, 29:256,
 29:309, 29:354, 29:382, 29:476, 30:162
 \mathbin 28:1175, 30:243, 30:244, 30:246,
 30:369, 30:370, 30:371, 30:372,
 30:375, 30:376, 30:377, 30:378,
 30:381, 30:382, 30:383, 30:384,
 30:385, 30:386, 30:387, 30:388,
 30:389, 30:390, 30:391, 30:392,
 30:393, 30:394, 30:395, 30:396,
 30:397, 30:398, 30:399, 30:400,
 30:401, 30:402, 30:403, 30:404,
 30:405, 30:406, 30:407, 30:408, 38:37
 \mathcal

 30:161
 \mathchar

 28:932, 28:976,
 30:346, 30:347, 30:628, 02:533, 1332
 \mathchardef 12:3, 12:4, 12:5, 12:6, 21:88,
 04:227, 28:967, 02:21, 02:22, 02:23,
 02:24, 02:107, 02:110, 02:111, 1305
 \mathcharzero

 04:227
 \mathchoice

 38:61
 \mathclose

 28:1178, 30:242,
 30:251, 30:253, 30:256, 30:261,
 30:267, 30:269, 30:271, 30:577,
 30:604, 30:608, 30:612, 30:616,
 30:622, 38:43, 38:46, 38:49, 38:52
 \mathcode .. 28:964, 30:263, 30:264, 30:265
 \MathCollectFalse

 1244
 \MathCollectFalse

 55:33
 \MathCollectTrue

 1244
 \MathCollectTrue

 55:33
 \mathdefaultsmode

 38:436, 38:437
 \mathdollar

 21:327, 30:625, 1305
 \mathellipsis

 21:341, 30:630, 1305
 \mathfontset

 1280
 \mathgroup 24:15, 26:304, 26:310,
 26:316, 26:317, 26:328, 30:654, 33:8,
 33:14, 33:578, 33:1100, 02:79, 1280
 \mathhexbox

 29:653, 02:533, 1312
 \mathindent

 38:459,
 38:471, 38:483, 38:511, 38:522, 1303
 \mathinner . 30:515, 30:519, 30:524, 30:630
 \mathit 25:3216, 29:30, 30:164, 30:167, 30:628
 \mathnormal

 30:160
 \mathop 28:1174, 30:352, 30:353, 30:354,
 30:355, 30:356, 30:357, 30:358,
 30:360, 30:361, 30:362, 30:363,
 30:364, 30:365, 30:367, 30:368,
 30:548, 30:551, 38:3, 38:4, 38:5,
 38:6, 38:7, 38:8, 38:9, 38:10, 38:11,

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lt hooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx,
 16=ltpara.dtx, 17=ltmeta.dtx, 18=lt space.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
 56=ltthyphen.dtx, 57=ltfinal.dtx

- 38:12, 38:13, 38:14, 38:15, 38:16,
38:17, 38:18, 38:19, 38:20, 38:21,
38:22, 38:23, 38:24, 38:25, 38:26,
38:27, 38:28, 38:29, 38:30, 38:31,
38:32, 38:33, 38:34, 38:162, 38:353
- `\mathopen` 28:1177, 30:252, 30:255, 30:260,
30:266, 30:268, 30:270, 30:575,
30:606, 30:610, 30:614, 30:618,
30:620, 38:41, 38:44, 38:47, 38:50
- `\mathord` 28:991, 28:1173, 30:247,
30:254, 30:257, 30:262, 30:274,
30:275, 30:276, 30:278, 30:279,
30:280, 30:281, 30:282, 30:283,
30:284, 30:285, 30:286, 30:287,
30:288, 30:289, 30:290, 30:291,
30:292, 30:293, 30:294, 30:295,
30:296, 30:297, 30:298, 30:299,
30:300, 30:301, 30:302, 30:303,
30:304, 30:305, 30:306, 30:307,
30:319, 30:320, 30:321, 30:322,
30:323, 30:324, 30:325, 30:326,
30:327, 30:328, 30:329, 30:330,
30:331, 30:332, 30:333, 30:334,
30:335, 30:336, 30:338, 30:339,
30:340, 30:341, 30:342, 30:343,
30:344, 30:345, 30:535, 30:537,
30:538, 30:560, 30:561, 30:564,
30:565, 30:566, 30:567, 30:579,
30:581, 30:583, 30:586, 30:588,
30:602, 30:624, 30:625, 30:626, 30:627
- `\mathpalette` 30:468, 30:472,
30:475, 38:60, 38:69, 38:99, 38:129
- `\mathparagraph`
. 21:330, 22:221, 22:233, 30:625, 1322
- `\mathpunct` 28:1179,
30:245, 30:249, 30:512, 30:513, 30:514
- `\mathrel` 28:1176, 30:248, 30:250,
30:258, 30:259, 30:272, 30:273,
30:349, 30:409, 30:410, 30:411,
30:412, 30:413, 30:414, 30:415,
30:416, 30:417, 30:418, 30:419,
30:420, 30:421, 30:422, 30:425,
30:426, 30:429, 30:430, 30:431,
30:432, 30:433, 30:434, 30:435,
30:436, 30:437, 30:438, 30:439,
30:440, 30:441, 30:443, 30:444,
30:445, 30:446, 30:447, 30:448,
30:449, 30:452, 30:453, 30:454,
30:456, 30:457, 30:458, 30:459,
30:460, 30:461, 30:462, 30:463,
30:464, 30:465, 30:466, 30:468,
30:472, 30:475, 30:482, 30:484,
30:487, 30:488, 30:490, 30:493,
30:495, 30:590, 30:592, 30:594,
30:596, 30:598, 30:600, 38:42,
38:45, 38:48, 38:51, 38:162, 38:353
- `\mathring` 30:539, 1324
- `\mathrm` 29:6,
29:407, 29:450, 29:482, 30:159, 1281
- `\mathsection`
. 21:331, 22:220, 22:232, 30:625, 1322
- `\mathsf` 29:9,
29:412, 29:455, 29:485, 30:163, 30:166
- `\mathsterling` 21:339, 30:625, 1305
- `\mathstrut` . . . 38:84, 38:93, 38:171, 38:172
- `\mathsurround` 02:400, 02:521, 1322
- `\mathsymbol` 28:969
- `\mathtt` 29:12, 29:417, 29:460, 29:488, 30:165
- `\mathunderscore` 30:625, 1309
- `\mathversion` . 24:399, 29:615, 29:617, 1281
- `\mathversion.` 1280
- `\matrix` 38:169, 38:173, 38:180
- `\max` 38:22
- `\maxdeadcycles` 54:3, 02:373
- `\maxdepth` 18:302, 18:335,
20:60, 20:129, 20:184, 54:88, 54:167,
54:168, 54:497, 54:720, 54:998,
54:1038, 54:1265, 57:152, 02:393, 1293
- `\maxdimen` 02:569, 02:570,
02:588, 02:604, 02:619, 24:742,
24:752, 24:788, 24:803, 26:385,
26:438, 30:469, 42:474, 42:502,
42:532, 42:610, 42:627, 50:1597,
50:1645, 50:1654, 53:351, 53:353,
53:409, 54:289, 54:2353, 54:2373,
54:2378, 54:2764, 54:2832, 54:2833,
54:2835, 57:156, 02:309, 02:394,
02:395, 02:500, 02:538, 02:554, 1282
- `\mbox` 878
- `\mbox` 19:13, 21:313, 21:429,
21:589, 21:1187, 29:649, 30:517,
40:11, 40:20, 40:24, 42:52, 45:409,
45:416, 45:437, 45:444, 02:533, 1310
- `\mddefault` 29:19, 29:285,
29:291, 29:292, 29:293, 29:332,
29:333, 29:334, 29:343, 29:370,
29:386, 29:403, 29:444, 29:480,
30:104, 30:116, 30:118, 30:132, 700
- `\mdseries` 29:17, 29:18,
29:222, 29:278, 29:329, 29:330,
29:364, 29:365, 29:384, 29:385,
29:478, 29:479, 29:652, 32:20, 226
- `mdseries` 29:421
- `mdseries/defaults` 29:421
- `\meaning` 01:203,
01:212, 28:643, 28:656, 28:757,
28:822, 28:869, 28:933, 28:1027,

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpara.dtx,
16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
56=ltthyphen.dtx, 57=ltfinal.dtx

- 28:1123, 28:1227, 06:240, 06:302, 30:561, 30:562, 30:563, 38:36, 38:37,
06:340, 06:368, 01:307, 06:451, 06:804 38:40, 38:73, 38:74, 44:240, 44:263
- `\medbreak` 06:885, 06:906, [02:515](#)
- `\mediumseries` [703](#)
- `\medmuskip` 30:656,
38:36, 38:38, 38:224, 38:227, 38:241
- `\medskip` [18:447](#), [02:518](#)
- `\medskipamount` [18:448](#), [18:450](#), [02:517](#)
- `\medspace` [38:214](#)
- `\MessageBreak` [03:101](#), [14:3](#), [14:6](#), [14:13](#),
[14:33](#), [14:46](#), [14:60](#), [14:73](#), [14:220](#),
[14:222](#), [14:228](#), [14:235](#), [17:13](#),
[21:179](#), [21:1010](#), [21:1571](#), [21:1573](#),
[21:1575](#), [21:1578](#), [24:35](#), [24:36](#),
[24:633](#), [24:667](#), [25:2844](#), [26:21](#),
[26:22](#), [26:68](#), [26:89](#), [26:328](#), [26:479](#),
[26:499](#), [26:531](#), [26:547](#), [26:562](#),
[26:575](#), [27:31](#), [27:33](#), [28:281](#), [28:329](#),
[28:598](#), [28:607](#), [28:745](#), [29:62](#), [29:95](#),
[05:78](#), [05:81](#), [05:82](#), [05:83](#), [05:84](#),
[05:85](#), [05:86](#), [05:99](#), [05:100](#), [05:101](#),
[05:102](#), [05:103](#), [32:144](#), [33:23](#), [33:43](#),
[33:45](#), [33:64](#), [33:812](#), [33:814](#), [33:815](#),
[33:816](#), [33:818](#), [33:820](#), [33:821](#),
[33:822](#), [33:823](#), [33:824](#), [33:874](#),
[33:876](#), [33:883](#), [33:890](#), [33:1105](#),
[37:54](#), [37:119](#), [37:158](#), [06:204](#),
[06:291](#), [06:330](#), [06:358](#), [50:353](#),
[50:367](#), [50:742](#), [50:753](#), [50:755](#),
[50:757](#), [50:768](#), [50:954](#), [50:955](#),
[50:956](#), [50:957](#), [50:967](#), [50:968](#),
[50:970](#), [50:971](#), [50:972](#), [50:974](#),
[50:976](#), [50:1062](#), [50:1063](#), [50:1065](#),
[50:1066](#), [50:1067](#), [50:1069](#), [50:1071](#),
[50:1089](#), [50:1090](#), [50:1091](#), [50:1092](#),
[50:1174](#), [50:1191](#), [50:1192](#), [50:1264](#),
[50:1281](#), [50:1320](#), [50:1395](#), [50:1414](#),
[50:1453](#), [50:1508](#), [50:1542](#), [50:1658](#),
[50:1660](#), [50:1743](#), [50:1746](#), [50:1759](#),
[50:1761](#), [52:488](#), [53:99](#), [53:175](#),
[53:371](#), [53:477](#), [53:478](#), [53:479](#),
[53:480](#), [54:777](#), [54:2494](#), [54:2541](#),
[57:300](#), [57:301](#), [57:302](#), [57:304](#), [1302](#)
- `\mho` [29:725](#)
- `\mid` [30:413](#)
- `\min` [38:23](#)
- `minipage (env.)` [879](#)
- `\minipage` [40:424](#)
- `\mit` [29:777](#)
- `\mkern` [30:346](#),
[30:349](#), [30:351](#), [30:473](#), [30:482](#),
[30:524](#), [30:525](#), [30:526](#), [30:556](#),
[30:557](#), [30:558](#), [30:559](#), [30:560](#),
- mlist commands:
`mlist_to_hlist` [04:1008](#)
- mode commands:
`\mode_if_horizontal:TF` . [16:95](#), [16:100](#)
`\mode_if_inner:TF` [16:96](#)
`\mode_if_math:TF` [28:360](#)
`\mode_if_vertical:TF` . [16:114](#), [16:126](#)
- `\models` [30:495](#), [1326](#)
- module commands:
`module_error` [04:344](#)
`module_info` [04:344](#)
`module_warning` [04:344](#)
- `\module_error` [45](#)
`\module_info` [45](#)
`\module_warning` [45](#)
- modules [04:297](#)
- `\month` [03:11](#), [03:17](#), [01:169](#),
[50:1298](#), [50:1431](#), [50:1520](#), [02:385](#)
- `\moveright` [54:835](#), [54:903](#), [54:962](#)
- `\mp` [30:400](#)
- `\mcount` [41:379](#)
- msg commands:
`\msg...` [1344](#)
`\msg_error:nn` [10:49](#), [11:267](#),
[16:122](#), [16:132](#), [36:14](#), [08:436](#), [08:453](#)
`\msg_error:nnn` [08:2164](#), [10:30](#), [10:96](#),
[10:115](#), [10:136](#), [11:62](#), [11:69](#), [11:185](#),
[11:233](#), [11:247](#), [11:291](#), [11:313](#),
[11:420](#), [11:433](#), [11:450](#), [11:506](#),
[11:544](#), [11:623](#), [11:632](#), [11:738](#),
[11:890](#), [48:11](#), [48:259](#), [51:247](#), [52:75](#),
[07:348](#), [07:754](#), [07:3129](#), [07:3135](#),
[07:3146](#), [07:3165](#), [07:3168](#), [07:3176](#),
[08:76](#), [08:94](#), [08:179](#), [08:208](#), [08:211](#),
[08:461](#), [08:525](#), [08:543](#), [08:597](#), [08:617](#)
`\msg_error:nnnn`
..... [09:60](#), [09:118](#), [09:414](#),
[10:83](#), [10:112](#), [10:133](#), [11:56](#), [11:75](#),
[11:199](#), [11:562](#), [11:590](#), [11:602](#),
[11:691](#), [11:703](#), [11:803](#), [11:907](#),
[11:913](#), [16:20](#), [16:33](#), [16:53](#), [16:65](#),
[16:108](#), [36:35](#), [48:230](#), [51:38](#), [51:77](#),
[51:221](#), [07:462](#), [07:470](#), [07:494](#),
[07:498](#), [07:534](#), [07:565](#), [07:580](#),
[07:666](#), [07:683](#), [07:711](#), [07:723](#),
[07:742](#), [07:779](#), [07:1149](#), [07:2047](#),
[07:2684](#), [07:2688](#), [07:3077](#), [07:3091](#),
[07:3196](#), [07:3210](#), [08:112](#), [08:507](#), [08:641](#)
`\msg_error:nnnnn` [11:48](#),
[11:417](#), [07:673](#), [07:698](#), [07:2207](#),
[07:2214](#), [07:2523](#), [08:440](#), [08:471](#), [08:481](#)

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpara.dtx,
16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
26=ltfssstrc.dtx, 27=ltfsscscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
56=lthyphen.dtx, 57=ltfinal.dtx

- `\msg_error:nnnnnn`
.. 08:1322, 08:1334, 08:1359, 08:1761
 - `\msg_expandable_error:nn`
..... 08:341, 08:342
 - `\msg_expandable_error:nnn`
..... 08:2559, 05:185, 36:77,
36:113, 07:419, 07:3242, 08:370, 08:1262
 - `\msg_expandable_error:nnnn`
..... 52:464, 07:2415, 07:2442
 - `\msg_info:nnnn`
..... 11:109, 11:193, 11:403,
11:648, 11:671, 11:687, 07:72,
07:83, 07:233, 07:234, 07:250, 07:254
 - `\msg_line_context:`
08:2641, 08:2646, 08:2650, 08:2697,
11:1013, 11:1019, 11:1026, 11:1033,
11:1040, 11:1047, 11:1055, 11:1142,
11:1144, 11:1148, 11:1151, 48:20,
48:93, 48:166, 48:245, 07:3034,
07:3039, 07:3044, 07:3049, 08:564
 - `\msg_module_name:n`
..... 51:96, 51:226, 51:232, 51:236
 - `\g_msg_module_name_prop` 08:2583
 - `\g_msg_module_type_prop` . 08:2581,
08:2582, 10:181, 11:1162, 48:299, 07:57
 - `\msg_new:nnn`
08:2639, 08:2644, 08:2648, 08:2693,
08:2699, 08:2704, 08:2709, 08:2714,
08:2718, 08:2725, 08:2890, 11:1141,
11:1143, 11:1145, 11:1150, 11:1152,
11:1157, 05:187, 51:224, 52:470,
07:1150, 07:2988, 07:3031, 07:3036,
07:3041, 07:3046, 07:3051, 07:3063
 - `\msg_new:nnnn` 08:2584,
08:2594, 08:2601, 08:2608, 08:2617,
08:2631, 08:2653, 08:2669, 08:2682,
09:592, 09:602, 10:158, 10:163,
10:167, 10:171, 10:177, 11:969,
11:977, 11:983, 11:992, 11:998,
11:1005, 11:1012, 11:1018, 11:1025,
11:1032, 11:1039, 11:1046, 11:1054,
11:1061, 11:1070, 11:1076, 11:1082,
11:1089, 11:1095, 11:1101, 11:1107,
11:1113, 11:1129, 11:1135, 16:144,
16:155, 36:245, 36:251, 36:257,
48:300, 48:308, 48:315, 51:92,
51:229, 51:252, 07:2812, 07:2819,
07:2826, 07:2833, 07:2840, 07:2846,
07:2852, 07:2859, 07:2866, 07:2873,
07:2880, 07:2887, 07:2895, 07:2902,
07:2909, 07:2916, 07:2924, 07:2932,
07:2941, 07:2948, 07:2955, 07:2962,
07:2969, 07:2975, 07:2981, 07:2990,
07:2996, 07:3003, 07:3010, 07:3021
 - `\msg_redirect_module:nnn` 07:56
 - `\msg_show:nnnnn` 11:960
 - `\msg_show:nnnnnn` 11:949
 - `\msg_show_item_unbraced:nn`
..... 11:953, 11:965
 - `\msg_warning:nnn` . 07:80, 08:295, 08:313
 - `\msg_warning:nnnn` 08:2889,
51:220, 07:735, 07:1615, 07:1733,
08:979, 08:992, 08:1014, 08:1028
 - `\msg_warning:nnnnn` 08:769
 - `\msg_warning:nnnnnn` 08:1769
 - msg internal commands:
 - `\c__msg_coding_error_text_tl` ...
..... 48:303, 48:311, 48:318
 - `\c__msg_return_text_tl` 48:306, 48:323
 - `\mskip` 38:36, 38:38,
38:219, 38:238, 38:241, 38:243, 38:244
 - `\mu` 30:290
 - `\mubyte` 57:352
 - `\multicolumn` 41:233, 1252
 - `\multipt` 42:78, 42:810, 42:827
 - `\multispan` 41:233, 41:379, 1315
 - `\muskip` 04:34,
30:556, 30:557, 02:29, 02:55, 02:93
 - muskip commands:
 - `\muskip_eval:n` 11:366
 - `\muskip_new:N` 11:35
 - `\l_tmpa_muskip` 354
 - `\muskipdef` 04:228, 02:55, 02:93
 - `\muskipzero` 04:228
- N
- `\n` 04:331, 04:333, 04:340, 04:342, 04:469,
04:688, 04:713, 04:737, 04:778,
04:800, 04:819, 04:827, 04:828,
04:853, 04:870, 04:909, 04:916,
04:917, 04:924, 04:936, 57:121, 57:126
 - `\nabla` 30:330
 - `\NAME` 81
 - `\narrower` 02:526
 - `\natural` 30:340
 - `\ncallback` 04:783
 - `\ndefault` 04:788, 04:792
 - `\ne` 30:424, 736
 - `\narrow` 30:416
 - `\NeedsFormat` 1285
 - `\NeedsTeXFormat`
.. 26:12, 33:781, 50:747, 50:1805, 1289
 - `\neg` 30:336, 30:337
 - `\negmedspace` 38:214, 1341
 - `\negthickspace` 38:214, 1341
 - `\negthinspace` 18:544, 38:214, 1338
 - `\neq` 30:423, 736

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

new commands:	39:15, 39:16, 39:17, 39:18, 39:19,
new_attribute	04:410, 39:20, 39:21, 39:22, 40:210, 40:211,
new_bytecode	04:444, 41:3, 41:5, 41:6, 41:7, 41:8, 41:166,
new_chunkname	04:457, 41:338, 41:339, 41:340, 41:341, 42:3,
new_luafunction	04:473, 42:4, 42:5, 42:7, 42:431, 42:432,
new_whatsit	04:432, 42:433, 42:434, 42:435, 42:436,
\new_attribute	43, 42:667, 42:668, 42:670, 42:671,
\new_bytecode	43, 42:672, 42:673, 45:451, 54:67,
\new_chunkname	43, 54:68, 54:69, 54:71, 54:72, 54:73,
\new_luafunction	43, 54:74, 54:75, 54:76, 54:77, 54:78,
\new_whatsit	43, 54:79, 54:80, 54:81, 54:87, 54:89,
\newattribute	42, 54:90, 54:91, 54:92, 54:104, 54:106,
\newattribute	04:74, 04:238, 54:108, 54:110, 54:111, 54:112,
\newbox	12:13, 37:583, 38:66, 39:27, 54:113, 54:114, 54:115, 54:116,
40:154, 41:16, 41:17, 41:18, 41:343,	54:500, 02:47, 54:2521, 54:2522,
42:6, 42:669, 42:674, 54:82, 54:118,	54:2527, 02:309, 02:311, 02:312, 02:426
54:119, 54:120, 02:47, 02:314, 02:523	
\newcatcodetable	42
\newcatcodetable	04:84,
04:93, 04:94, 04:120, 04:121, 04:242	
\newcommand	81
\newcommand	
.	21:4, 25:2918, 25:2923, 25:2928,
29:37, 29:73, 30:63, 30:64, 30:65,	08:2730, 08:2732, 08:2734,
30:66, 30:68, 30:69, 30:71, 30:72,	08:2738, 08:2740, 08:2742, 08:2753,
30:104, 30:105, 30:106, 30:107,	08:2755, 08:2765, 08:2767, 08:2776,
30:108, 30:109, 30:131, 30:132,	08:2778, 08:2785, 08:2787, 08:2789,
30:133, 06:77, 36:240, 37:396,	08:2791, 08:2793, 08:2817, 08:2819,
37:397, 37:398, 37:399, 42:681,	08:2822, 17:24, 17:46, 17:55, 17:56,
52:573, 52:574, 52:575, 52:576,	17:57, 20:750, 22:132, 22:140,
52:578, 54:2915, 54:2918, 54:2921,	35:66, 35:68, 35:163, 36:80, 48:380,
54:2922, 54:2925, 54:2926, 57:561, 310	51:187, 51:189, 51:200, 51:258,
\NewCommandCopy	06:473, 06:475, 07:1126, 07:1178,
.	07:1196, 07:1209, 07:1224, 07:1342, 98
\newcount	12:7, 12:8, 18:125,
20:7, 22:43, 22:63, 24:422, 26:26,	
28:28, 28:143, 28:445, 38:55, 38:357,	
38:358, 39:23, 39:24, 39:25, 39:26,	
39:56, 39:257, 39:272, 40:473, 41:11,	
41:12, 41:13, 41:14, 41:15, 41:335,	
41:336, 41:337, 42:663, 42:664,	
42:665, 42:666, 42:675, 44:36,	
44:140, 44:141, 45:3, 45:267, 45:268,	
45:269, 45:270, 50:1594, 54:101,	
54:103, 54:105, 54:107, 54:109,	
54:117, 02:47, 54:2520, 54:2913,	
54:2916, 54:2919, 54:2923, 57:3,	
57:4, 57:5, 57:91, 02:356, 02:427, 1328	
\newcounter	521
\newcounter	22:10
\newdimen	12:10, 12:11, 12:12,
18:124, 26:399, 26:400, 38:53, 39:9,	
39:10, 39:11, 39:12, 39:13, 39:14,	
39:15, 39:16, 39:17, 39:18, 39:19,	
39:20, 39:21, 39:22, 40:210, 40:211,	
41:3, 41:5, 41:6, 41:7, 41:8, 41:166,	
41:338, 41:339, 41:340, 41:341, 42:3,	
42:4, 42:5, 42:7, 42:431, 42:432,	
42:433, 42:434, 42:435, 42:436,	
42:667, 42:668, 42:670, 42:671,	
42:672, 42:673, 45:451, 54:67,	
54:68, 54:69, 54:71, 54:72, 54:73,	
54:74, 54:75, 54:76, 54:77, 54:78,	
54:79, 54:80, 54:81, 54:87, 54:89,	
54:90, 54:91, 54:92, 54:104, 54:106,	
54:108, 54:110, 54:111, 54:112,	
54:113, 54:114, 54:115, 54:116,	
54:500, 02:47, 54:2521, 54:2522,	
54:2527, 02:309, 02:311, 02:312, 02:426	
\NewDocumentCommand	
.	08:2730, 08:2732, 08:2734,
08:2738, 08:2740, 08:2742, 08:2753,	
08:2755, 08:2765, 08:2767, 08:2776,	
08:2778, 08:2785, 08:2787, 08:2789,	
08:2791, 08:2793, 08:2817, 08:2819,	
08:2822, 17:24, 17:46, 17:55, 17:56,	
17:57, 20:750, 22:132, 22:140,	
35:66, 35:68, 35:163, 36:80, 48:380,	
51:187, 51:189, 51:200, 51:258,	
57:633, 57:640, 57:647, 07:3071, 1338	
\NewDocumentEnvironment	
.	07:2855, 07:2862, 07:3107, 222
\newenvironment	82
\newenvironment	
06:146, 50:1296, 50:1429, 50:1518, 222	
\NewEnvironmentCopy	06:514, 06:516, 1351
\NewExpandableDocumentCommand	
.	36:118, 48:388, 48:390, 48:392,
48:394, 48:397, 48:400, 57:601, 07:3190	
\newfam	04:38, 24:17, 02:47, 1281
\newfont	29:619
\newgroup	28:48
\newhelp	02:306
\NewHook	08:2730, 08:2905,
20:73, 20:74, 20:75, 20:348, 20:351,	
20:400, 26:151, 29:421, 29:422,	
29:423, 29:424, 29:425, 29:426,	
29:427, 29:428, 29:429, 37:34,	
37:35, 37:36, 37:37, 37:38, 37:99,	
37:100, 37:101, 37:102, 37:103,	
50:1040, 50:1041, 52:177, 54:683, 221	
\NewHookPair	311
\NewHookWithArguments	08:2736, 35:101, 312
\newif	03:73, 12:9, 20:5, 20:6,
24:267, 25:2789, 25:2801, 28:16,	
29:529, 32:82, 33:832, 33:1185,	
33:1186, 35:3, 06:168, 38:75, 38:76,	

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpara.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- 38:203, 38:359, 39:28, 39:29, 39:30,
39:31, 39:32, 39:33, 39:140, 40:423,
40:529, 41:19, 41:251, 42:157,
42:427, 42:428, 42:429, 42:430,
42:459, 42:460, 44:38, 44:124, 50:2,
06:872, 54:93, 54:94, 54:95, 54:96,
54:97, 54:98, 54:99, 54:100, 1280
- `\newinsert` 40:474, 45:390,
54:23, 54:2352, 02:193, 02:242, 1331
- `\newlabel` 35:84, 35:106, 35:174, 1314
- `\newlanguage` 02:47, 57:284
- `\newlength` 531
- `\newlength` 23:3
- `\newline` 18:93, 18:100, 18:106
- `\newlinechar` 06:20, 01:56, 02:381
- `\newluabytecode` 43
- `\newluabytecode` 04:197, 04:252
- `\newluachunkname` 43
- `\newluachunkname` 04:205, 04:254
- `\newluacmd` 42
- `\newluacmd` 04:181, 43
- `\newluafunction` 42
- `\newluafunction`
. 04:4, 04:173, 04:236, 04:248, 42
- `\NewMarkClass` 48:385,
48:506, 57:25, 57:26, 57:27, 1004
- `\newmarks` 48:22, 57:6, 1010
- `\newmathalphabet` 27:13, 27:109
- `\NewMirroredHookPair`
. 08:2730, 08:2907, 54:682, 54:684, 200
- `\NewMirroredHookPairWithArguments` . .
. 08:2736, 200
- `\NewModuleRelease`
09:4, 10:4, 11:7, 03:152, 16:4, 17:3,
33:2, 36:4, 48:4, 07:9, 55:303, 08:4, 1343
- `\newmuskip` 02:47
- `\newpage` 54:131, 54:137, 54:148, 335
- `\NewProperty` 36:38, 36:267, 806
- `\newprotectedluacmd` 43
- `\newprotectedluacmd` 04:181, 21:1032, 53:28
- `\newread` 02:47, 02:307, 1283
- `\NewReversedHook` 08:2730,
08:2906, 20:349, 20:350, 20:401,
20:402, 50:1042, 50:1043, 52:178, 200
- `\NewReversedHookWithArguments`
. 08:2736, 312
- `\newrobustcmd` 96
- `\newsavebox` 878
- `\newsavebox` 40:154
- `\newskip` 12:14,
12:15, 12:17, 18:450, 18:451, 18:452,
18:517, 18:530, 23:3, 37:493, 38:360,
38:460, 39:2, 39:3, 39:4, 39:5, 39:6,
39:7, 39:8, 02:47, 54:2927, 54:2928,
- 54:2929, 54:2933, 54:2934, 54:2937,
54:2938, 54:2939, 54:2943, 54:2944,
54:2945, 02:310, 02:313, 02:424, 02:425
- `\NewSocket` 10:182, 10:203, 35:118,
35:119, 54:631, 54:681, 55:36, 55:46,
55:60, 55:61, 55:64, 55:65, 55:68,
55:69, 55:70, 55:71, 55:72, 55:73,
55:74, 55:75, 55:76, 55:77, 55:78,
55:79, 55:80, 55:81, 55:82, 55:84,
55:85, 55:86, 55:87, 55:88, 55:89,
55:90, 55:91, 55:92, 55:93, 55:94,
55:95, 55:96, 55:97, 55:98, 55:99,
55:101, 55:102, 55:103, 55:104,
55:109, 55:110, 55:111, 55:112,
55:113, 55:114, 55:115, 55:116,
55:117, 55:118, 55:119, 55:120,
55:121, 55:126, 55:128, 55:130,
55:133, 55:134, 55:135, 55:136,
55:138, 55:140, 55:142, 55:144,
55:145, 55:146, 55:147, 55:148,
55:149, 55:152, 55:155, 55:158, 339
- `\NewSocketPlug`
. 10:173, 10:182, 10:207, 35:120,
54:632, 54:642, 54:649, 54:656,
54:665, 54:670, 54:675, 55:37, 55:48,
55:105, 55:106, 55:122, 55:124,
55:131, 55:150, 55:153, 55:156, 337
- `\NewTemplateType`
. 11:1073, 11:1132, 11:1163, 351
- `\newtheorem` 43:1, 1315
- `\newtie` 21:873, 33:152, 33:153, 33:172,
33:669, 33:976, 33:977, 33:1284, 760
- `\newtoks`
03:36, 12:16, 16:77, 24:420, 24:421,
25:2887, 26:248, 02:47, 02:306, 1331
- `\newwhatsit` 43
- `\newwhatsit` 04:189, 04:250
- `\newwrite` 20:3, 20:4,
44:154, 46:4, 46:21, 02:47, 02:308, 1305
- `\newXeTeXintercharclass` 57:35
- `\next` 1308
- `\NextLinkTarget` 17:20
- nfss internal commands:
- `_nfss_init_mv_freeze:N`
. 28:334, 28:359, 28:360
- `\NG` 21:552, 21:1167, 57:656, 1300
- `\ng` 21:573, 21:1168, 57:656, 1300
- `\ni` 30:441, 30:442
- `\noalign` 30:350, 30:542, 30:545, 30:548,
30:549, 30:553, 30:554, 38:171,
38:172, 38:188, 38:191, 38:205,
38:404, 38:414, 41:224, 41:230,
41:359, 41:378, 42:148, 42:154, 1280

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- `\nobreak` 18:60, 18:72, 18:116,
 18:142, 18:148, 18:161, 18:174,
 18:200, 18:398, 18:406, 18:432,
 18:440, 18:461, 18:468, 18:528,
 20:203, 20:215, 21:429, 21:455,
 21:457, 21:589, 21:1187, 37:418,
 37:425, 40:599, 40:609, 44:90,
 44:237, 44:238, 44:242, 44:260,
 44:261, 44:265, 45:531, 48:256,
 49:48, 49:58, 49:71, 49:79, 49:94,
 49:102, 06:886, 06:907, 54:334,
 54:1424, 54:1573, 54:1734, 57:208,
 57:210, 57:214, 57:215, 57:216,
 57:220, 02:503, 02:506, 02:508, 1314
`\nobreakdashes` 18:453
`\nobreakspace` 18:467, 447
`\nobreakspaceU` 1320
`\NoCaseChange` 57:571, 1349
`\nocite` 998
`\nocite` 47:59, 1001
`\nocorr` . . . 32:43, 32:58, 32:62, 32:65, 1295
`\nocorrlist` 32:89, 32:121
`\noexpand` 1190
`\nofiles` 451
`\nofiles` 20:199, 825
`\noindent` 24:766, 24:792, 44:139, 418
`\nointerlineskip` 30:350,
 30:542, 30:545, 30:549, 30:553,
 38:297, 38:323, 42:562, 42:565,
 42:575, 42:577, 54:2342, 54:2350, 02:498
`\nolimits` 30:359, 30:366,
 38:3, 38:4, 38:5, 38:9, 38:10, 38:11,
 38:12, 38:13, 38:14, 38:15, 38:16,
 38:17, 38:18, 38:19, 38:20, 38:21,
 38:26, 38:27, 38:28, 38:29, 38:31, 38:34
`\nolinebreak` 430
`\nolinebreak` 18:9, 18:28
`\nonfrenchspacing` 02:709, 20:48,
 20:118, 20:176, 06:887, 06:908, 02:431
`\nonscript` 38:36, 38:38
`\nonstopmode` 1017
`\nonumber` 38:387, 38:426, 38:427
`\nopagebreak` 430
`\nopagebreak` 18:7, 18:26
`\noprotrusion` 44:247, 44:270
`\normalbaselines` 38:167,
 38:169, 02:396, 02:409, 02:410, 02:435
`\normalbaselineskip`
 26:189, 40:396, 40:415, 02:424, 02:436
`\normalcolor`
 . . . 38:352, 38:456, 40:110, 40:465,
 44:243, 44:266, 45:97, 45:166,
 54:214, 54:591, 54:699, 54:842,
 54:857, 54:907, 54:917, 54:966,
 54:976, 54:2810, 54:2853, 54:2883, 1310
`\normalfont` 24:743, 24:804, 29:654,
 29:674, 29:676, 29:685, 29:690,
 29:692, 29:700, 29:707, 29:709,
 29:715, 32:18, 37:571, 38:352,
 38:456, 44:243, 44:266, 45:401, 1315
`normalfont` 29:421
`\normalize` 1293
`\normallineskip`
 40:394, 40:414, 02:424, 02:435
`\normallineskiplimit` 38:206,
 40:379, 40:395, 40:401, 02:424, 02:436
`\normalmarginpar` 45:387
`\normalsfcodes`
 20:44, 20:46, 20:48, 20:114, 20:116,
 20:118, 20:172, 20:174, 20:176,
 20:198, 54:826, 54:896, 54:955, 1322
`\normalshape` 25:3202, 25:3243, 623
`\normalsize` 20:42, 20:112, 20:170,
 32:142, 45:23, 45:176, 45:372,
 50:5, 54:825, 54:895, 54:954, 1295
`\not` 30:349, 30:423, 30:424, 30:446
`\notexpanded` 325
`\notin` 30:472
`\noexpand` 1285
`\nu` 30:291
`\null` 21:347,
 21:383, 21:507, 21:510, 21:856,
 21:859, 21:1275, 35:21, 35:38, 35:55,
 37:536, 37:557, 37:720, 37:730,
 38:112, 38:121, 38:169, 38:198,
 44:237, 44:260, 53:385, 02:449, 1310
`\nulldelimiterspace` 30:653, 02:398
`\nullfont` 37:242
`\number` 03:52, 03:61,
 04:105, 22:195, 24:693, 24:696,
 26:440, 28:65, 28:94, 28:114, 28:129,
 28:165, 28:196, 28:226, 28:258,
 29:646, 06:2, 06:114, 50:1205,
 50:1298, 50:1431, 50:1520, 53:351, 01:70
`\numberline` 44:72, 44:82, 44:273, 45:17, 1284
`\numexpr` 04:82, 04:105,
 04:157, 21:1056, 28:151, 06:747,
 54:32, 02:189, 02:205, 02:215, 02:246
`\unknown` 04:805
`\narrow` 30:418
`\nxt` 1308

O

- `\O` 21:264,
 21:420, 21:554, 21:792, 21:1152, 57:655
`\o` 21:273,
 21:425, 21:575, 21:803, 21:1158, 57:655

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
 06=ltdefs.dtx, 07=ltcmd, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx,
 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
 56=ltthyphen.dtx, 57=ltfinal.dtx

- 45:496, 49:135, 49:136, 02:11,
53:401, 54:164, 54:191, 54:255,
54:2386, 02:445, 02:461, 02:463,
02:480, 02:502, 02:511, 02:512,
02:513, 02:515, 02:517, 02:519, 824
- para commands:
- \para_end: 16:93,
16:93, 16:140, 16:141, 16:142, 422
 - \g_para_indent_box .. 16:16, 16:49,
16:84, 16:86, 16:89, 16:91, 16:117, 418
 - \para_omit_indent: .. 16:88, 16:92, 415
 - \para_raw_end:
..... 16:113, 16:135, 16:138, 415
 - \para_raw_indent:
..... 16:113, 16:113, 16:136, 415
 - \para_raw_noindent:
..... 16:113, 16:125, 16:137, 424
- para internal commands:
- __para_handle_indent:
... 16:34, 16:66, 16:85, 16:85, 16:119
 - \g__para_standard_everypar_tl ...
16:12, 16:76, 16:78, 16:118, 16:129, 420
 - __para_tmp:w 16:37, 16:41, 16:68, 16:72
- para/after 16:6, 413
- para/before 16:6, 413
- para/begin 16:6, 413
- para/end 16:6, 413
- \paracntvalue 416
- \paragraphmark 44:143
- \parallel 30:412
- \parbox 878
- \parbox 40:298, 413
- \parfillskip ... 24:742, 24:758, 24:803,
37:454, 37:464, 37:473, 37:481,
37:530, 37:552, 39:76, 40:393,
40:414, 44:232, 44:255, 02:423, 424
- \parindent 37:454, 37:459, 37:464, 37:473,
37:477, 37:481, 37:530, 37:552,
39:50, 40:388, 40:409, 44:233,
44:256, 02:404, 02:527, 02:528, 410
- \parsep 39:1, 39:49, 39:90, 862
- \parseunicodedataI 04:123, 04:162
- \parseunicodedataII 04:124, 04:126
- \parseunicodedataIII 04:128, 04:134
- \parseunicodedataIV 04:130, 04:142
- \parseunicodedataV 04:146, 04:149
- \parshape 39:54, 418
- \parskip 37:429, 37:528, 37:530,
37:550, 37:552, 38:528, 39:49, 39:73,
39:88, 39:90, 39:117, 39:184, 39:203,
39:254, 40:388, 40:409, 41:79,
54:1434, 54:1583, 54:1746, 02:411, 410
- \partial 30:326
- \partopsep 38:526, 39:1, 39:61, 862
- \PassOptionsToClass 1040
- \PassOptionsToClass 50:441
- \PassOptionsToPackage 1040
- \PassOptionsToPackage 50:441
- \patterns 21:225, 1309
- \pausing 02:338
- \pdffilesize 05:71, 05:117
- \pdfgentounicode 57:315, 57:316,
57:320, 57:335, 57:340, 57:341, 57:342
- \pdfhorigin 53:314
- \pdfsavepos 810
- \pdftexrevision 57:321
- \pdftexversion 57:319, 57:320, 57:321
- \pdfvariable 53:313, 53:318
- \pdfvorigin 53:319
- peek commands:
- \peek_meaning:NTF 07:2773, 187
 - \peek_meaning_remove:NTF
..... 07:2088, 07:2205, 07:2761, 171
 - \peek_N_type:TF
..... 07:2094, 07:2118, 07:2150
 - \peek_remove_spaces:n 07:2086
- \penalty 18:35, 18:38, 18:47,
18:296, 18:306, 18:329, 18:339,
18:363, 18:367, 32:118, 37:536,
37:539, 37:557, 37:560, 38:37,
38:207, 38:404, 38:414, 39:221,
41:56, 45:195, 45:199, 45:201,
45:217, 45:221, 45:223, 47:37,
54:134, 54:174, 54:193, 54:196,
54:1432, 54:1581, 54:1742, 02:507,
02:508, 02:509, 02:510, 02:511,
02:512, 02:516, 02:518, 02:520, 1291
- \perp 30:458
- \phantom 38:75
- \Phi 30:316
- \phi 30:298
- \Pi 30:313
- \pi 30:293
- picture (env.) 42:21
- \picture 42:21
- Plugs:
- default 55:37
 - floats:footnotes 54:665
 - floats:space:footnotes 54:649
 - footnotes:floats 54:670
 - footnotes:floats:legacy 54:675
 - footnotes:space:floats 54:642
 - kernel (refstepcounter/target) 35:120
 - kernel (tagssupport/recordtarget)
..... 55:47
 - space:floats:footnotes 54:656
 - space:footnotes:floats 54:632
- \pm 30:401

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- \pmatrix [38:173](#), [38:181](#)
- \pmod [38:39](#)
- \PopDefaultHookLabel [08:2789](#), [206](#)
- \poptabs [14:256](#), [41:142](#), [41:161](#)
- \poptracing [26:149](#), [26:341](#)
- \postdisplaypenalty
 - [18:13](#), [38:468](#), [38:480](#), [38:506](#), [02:330](#)
- \pounds [21:338](#), [1301](#)
- \Pr [38:32](#)
- pre commands:
 - pre_shipout_filter [1128](#)
- \prec [30:432](#)
- \preceq [30:435](#)
- \predisdisplaypenalty [18:12](#),
 - [38:467](#), [38:479](#), [38:505](#), [02:329](#), [1351](#)
- \predisplaysize [02:401](#)
- \pretolerance [24:744](#), [24:760](#), [24:805](#), [02:316](#)
- \prevdepth [18:302](#), [18:303](#), [18:335](#),
 - [18:336](#), [18:396](#), [18:401](#), [18:430](#),
 - [18:435](#), [38:205](#), [45:196](#), [45:198](#),
 - [45:218](#), [45:220](#), [54:165](#), [54:167](#),
 - [54:170](#), [02:498](#), [02:502](#), [02:503](#), [442](#)
- \PreviousTotalPages [53:407](#), [1147](#)
- prg commands:
 - \prg_break:n [07:3232](#),
 - [07:3234](#), [07:3236](#), [07:3238](#), [07:3239](#)
 - \prg_break_point: [07:3240](#)
 - \prg_do_nothing:
 - [08:2315](#), [09:210](#), [09:285](#),
 - [09:404](#), [09:623](#), [48:251](#), [52:132](#),
 - [52:134](#), [53:166](#), [53:167](#), [53:172](#),
 - [53:181](#), [53:185](#), [53:321](#), [55:17](#),
 - [55:27](#), [07:410](#), [07:993](#), [07:2030](#), [141](#)
 - \prg_generate_conditional_ -
variant:Nnn
 - [11:83](#), [36:165](#), [36:180](#), [36:195](#)
 - \prg_new_conditional:Npnn
 - [08:2430](#), [08:2445](#),
 - [08:2462](#), [08:2479](#), [08:2485](#), [08:2491](#),
 - [08:2497](#), [08:2506](#), [08:2514](#), [08:2528](#),
 - [08:2546](#), [09:328](#), [10:53](#), [10:117](#),
 - [10:138](#), [11:77](#), [11:84](#), [11:90](#), [36:154](#),
 - [36:169](#), [36:184](#), [48:285](#), [48:292](#),
 - [55:7](#), [08:273](#), [08:882](#), [08:1195](#), [08:1405](#)
 - \prg_new_protected_conditional:Npnn
..... [09:124](#),
 - [09:467](#), [08:733](#), [08:791](#), [08:821](#), [08:853](#)
 - \prg_replicate:nn
 - [08:2298](#), [08:2306](#), [08:2376](#), [08:2392](#),
 - [52:552](#), [07:900](#), [07:943](#), [07:1004](#),
 - [07:1059](#), [07:2530](#), [07:2537](#), [08:1169](#)
 - \prg_return_false: [08:2440](#),
 - [08:2456](#), [08:2474](#), [08:2483](#), [08:2489](#),
 - [08:2495](#), [08:2503](#), [08:2512](#), [08:2521](#),
 - [08:2524](#), [08:2538](#), [08:2553](#), [09:139](#),
 - [09:341](#), [09:482](#), [10:56](#), [10:120](#),
 - [10:142](#), [11:81](#), [11:88](#), [11:94](#), [36:162](#),
 - [36:177](#), [36:192](#), [48:290](#), [48:297](#), [55:8](#),
 - [08:279](#), [08:760](#), [08:814](#), [08:826](#),
 - [08:846](#), [08:858](#), [08:873](#), [08:890](#),
 - [08:894](#), [08:897](#), [08:1200](#), [08:1410](#), [251](#)
 - \prg_return_true:
 - [08:2438](#), [08:2454](#), [08:2473](#), [08:2476](#),
 - [08:2482](#), [08:2488](#), [08:2494](#), [08:2501](#),
 - [08:2511](#), [08:2522](#), [08:2536](#), [08:2551](#),
 - [09:138](#), [09:341](#), [09:481](#), [10:55](#),
 - [10:119](#), [10:141](#), [11:80](#), [11:87](#), [11:93](#),
 - [36:159](#), [36:174](#), [36:189](#), [48:289](#),
 - [48:296](#), [08:278](#), [08:751](#), [08:805](#),
 - [08:844](#), [08:871](#), [08:893](#), [08:1198](#), [08:1408](#)
 - \prime [30:264](#), [30:328](#), [38:256](#)
 - \ProcessedArgument [07:391](#),
 - [07:395](#), [07:402](#), [07:2463](#), [07:2464](#),
 - [07:2481](#), [07:2483](#), [07:2505](#), [07:2514](#),
 - [07:2520](#), [07:2528](#), [07:2545](#), [07:2557](#),
 - [07:2584](#), [07:2650](#), [07:2652](#), [07:3267](#), [183](#)
 - \ProcessKeyOptions ... [51:68](#), [51:200](#), [1091](#)
 - \ProcessKeyPackageOptions [1348](#)
 - \ProcessList [07:3272](#)
 - \ProcessOption* [1285](#)
 - \ProcessOptions [21:1594](#), [26:72](#), [33:805](#),
 - [33:842](#), [50:537](#), [50:628](#), [50:1193](#), [1285](#)
 - \ProcessOptions* [50:537](#)
 - \prod [30:360](#)
 - prop commands:
 - \prop_clear:N
 - [11:144](#), [11:154](#), [11:170](#), [11:180](#),
 - [11:209](#), [11:210](#), [11:412](#), [07:1938](#), [08:1142](#)
 - \prop_clear_new:N [11:123](#)
 - \prop_const_from_keyval:Nn
 - [08:936](#), [08:944](#), [08:946](#), [08:948](#)
 - \prop_gclear:N [09:75](#), [11:110](#),
 - [08:969](#), [08:1004](#), [08:1426](#), [08:1453](#)
 - \prop_gclear_new:N
 - [08:2417](#), [08:2427](#), [11:99](#), [11:131](#)
 - \prop_get:NnN [11:44](#),
 - [11:557](#), [11:560](#), [11:587](#), [11:698](#),
 - [11:719](#), [11:800](#), [08:566](#), [08:1596](#), [08:1682](#)
 - \prop_get:NnNTF .. [11:425](#), [11:538](#),
 - [11:629](#), [11:724](#), [11:726](#), [11:811](#),
 - [11:884](#), [51:205](#), [07:1944](#), [08:575](#), [08:620](#)
 - \prop_gpop:NnNTF [08:977](#), [08:1012](#)
 - \prop_gput:Nnn
 - [08:2581](#), [08:2582](#), [08:2583](#),
 - [09:68](#), [10:181](#), [11:195](#), [11:1162](#),
 - [48:299](#), [07:57](#), [55:52](#), [08:576](#), [08:577](#),
 - [08:623](#), [08:626](#), [08:1503](#), [08:1541](#)

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- `\prop_gset_eq:NN`
..... 11:100, 11:113, 11:132, 08:1144
- `\prop_if_empty:NTF`
.. 08:1973, 08:2450, 08:1483, 08:1518
- `\prop_if_empty_p:N` .. 08:1869, 08:2467
- `\prop_if_exist:NTF`
..... 08:2077, 08:2449, 08:2487,
11:107, 11:138, 11:148, 11:164, 11:174
- `\prop_if_exist_p:N` 08:1868
- `\prop_if_in:NnTF` ... 11:67, 11:184,
11:551, 11:555, 08:828, 08:836,
08:840, 08:860, 08:863, 08:867, 08:892
- `\prop_map_break:`
..... 08:2085, 08:1573, 08:1655
- `\prop_map_function:NN`
.. 09:74, 11:953, 11:964, 08:1143, 261
- `\prop_map_inline:Nn`
..... 08:1811, 08:1872, 08:1976,
08:2080, 08:2082, 11:416, 51:240,
08:1438, 08:1459, 08:1561, 08:1568,
08:1570, 08:1643, 08:1650, 08:1652
- `\prop_new:N` ... 09:17, 11:18, 11:28,
11:30, 11:31, 11:112, 07:52, 08:30,
08:31, 08:152, 08:164, 08:1293, 08:1294
- `\prop_put:Nnn` 11:261, 11:353, 11:355,
11:360, 11:474, 11:488, 11:503,
11:532, 11:598, 11:667, 07:1935,
08:37, 08:42, 08:1178, 08:1781, 08:1792
- `\prop_remove:Nn` 11:431
- `\prop_set_eq:NN` .. 11:124, 11:141,
11:151, 11:167, 11:177, 08:1499, 08:1536
- `\prop_show:N` 286
- property commands:
- `\property_gset:nnnn`
..... 36:6, 36:17, 36:46, 806
- `\property_if_exist:n` . 36:154, 36:165
- `\property_if_exist:NTF`
.. 36:154, 36:166, 36:167, 36:168, 808
- `\property_if_exist_p:n` ... 36:154, 808
- `\property_if_recorded:n` 36:169, 36:180
- `\property_if_recorded:nn`
..... 36:184, 36:195
- `\property_if_recorded:nnTF` 36:184,
36:196, 36:197, 36:198, 36:218, 808
- `\property_if_recorded:NTF`
..... 36:169, 36:181,
36:182, 36:183, 36:208, 36:216, 808
- `\property_if_recorded_p:n` 36:169, 808
- `\property_if_recorded_p:nn` 36:184, 808
- `\property_new:nnnn`
.. 36:6, 36:6, 36:41, 36:232, 36:234,
36:235, 36:236, 36:237, 36:239,
36:241, 36:242, 36:243, 36:244, 806
- `\property_record:nN` . 36:48, 36:48, 807
- `\property_record:nn`
.. 36:48, 36:49, 36:50, 36:52, 36:85, 807
- `\property_ref:nn`
..... 36:89, 36:89, 36:96, 36:122, 807
- `\property_ref:nnn`
..... 36:97, 36:97, 36:117, 36:125, 807
- `\property_ref_undefined_warn:` ...
..... 36:199, 36:199, 807
- `\property_ref_undefined_warn:n` ..
..... 36:206, 36:206, 807
- `\property_ref_undefined_warn:nn` .
.. 36:214, 36:214, 36:230, 36:231, 807
- property internal commands:
- `__property_data:nnn`
..... 36:128, 36:140, 36:142, 36:147
- `__property_gset:nnnn`
..... 36:6, 36:10, 36:19, 36:22, 36:37
- `__property_record:nn`
..... 36:48, 36:51, 36:53, 36:62
- `__property_record_value:n`
..... 36:48, 36:59, 36:63
- `__property_record_value_aux:n` ..
..... 36:48, 36:64, 36:65, 36:79
- `__property_ref:nnn`
... 36:91, 36:97, 36:99, 36:104, 36:116
- `l__property_ref_flag` 36:88
- `\propto` 30:409
- `\protect` 14:246, 14:248, 14:250,
14:256, 14:262, 14:269, 14:279,
14:282, 14:288, 20:211, 21:44, 21:50,
21:69, 21:73, 21:229, 21:237, 28:706,
28:1254, 29:639, 32:143, 35:16,
35:33, 35:50, 06:102, 37:280, 37:283,
37:317, 37:327, 06:232, 06:246,
06:255, 06:260, 06:263, 06:264,
06:266, 06:267, 06:272, 06:273,
06:278, 06:281, 06:282, 06:307,
41:264, 06:345, 06:373, 44:12, 44:72,
44:82, 44:164, 44:171, 44:177, 45:17,
47:5, 06:580, 06:600, 52:281, 52:304,
53:83, 53:95, 53:125, 53:132, 53:152,
54:798, 54:874, 54:933, 57:358, 1343
- `\protected` 17:6, 18:57, 18:238, 18:289,
18:474, 18:538, 04:186, 21:328,
21:329, 22:15, 22:247, 25:3202,
25:3206, 25:3209, 25:3212, 25:3215,
25:3218, 25:3221, 25:3224, 28:1096,
29:574, 06:7, 37:201, 37:247, 37:417,
38:199, 38:388, 38:426, 38:445,
38:446, 41:56, 41:201, 41:208,
42:142, 06:455, 53:496, 53:497,
53:498, 54:793, 54:794, 02:463, 1333
- `\Provide` 1304

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- \providecommand 08:2914,
08:2917, 21:6, 21:13, 21:1003,
35:98, 35:99, 35:100, 06:178, 54:2504
 - \ProvideCommandCopy 98
 - \ProvideDocumentCommand 07:3071
 - \ProvideDocumentEnvironment ... 07:3107
 - \ProvideExpandableDocumentCommand ..
..... 07:3190
 - \ProvideHook 08:2860, 1344
 - \ProvideMirroredHookPair 08:2860
 - \ProvideReversedHook 08:2860
 - provides commands:
 provides_module 04:298
 - \provides_module 45
 - \ProvidesClass 1040
 - \ProvidesClass 50:421
 - \ProvidesExplPackage 52:541
 - \ProvidesFile .. 30:676, 30:678, 30:679,
30:680, 33:1178, 50:430, 01:73, 1313
 - \ProvidesPackage 1040
 - \ProvidesPackage 26:13,
33:779, 33:810, 50:344, 50:423,
50:425, 50:1806, 52:570, 53:491, 1323
 - \ProvideTextCommand 21:3, 21:78, 1304
 - \ProvideTextCommandDefault .. 21:75, 1313
 - \Psi 30:317
 - \psi 30:300
 - \PushDefaultHookLabel 08:2789, 206
 - \pushtabs
14:256, 41:139, 41:138, 41:158, 41:160
 - \pushtracing 26:118, 26:322
 - \put 42:56, 42:58, 42:70, 42:72,
42:325, 42:326, 42:327, 42:328,
42:336, 42:338, 42:351, 42:352,
42:353, 42:354, 42:361, 42:364,
42:384, 42:385, 42:386, 42:387,
42:393, 42:395, 42:407, 42:408,
42:409, 42:410, 42:415, 42:420,
42:728, 42:784, 42:812, 42:829, 1131
- Q**
- \qbezier 922
 - \qbezier 42:681, 42:813, 42:830
 - \qbeziermax . 42:680, 42:706, 42:707, 42:767
 - \qqquad 18:549
 - \quad 18:549,
38:168, 38:170, 38:190, 44:111, 1311
 - quark commands:
 \q_mark 07:1334,
07:1337, 07:2364, 07:2368, 07:2380,
07:2710, 07:2718, 07:2721, 07:2726
 - \q_nil 11:79, 11:92, 07:604, 07:1326,
07:1337, 07:1504, 07:1751, 07:1752,
07:1754, 07:1755, 07:1756, 07:1757,
07:1758, 07:1844, 07:1865, 07:1869,
07:1875, 07:1884, 07:1886, 07:2284,
07:2308, 07:2317, 07:2349, 07:2364,
07:2368, 07:2380, 07:2386, 07:2412, 175
 - \quark_if_nil:NTF 07:615, 07:2370, 176
 - \quark_if_nil:nTF 07:1501
 - \quark_if_recursion_tail_stop:N .
..... 07:822, 07:1485
 - \quark_if_recursion_tail_stop:n .
.... 36:144, 51:7, 07:482, 07:672,
07:1252, 07:2723, 07:2724, 08:422
 - \quark_if_recursion_tail_stop_-
do:Nn 07:623
 - \quark_if_recursion_tail_stop_-
do:nn 09:423,
09:442, 07:508, 07:513, 07:522,
07:531, 07:577, 07:590, 07:600, 07:644
 - \quark_new:N 09:15,
09:16, 11:40, 07:49, 07:50, 07:1853
 - \q_recursion_stop
... 09:412, 36:140, 51:20, 07:335,
07:346, 07:459, 07:806, 07:1248,
07:1481, 07:1936, 07:1951, 07:1953,
07:1960, 07:2713, 07:2719, 08:428
 - \q_recursion_tail
..... 09:412, 36:140, 51:19,
07:459, 07:806, 07:1248, 07:1481,
07:1951, 07:1955, 07:2709, 07:2712,
07:2718, 07:2719, 08:427, 08:428, 129
 - \q_stop 11:79,
11:92, 07:423, 07:604, 07:613,
07:618, 07:1334, 07:1336, 07:1752,
07:1758, 07:1869, 07:1886, 07:2521
 - quark internal commands:
 \q__cmd ... 07:407, 07:411, 07:1028,
07:1364, 07:1365, 07:1447, 07:1455,
07:1851, 07:2268, 07:2269, 07:2272,
07:2277, 07:2283, 07:2284, 07:2286,
07:2290, 07:2295, 07:2298, 07:2301,
07:2312, 07:2316, 07:2317, 07:2320,
07:2321, 07:2324, 07:2335, 07:2338,
07:2342, 07:2353, 07:2355, 07:2356,
07:2357, 07:2358, 07:2359, 07:2360,
07:2361, 07:2365, 07:2384, 07:2385,
07:2386, 07:2387, 07:2388, 07:2389,
07:2390, 07:2391, 07:2392, 07:2393,
07:2394, 07:2395, 07:2396, 07:2397,
07:2400, 07:2405, 07:2411, 07:2412,
07:2418, 07:2423, 07:2424, 07:2427,
07:2438, 07:2445, 07:2450, 07:2451,
07:2452, 07:2455, 07:2456, 07:2458, 177
 - \q__cmd_recursion_stop
..... 07:50, 07:2590,
07:2592, 07:2601, 07:2624, 07:2633, 116

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=ltthyphen.dtx, 57=ltfinal.dtx

- `\q_cmd_recursion_tail`
..... [07:49](#), [07:2590](#), [116](#)
 - `\q_hook_recursion_stop`
..... [09:15](#), [09:343](#), [09:344](#), [09:353](#), [09:378](#)
 - `\q_hook_recursion_tail`
..... [09:15](#), [09:343](#), [09:357](#)
 - `\q_template_nil` ... [11:40](#), [11:888](#), [361](#)
 - `\quotedblbase` [21:576](#), [21:805](#), [21:1198](#)
 - `\quotingslbase` [21:577](#), [21:1195](#)
- R**
- `\r` [21:256](#), [21:408](#), [21:451](#),
[21:491](#), [21:631](#), [21:658](#), [21:668](#),
[21:694](#), [21:778](#), [21:817](#), [21:1267](#),
[21:1285](#), [21:1311](#), [21:1433](#), [21:1434](#),
[33:147](#), [33:169](#), [02:443](#), [02:444](#), [1300](#)
 - `\radical` .. [28:1120](#), [28:1123](#), [28:1153](#), [1332](#)
 - `\raggedbottom` [49:126](#), [1181](#)
 - `\raggedleft` . [37:460](#), [37:478](#), [37:489](#), [37:496](#)
 - `\raggedright` [37:455](#), [37:474](#), [37:488](#), [37:494](#)
 - `\raise` .. [21:347](#), [21:379](#), [21:450](#), [21:453](#),
[21:754](#), [21:819](#), [21:917](#), [21:1275](#),
[29:652](#), [30:476](#), [30:524](#), [30:526](#),
[38:73](#), [40:579](#), [40:588](#), [42:61](#), [42:73](#),
[42:105](#), [42:117](#), [42:195](#), [42:305](#),
[42:452](#), [42:494](#), [42:525](#), [42:550](#),
[42:618](#), [42:635](#), [42:636](#), [42:739](#), [42:795](#)
 - `\raisebox` [879](#)
 - `\raisebox` [21:894](#), [21:1244](#), [40:556](#)
 - `\rangle` [30:603](#)
 - `\RawIndent` [16:136](#), [16:172](#), [415](#)
 - `\RawNoIndent` [16:136](#)
 - `\RawNoindent` [16:137](#), [16:173](#), [415](#)
 - `\RawParEnd` ... [16:136](#), [16:174](#), [02:482](#), [415](#)
 - `\RawShipout` [53:151](#), [53:438](#), [1126](#)
 - `\rbrace` [21:329](#), [30:607](#)
 - `\rbrack` [02:441](#)
 - `\rceil` [30:611](#)
 - `\Re` [30:324](#)
 - `\read` [471](#)
 - `\ReadOnlyShipoutCounter`
..... [53:346](#), [53:440](#), [1130](#)
 - `\RecordProperties` [36:80](#), [36:270](#), [808](#)
 - `\Ref` [798](#)
 - `\Ref` [35:113](#), [35:156](#), [35:158](#), [35:178](#),
[35:186](#), [35:190](#), [35:197](#), [35:201](#), [35:208](#)
 - `\ref` [35:10](#), [35:186](#), [35:197](#), [45:553](#), [875](#)
 - `\RefProperty` [36:118](#), [36:271](#), [807](#)
 - `\refstepcounter` [521](#)
 - `refstepcounter` (socket) [35:118](#)
 - `\refstepcounter`
... [35:111](#), [35:112](#), [35:180](#), [35:190](#),
[35:192](#), [35:201](#), [35:203](#), [38:350](#),
[38:507](#), [39:233](#), [43:37](#), [44:59](#), [45:9](#), [802](#)
 - `refstepcounter/target` (socket) ... [35:119](#)
 - `\RefUndefinedWarn` ... [36:214](#), [36:272](#), [809](#)
 - regex commands:
 `\regex_match:nnTF` [11:385](#)
 - `\registernumber` [44](#)
 - `registernumber` [04:389](#)
 - `\relax` [1069](#)
 - `\relax_` [1298](#)
 - `\Relbar` [30:487](#), [30:495](#), [30:497](#), [30:503](#), [1326](#)
 - `\relbar` [30:484](#), [30:499](#), [30:501](#)
 - `\relpenalty` [02:324](#)
 - remove commands:
 `remove_from_callback` [04:907](#)
 - `\remove_from_callback` [46](#)
 - `\RemoveFromHook`
... [08:2787](#), [08:2916](#), [37:39](#), [37:40](#),
[37:41](#), [37:104](#), [37:105](#), [37:106](#), [202](#)
 - `\removelastskip`
..... [02:514](#), [02:516](#), [02:518](#), [02:520](#)
 - `\renewcommand` [81](#)
 - `\renewcommand` [30:78](#), [30:80](#), [30:82](#),
[30:83](#), [30:85](#), [30:87](#), [30:89](#), [30:90](#),
[30:96](#), [30:98](#), [30:100](#), [30:101](#), [30:114](#),
[30:115](#), [30:116](#), [30:124](#), [30:125](#),
[06:124](#), [38:455](#), [38:475](#), [38:496](#), [728](#)
 - `\RenewCommandCopy` [06:473](#), [06:475](#), [98](#)
 - `\RenewDocumentCommand` [07:3071](#)
 - `\RenewDocumentEnvironment`
..... [07:2869](#), [07:3107](#)
 - `\renewenvironment` [82](#)
 - `\renewenvironment` . [06:152](#), [38:504](#), [38:516](#)
 - `\RenewEnvironmentCopy` [06:514](#), [06:516](#), [1351](#)
 - `\RenewExpandableDocumentCommand`
..... [48:508](#),
[48:509](#), [48:510](#), [48:511](#), [07:3190](#), [196](#)
 - `\repeat` [04:154](#), [04:164](#), [24:786](#),
[41:382](#), [50:1246](#), [50:1307](#), [50:1377](#),
[50:1440](#), [50:1491](#), [50:1529](#), [57:369](#),
[57:380](#), [57:390](#), [57:401](#), [57:431](#),
[57:457](#), [57:467](#), [01:65](#), [01:67](#), [02:490](#)
 - `\requestedLaTeXdate` [08:2936](#),
[50:1592](#), [50:1632](#), [50:1652](#), [50:1739](#)
 - `\requestedpatchdate` [50:1662](#), [50:1740](#)
 - `\RequirePackage` [1040](#)
 - `\RequirePackage` [04:24](#), [50:676](#), [50:683](#),
[50:726](#), [50:735](#), [50:1189](#), [54:2501](#), [1285](#)
 - `\RequirePackageWithOptions` [1040](#)
 - `\RequirePackageWithOptions` [50:718](#)
 - reserved@a commands:
 `\reserved@a:` [20:233](#),
[20:305](#), [20:367](#), [20:414](#), [50:1219](#), [50:1354](#)
 - reserved@b commands:
 `\reserved@b:` [50:232](#), [50:249](#)

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- reserved@c commands:
- \reserved@c: 20:755
 - \RestoreAtCatcode 1296
 - \restorecr 18:556
 - \ResumeTagging 1243
 - \ResumeTagging 23:22, 55:3, 1243
 - \ReverseBoolean 07:3268
 - \reversemarginpar 45:387
 - \rfloor 30:615
 - \rgroup 30:619
 - \rhd 29:734
 - \rho 30:294
 - \rhook 30:490, 30:491
 - \right 30:636, 30:638,
30:640, 30:642, 30:647, 30:648,
30:649, 30:650, 38:168, 38:173, 38:197
 - \Rightarrow 30:422, 30:497, 30:509
 - \rightarrow 30:449, 30:451,
30:455, 30:489, 30:499, 30:507, 30:560
 - \rightarrowfill 30:542, 30:558
 - \rightharpoondown 30:466
 - \rightharpoonup 30:465, 30:477
 - \righthyphenmin 56:11, 02:371
 - \rightleftharpoons 30:475
 - \rightline 40:614
 - \rightmargin 39:9, 39:40, 39:51, 862
 - \rightmark 49:106, 1289
 - \rightskip 24:757, 37:452, 37:456,
37:462, 37:472, 37:475, 37:480,
37:529, 37:551, 39:75, 40:392,
40:413, 44:232, 44:255, 02:417, 02:528
 - \rlap 21:450, 21:453, 21:819,
38:428, 38:456, 40:618, 41:81, 1312
 - \rm 1288
 - \rmdefault 29:7, 29:209,
29:399, 29:408, 29:440, 29:451,
29:483, 30:62, 30:131, 33:7, 33:577, 702
 - \rmfamily 29:5, 29:6, 29:406, 29:449,
29:450, 29:481, 29:482, 32:15, 1311
 - rmfamily 29:421
 - \rmmath 1281
 - \rmoustache 30:576
 - \rmsubstdefault
. 30:18, 30:30, 33:28, 33:39, 33:49
 - \Roman 521
 - \Roman 22:191, 1129
 - \roman 521
 - \roman 22:190
 - \romannumeral
. 22:196, 22:197, 37:316, 37:348,
37:365, 39:43, 39:265, 39:276, 830
 - \root 38:66, 38:356
 - \rootbox 38:66
 - \rq 02:439
 - \rule 879
 - \rule 40:490, 40:507,
40:524, 40:530, 45:477, 45:495, 45:513
- S
- \S 21:331, 1305
 - \safesubencodingfoundfalse 33:1219
 - \safesubencodingfoundtrue 33:1209
 - \samepage 430
 - \samepage 18:11, 18:29, 1282
 - \SaveAtCatcode 1299
 - \savebox 878
 - \savebox 40:155
 - \savecatcodetable 04:117, 04:168, 04:170
 - \savepos 810
 - \sb 38:212
 - \sbox 878
 - \sbox 19:4, 21:519,
21:535, 39:236, 40:161, 40:168,
40:172, 40:177, 40:182, 02:522, 1303
- scan commands:
- \scan_new:N 11:38, 11:39, 08:44
 - \scan_stop: 08:2547, 08:2548,
09:469, 09:470, 09:528, 09:529,
09:564, 09:565, 16:94, 48:105,
48:107, 48:109, 48:110, 48:266,
48:267, 48:268, 51:88, 51:100,
53:44, 53:325, 53:327, 53:328,
07:1602, 07:1603, 07:1625, 07:1631,
07:1640, 07:1647, 07:2083, 07:2084,
08:662, 08:669, 08:680, 08:689,
08:724, 08:735, 08:771, 08:783,
08:793, 08:823, 08:855, 08:1197, 1095
- scan internal commands:
- \s__file_stop 52:103,
52:105, 52:108, 52:110, 52:112, 52:125
 - \s__hook_mark 08:2264,
08:2278, 08:2327, 08:2335, 08:2438,
08:2440, 08:2501, 08:2503, 08:2507,
08:2508, 08:2515, 08:2516, 08:2529,
08:2530, 09:24, 09:26, 09:41, 09:43,
09:52, 09:54, 09:130, 09:135, 09:329,
09:340, 09:474, 09:477, 09:489,
09:493, 09:498, 09:533, 09:553,
09:569, 09:581, 08:44, 08:45,
08:46, 08:348, 08:351, 08:354,
08:358, 08:361, 08:362, 08:713,
08:764, 08:766, 08:774, 08:776,
08:779, 08:781, 08:883, 08:908,
08:909, 08:913, 08:1218, 08:1235,
08:1239, 08:1281, 08:1282, 08:1337, 229
 - \s__keys_stop 51:47, 51:178, 51:179
 - \s__template_mark 11:38, 11:888, 361

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpara.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- `\s__template_stop` 11:39, 11:286, 11:297, 11:308, 11:329, 11:343, 11:351, 11:437, 11:440, 11:747, 11:750, 11:888, 11:892, 361
- `\scdefault` 25:3214, 29:28, 30:106
- `\scriptfont` 26:339
- `\scriptscriptfont` 26:340
- `\scriptscriptstyle` 38:65, 38:68
- `\scriptspace` 02:399
- `\scriptstyle` 30:348, 38:64
- `\scshape` 21:320, 25:3212, 25:3213, 29:26, 29:27, 32:23, 626
- `\searrow` 30:417
- `\sec` 38:20
- `\secdef` 44:142
- `\secondoftwo` 466
- `\section` 350
- `\sectionmark` 44:143, 1009
- `\selectfont` 19:7, 21:322, 21:349, 21:380, 21:469, 21:831, 21:893, 21:1243, 21:1277, 21:1595, 24:354, 24:364, 24:374, 25:2916, 25:2921, 25:2926, 25:3204, 25:3208, 25:3211, 25:3214, 25:3217, 25:3220, 25:3223, 25:3226, 26:115, 26:116, 26:160, 26:163, 26:165, 29:7, 29:10, 29:13, 29:16, 29:19, 29:22, 29:25, 29:28, 29:31, 29:265, 29:288, 29:326, 29:346, 29:361, 29:372, 29:383, 29:386, 29:410, 29:415, 29:420, 29:453, 29:458, 29:463, 29:477, 29:480, 29:483, 29:486, 29:489, 29:565, 29:642, 29:666, 29:683, 29:698, 33:36, 33:58, 33:68, 33:600, 33:637, 33:877, 33:894, 45:403, 45:424, 709
- `selectfont` 26:151
- seq commands:
 - `\seq_clear:N` 11:160, 11:211, 08:1558, 08:1640
 - `\seq_clear_new:N` 08:1566, 08:1648
 - `\seq_const_from_clist:Nn` 11:16
 - `\seq_count:N` 55:191
 - `\seq_gclear_new:N` 11:115
 - `\seq_gpop:NN` 08:2578
 - `\seq_gpop:NNTF` .. 52:72, 08:451, 08:459
 - `\seq_gpop_right:NN` 08:430
 - `\seq_gpush:Nn` 08:2566, 08:2572, 52:64, 08:444
 - `\seq_gput_right:Nn` 48:23, 08:108, 08:137, 08:419, 08:423
 - `\seq_gset_eq:NN` 11:116
 - `\seq_if_empty:NTF` 08:418, 08:469
 - `\seq_if_exist:NTF` 11:155, 52:60
 - `\seq_if_in:NnTF` . 11:230, 48:9, 48:234
 - `\seq_map_break:` 11:249, 11:340
 - `\seq_map_function:NN` 11:226, 11:346, 11:714
 - `\seq_map_inline:Nn` 08:2959, 48:125, 48:158, 48:200, 48:211, 48:222, 48:376, 48:453, 07:2482, 08:1427, 08:1454, 08:1582, 08:1600, 08:1665, 08:1688
 - `\seq_mapthread_function:NNN` ... 185
 - `\seq_new:N` 08:2556, 10:36, 11:29, 48:6, 52:61, 55:194, 07:2466, 08:28, 08:33, 08:1549
 - `\seq_put_right:Nn` .. 10:91, 11:263, 08:1564, 08:1646, 08:1749, 08:1756
 - `\seq_set_eq:NN` 11:157
 - `\seq_set_split:Nnn` ... 55:190, 07:2480
 - `\seq_use:Nnnn` . 08:1809, 08:1814, 10:65
- seq internal commands:
 - `\g__mark_classes_seq` 48:6, 48:9, 48:23, 48:125, 48:158, 48:200, 48:211, 48:222, 48:234, 48:376, 48:453, 1013
 - `\series` 1281
 - `\seriesdefault` 21:1596, 28:406, 29:221, 29:223, 29:661, 29:679, 29:695, 29:712, 29:774, 30:131, 703
 - `\setattribute` 43
 - `\setattribute` 04:82, 04:239
 - `\setbox` 871
 - `\setbox...` 1320
 - `\setbox0` 1317
 - `\setcounter` 521
 - `\setcounter` 20:445, 22:2, 22:44, 22:64, 28:144, 39:256, 54:2914, 54:2917, 54:2920, 54:2924, 668
 - `\SetDefaultHookLabel` 08:2789, 206
 - `\SetKeys` 20:751, 51:258, 1092
 - `\SetKnownTemplateKeys` 11:1201, 355
 - `\setlength` 531
 - `\setlength` 18:84, 18:244, 18:259, 18:518, 23:4, 38:524, 38:529, 38:530, 38:531, 40:47, 40:62, 40:247, 40:266, 40:333, 40:336, 40:360, 40:363, 40:439, 40:546, 40:547, 40:548, 40:577, 40:578, 40:585, 40:586, 40:587, 41:176, 41:384, 54:2930, 54:2931, 54:2932, 54:2935, 54:2936, 54:2940, 54:2941, 54:2942, 54:2946, 54:2947, 54:2948, 1324
 - `\SetMathAlphabet` 24:12, 27:140, 27:141, 28:711, 30:166, 30:167
 - `\setminus` 30:404
 - `\SetProperty` 36:38, 36:268, 806

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpara.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- `\setrangeatcode`
..... 04:96, 04:104, 04:113, 04:114
- `\SetSymbolFont`
..... 28:546, 30:156, 30:157, 30:158, 1285
- `\SetTemplateKeys` 11:1201, 354
- `\settodepth` 531
- `\settodepth` 23:17
- `\settoheight` 531
- `\settoheight` 23:17
- `\settowidth` 531
- `\settowidth` 23:17
- `\sf` 1288
- `\sfcode` 18:463, 20:45,
20:115, 20:173, 57:250, 57:545,
02:431, 02:432, 02:433, 02:434, 02:551
- `\sfdefault` 29:10, 29:213, 29:400,
29:413, 29:441, 29:456, 29:486, 30:62
- `\sffamily` .. 29:8, 29:9, 29:411, 29:454,
29:455, 29:484, 29:485, 32:16, 712
- `sffamily` 29:421
- `\sfsubstdefault` 30:19, 30:31, 33:30, 33:51
- `\shape` 1283
- `\shapedefault`
..... 21:1596, 25:3204, 28:407, 29:662,
29:680, 29:696, 29:713, 30:131, 703
- `\sharp` 30:341
- `\shipout` 53:4, 53:52, 53:431,
53:434, 54:805, 54:880, 54:938, 1124
- `shipout` 53:153
- shipout commands:
 - `\l_shipout_box`
..... 53:23, 53:35, 53:38, 53:50, 53:61,
53:126, 53:149, 53:179, 53:184,
53:207, 53:208, 53:215, 53:226,
53:229, 53:230, 53:234, 53:241,
53:251, 53:259, 53:266, 53:276,
53:282, 53:283, 53:286, 53:293,
53:295, 53:296, 53:302, 53:304, 1136
 - `\l_shipout_box_dp_dim`
..... 53:194, 53:197,
53:199, 53:230, 53:283, 53:512, 1125
 - `\l_shipout_box_ht_dim`
..... 53:193, 53:197, 53:199,
53:229, 53:249, 53:282, 53:511, 1125
 - `\l_shipout_box_ht_plus_dp_dim` ...
..... 53:196, 53:199,
53:215, 53:266, 53:277, 53:279, 1125
 - `\l_shipout_box_wd_dim` ... 53:195,
53:199, 53:241, 53:293, 53:513, 1125
 - `\shipout_debug_off:`
..... 53:7, 53:13, 53:415, 1129
 - `\shipout_debug_on:`
..... 53:7, 53:8, 53:414, 1129
- `\shipout_discard:`
..... 53:343, 53:343, 53:411, 1128
- `\g_shipout_readonly_int`
..... 36:233, 53:102, 53:104,
53:111, 53:116, 53:346, 53:355,
53:359, 53:363, 53:367, 53:371, 1129
- `\g_shipout_totalpage_int` 1129
- `\g_shipout_totalpages_int`
..... 53:87, 53:348, 1129
- shipout internal commands:
 - `__shipout_add_background_box:n` .
53:180, 53:206, 53:206, 53:338, 53:420
 - `__shipout_add_background_-
picture:n`
..... 53:69, 53:337, 53:337, 53:424
 - `__shipout_add_firstpage_-
material:Nn`
..... 53:173, 53:189, 53:189, 53:413, 53:418
 - `__shipout_add_firstpage_-
specials:` 53:110,
53:166, 53:178, 53:178, 53:181, 1138
 - `__shipout_add_foreground_box:n` .
53:117, 53:257, 53:257, 53:341, 53:422
 - `__shipout_add_foreground_-
picture:n`
..... 53:64, 53:340, 53:340, 53:426
 - `__shipout_debug:n` 53:7, 53:7,
53:20, 53:103, 53:115, 53:148, 1132
 - `\g_shipout_debug_bool`
..... 53:6, 53:10, 53:15, 53:21
 - `__shipout_debug_gset:`
..... 53:7, 53:11, 53:16, 53:18
 - `\g_shipout_discard_bool`
... 53:81, 53:88, 53:90, 53:203, 53:344
 - `__shipout_drop_firstpage_-
specials:` 53:127,
53:167, 53:178, 53:183, 53:185, 1138
 - `__shipout_excuse_extra_page:` ...
..... 53:384, 53:392, 53:392
 - `__shipout_execute:`
..... 53:46, 53:46, 53:52, 1133
 - `__shipout_execute_cont:`
..... 53:57, 53:59, 53:59
 - `__shipout_execute_main_cont:Nnnn`
.... 53:60, 53:77, 53:77, 53:147, 1137
 - `__shipout_execute_nohooks_cont:`
..... 53:144, 53:146
 - `__shipout_execute_raw:`
..... 53:135, 53:135, 53:151, 1137
 - `__shipout_execute_test_level:` ..
..... 53:49, 53:54, 53:54
 - `__shipout_execute_test_level_-
raw:` 53:135, 53:138, 53:141

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=ltthyphen.dtx, 57=ltfinal.dtx

- `__shipout_finalize_box:`
..... [53:26](#), [53:28](#), [53:44](#), [53:124](#)
- `\l__shipout_firstpage_box`
..... [53:170](#), [53:180](#), [53:187](#), [1139](#)
- `__shipout_get_box_size:N` . [53:85](#),
[53:107](#), [53:192](#), [53:192](#), [53:207](#), [1140](#)
- `\l__shipout_group_level_tl`
... [53:47](#), [53:53](#), [53:56](#), [53:136](#), [53:143](#)
- `\c__shipout_horigin_tl` [53:310](#), [53:325](#)
- `__shipout_init_page_origins:` ...
..... [53:310](#), [53:310](#), [53:321](#), [53:324](#)
- `\g__shipout_lastpage_handled_`-
bool [53:120](#), [53:188](#), [53:365](#)
- `__shipout_picture_overlay:n` ...
..... [53:323](#), [53:323](#), [53:338](#), [53:341](#)
- `\l__shipout_raw_box` [53:25](#), [53:139](#),
[53:147](#), [53:149](#), [53:179](#), [53:184](#), [1136](#)
- `__shipout_run_firstpage_hook:` ..
.. [53:108](#), [53:163](#), [53:163](#), [53:172](#), [1139](#)
- `\l__shipout_saved_badness_tl` ...
... [53:204](#), [53:210](#), [53:218](#), [53:231](#),
[53:236](#), [53:244](#), [53:253](#), [53:261](#),
[53:269](#), [53:281](#), [53:288](#), [53:298](#), [53:306](#)
- `__shipout_saved_protect:`
..... [53:83](#), [53:132](#), [53:152](#), [53:152](#)
- `\l__shipout_tmp_box`
... [53:204](#), [53:217](#), [53:219](#), [53:220](#),
[53:221](#), [53:225](#), [53:243](#), [53:245](#),
[53:246](#), [53:247](#), [53:250](#), [53:268](#),
[53:270](#), [53:271](#), [53:272](#), [53:278](#),
[53:297](#), [53:299](#), [53:300](#), [53:301](#),
[53:303](#), [53:329](#), [53:331](#), [53:332](#), [53:333](#)
- `\c__shipout_vorigin_tl` [53:310](#), [53:327](#)
- `shipout/after` [53:153](#), [1125](#)
- `shipout/background` [53:153](#), [1125](#)
- `shipout/before` [53:153](#), [1125](#)
- `shipout/firstpage` [53:153](#), [1125](#)
- `shipout/foreground` [53:153](#), [1125](#)
- `shipout/lastpage` [53:153](#), [1125](#)
- `\ShipoutBox` ... [53:23](#), [53:439](#), [53:494](#), [1139](#)
- `\ShipoutBoxDepth` [53:459](#), [53:511](#)
- `\ShipoutBoxHeight` .. [53:458](#), [53:511](#), [1151](#)
- `\ShipoutBoxWidth` [53:460](#), [53:511](#)
- `\shortstack`
[42:132](#), [42:147](#), [42:152](#), [42:814](#), [42:831](#)
- `\show` [06:552](#), [06:625](#), [103](#)
- show commands:
`\show_hook:n` [280](#)
- `\showbox` [54:2415](#)
- `\showboxbreadth` [02:631](#), [02:655](#),
[02:672](#), [02:697](#), [54:2415](#), [02:387](#), [02:554](#)
- `\showboxdepth` [02:630](#),
[02:654](#), [02:671](#), [02:698](#), [24:744](#),
[24:789](#), [24:806](#), [54:2415](#), [02:388](#), [02:554](#)
- `\ShowCommand` . [06:545](#), [07:1345](#), [07:1374](#),
[07:1386](#), [07:1394](#), [07:1397](#), [07:1558](#), [96](#)
- `\ShowEnvironment` [06:680](#), [105](#)
- `\ShowFloat` [54:2396](#)
- `\ShowHook` [08:2813](#), [08:2921](#), [210](#)
- `\showhyphens` [24:733](#), [1355](#)
- `\ShowInstanceValues` [11:1183](#), [358](#)
- `\ShowMarksAt` [48:380](#)
- `\showoutput` [02:553](#), [1314](#)
- `\showoverfull` [02:577](#),
[02:612](#), [02:620](#), [02:552](#), [02:555](#), [1314](#)
- `\ShowSocket` ... [10:161](#), [10:182](#), [10:204](#), [339](#)
- `\ShowTemplateCode` [11:1183](#), [358](#)
- `\ShowTemplateDefaults` [11:1183](#), [359](#)
- `\ShowTemplateInterface` [11:1183](#), [359](#)
- `\ShowTemplateVariables` [11:1183](#), [359](#)
- `\showtokens` [06:660](#), [104](#)
- `\Sigma` [30:314](#)
- `\sigma` [30:295](#)
- `\sim` [30:456](#), [30:468](#)
- `\simeq` [30:457](#)
- `\sin` [38:9](#)
- `\sinh` [38:11](#)
- `\size` [1281](#)
- `\skew` [30:555](#)
- `\skip` [04:33](#), [40:464](#), [45:391](#),
[02:28](#), [54:314](#), [54:588](#), [54:697](#),
[02:53](#), [02:92](#), [02:209](#), [02:250](#), [02:295](#)
- skip commands:
`\skip_eval:n` [11:370](#), [05:165](#)
- `\skip_new:N` [11:36](#)
- `\skip_set:Nn` [16:25](#)
- `\skip_zero:N` [16:58](#), [53:222](#),
[53:223](#), [53:224](#), [53:273](#), [53:274](#), [53:275](#)
- `\l_tmpa_skip` [354](#)
- `\c_zero_skip` [16:26](#)
- `\skipdef` [04:229](#), [02:45](#), [02:53](#), [02:92](#)
- `\skipeval` [77](#)
- `\skipeval` [05:158](#), [05:175](#), [77](#)
- `\skipzero` [04:229](#)
- `\slash` [06:890](#), [06:911](#), [02:507](#)
- `\sldefault` [25:3211](#), [29:25](#), [30:106](#)
- `\SLiTeX` [1303](#)
- `\sloppy` ... [40:397](#), [40:416](#), [49:130](#), [49:135](#)
- `sloppypar (env.)` [49:135](#)
- `\sloppypar` [49:135](#)
- `\slshape` [21:460](#), [21:822](#), [25:3209](#),
[25:3210](#), [29:23](#), [29:24](#), [32:22](#), [33:591](#)
- `\small` [204](#)
- `\smallbreak` [06:891](#), [06:912](#), [02:515](#)
- `\smallint` [30:368](#)
- `\smallskip` [18:447](#), [02:516](#)
- `\smallskipamount` .. [18:447](#), [18:450](#), [02:515](#)

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpara.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- \smash 17:29, 17:51, 30:484, 30:558,
30:559, 30:562, 30:563, 38:126, 847
- \smile 30:461
- socket commands:
 - \socket_assign_plug:nn 10:42,
10:44, 10:122, 10:122, 10:186, 341
 - \socket_debug_off:
. 10:7, 10:14, 10:189, 343
 - \socket_debug_on: 10:7, 10:9, 10:188, 343
 - \socket_if_exist:n 10:53
 - \socket_if_exist:nTF
. 10:28, 10:53, 10:60, 10:79, 10:99,
10:123, 10:190, 10:191, 10:192, 342
 - \socket_if_exist_p:n 10:53
 - \socket_if_plug_assigned:nn . 10:138
 - \socket_if_plug_assigned:nnTF
. 10:138, 10:196, 10:197, 10:198, 342
 - \socket_if_plug_assigned_p:nn 10:138
 - \socket_if_plug_exist:nn 10:117
 - \socket_if_plug_exist:nnTF
. 10:81, 10:101, 10:117,
10:125, 10:193, 10:194, 10:195, 342
 - \socket_if_plug_exist_p:nn . . 10:117
 - \socket_log:n
. 10:58, 10:58, 10:77, 10:184, 342
 - \socket_new:nn 10:27, 10:27, 10:182, 341
 - \socket_new_plug:nnn
. 10:38, 10:41, 10:78, 10:78, 10:185, 341
 - \socket_set_plug:nnn
. 10:78, 10:98, 10:175, 341
 - \socket_show:n 10:58, 10:77, 10:183, 342
 - \socket_use:n 10:144, 10:150, 342
 - \socket_use:nn . . . 10:144, 10:151, 342
 - \socket_use:nnn . . 10:144, 10:152, 342
 - \socket_use:nnnn . . 10:144, 10:153, 342
 - \socket_use:nw
. 10:144, 10:144, 10:150,
10:151, 10:152, 10:153, 10:187, 342
 - \socket_use_expandable:n
. 10:154, 10:157, 342
 - \socket_use_expandable:nw
. 10:154, 10:154, 10:157, 342
- socket internal commands:
 - __socket_debug:n 10:7, 10:7, 10:21, 343
 - \g__socket_debug_bool
. 10:6, 10:11, 10:16, 10:22, 10:24
 - __socket_debug_gset:
. 10:7, 10:12, 10:17, 10:19
 - __socket_debug_term:n
. 10:7, 10:8, 10:23, 10:45,
10:92, 10:108, 10:127, 10:145, 343
- Sockets:
 - build/column/footnotes 54:681
 - build/column/outputbox 54:631
 - refstepcounter 35:118
 - refstepcounter/target 35:119
 - tagssupport/build/column/footins
. 55:110
 - tagssupport/build/column/outputbox
. 55:109
 - tagssupport/build/page/footer 55:103
 - tagssupport/build/page/header 55:103
 - tagssupport/caption/begin 55:98
 - tagssupport/caption/end 55:98
 - tagssupport/caption/label/begin 55:101
 - tagssupport/caption/label/end 55:101
 - tagssupport/float/begin 55:96
 - tagssupport/float/end 55:96
 - tagssupport/float/hmode/begin . 55:94
 - tagssupport/float/hmode/end . . 55:94
 - tagssupport/marginpar/begin . . . 55:70
 - tagssupport/marginpar/end 55:70
 - tagssupport/math/luamml/annotate/false
. 55:130
 - tagssupport/math/luamml/array/finalize
. 55:134
 - tagssupport/math/luamml/array/finalizecol
. 55:136
 - tagssupport/math/luamml/array/initcol
. 55:135
 - tagssupport/math/luamml/array/save
. 55:133
 - tagssupport/math/luamml/artifact
. 55:158
 - tagssupport/math/luamml/finph@nt
. 55:152, 55:155
 - tagssupport/math/luamml/hbox . 55:149
 - tagssupport/math/luamml/mtable/aligncol
. 55:142
 - tagssupport/math/luamml/mtable/finalize
. 55:140
 - tagssupport/math/luamml/mtable/finalizecol
. 55:138
 - tagssupport/math/luamml/mtable/innertable/finalize
. 55:146
 - tagssupport/math/luamml/mtable/innertable/save
. 55:144
 - tagssupport/math/luamml/mtable/smallmatrix/save
. 55:145
 - tagssupport/math/luamml/mtable/tag/save
. 55:147
 - tagssupport/math/luamml/mtable/tag/set
. 55:148
 - tagssupport/math/luamml/save/nn 55:126
 - tagssupport/math/luamml/save/nNn
. 55:128
 - tagssupport/page@sofar 55:111
 - tagssupport/para/begin 55:46

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

tagsupport/para/end	55:46	\sqcap	30:387
tagsupport/para/restore	55:36	\sqcup	30:388
tagsupport/refstepcounter	55:46	\sqrt	38:355, 1312
tagsupport/tbl/cell/begin	55:72	\sqrtsign	30:540, 38:71, 38:355, 1312
tagsupport/tbl/cell/end	55:72	\sqsubset	29:730
tagsupport/tbl/colspan	55:82	\sqsubseteq	30:410
tagsupport/tbl/finalize	55:80	\sqsupset	29:731
tagsupport/tbl/hmode/begin	55:84	\sqsupseteq	30:411
tagsupport/tbl/hmode/end	55:84	\SS	21:324, 21:555, 21:1179, 57:656, 1317
tagsupport/tbl/init	55:78	\ss	21:274,
tagsupport/tbl/init/celldata	55:79		21:426, 21:578, 21:806, 21:1154, 57:656
tagsupport/tbl/leaders/begin	55:92	\sscdefault	25:2924, 25:2940, 25:3226
tagsupport/tbl/leaders/end	55:92	\sscshape	25:2924,
tagsupport/tbl/longtable/finalize	55:88		25:2939, 25:3224, 25:3225, 32:31, 626
tagsupport/tbl/longtable/foot	55:90	\stackrel	38:353
tagsupport/tbl/longtable/head	55:90	\stamp	1291
tagsupport/tbl/longtable/init	55:88	\star	30:408
tagsupport/tbl/pcell/end	55:72, 55:72	\stepcounter	521
tagsupport/tbl/restore/celldata	55:81	\stepcounter	22:19,
tagsupport/tbl/row/begin	55:72		22:29, 24:698, 28:49, 35:125, 35:138,
tagsupport/tbl/row/end	55:72		35:148, 35:180, 35:192, 35:203,
tagsupport/tbl/vmode/begin	55:84		38:363, 38:423, 38:517, 45:452,
tagsupport/tbl/vmode/end	55:84		45:520, 54:866, 54:924, 54:983, 1284
tagsupport/toc/contentsline/after	55:60	\stockheight	54:91
tagsupport/toc/contentsline/before	55:60	\stockwidth	54:91
tagsupport/toc/leaders/after	55:68	\stop	33:1600, 37:240
tagsupport/toc/leaders/before	55:68	\storedpar	04:156, 04:161
tagsupport/toc/starttoc/after	55:64	str commands:	
tagsupport/toc/starttoc/before	55:64	\c_backslash_str	
\sourceLaTeXdate	08:2935, 03:164, 50:65, 50:103		09:164, 09:415, 07:1624, 07:1631,
\sp	38:212		07:1647, 07:1651, 07:1699, 07:2809
\space	02:447, 1302	\c_hash_str	09:498,
space:floats:footnotes (plug)	54:656		09:502, 09:526, 07:1418, 07:1544
space:footnotes:floats (plug)	54:632	\c_right_brace_str	08:2057
\spacefactor	17:28, 17:30,	\str_case:nn	09:612, 11:517, 57:562, 07:1153
	17:50, 17:52, 18:130, 18:139, 18:158,	\str_case:nnTF	09:57, 11:455, 36:29, 51:30, 07:3056
	18:172, 18:184, 18:198, 18:212,	\str_case_e:nnTF	07:2730
	18:463, 18:504, 18:509, 21:88, 21:91,	\str_count:n	08:1854,
	45:531, 45:533, 02:505, 02:506, 1324		09:178, 09:253, 08:37, 08:43, 08:1250
\spacefactor _{in_{math}mode_{gh}} /643	1347	\str_gset:Nn	08:2798
\spaceskip	33:6, 33:576, 02:421	\str_head:n	07:2675
\spadesuit	30:345	\str_if_empty:NTF	57:588
\span	41:383	\str_if_eq:nn	268
\special	17:26, 17:48, 1139	\str_if_eq:NNTF	52:555
\SplitArgument	07:3268	\str_if_eq:nnTF	08:1802, 08:1910,
\splitfirstmark	54:2838		08:2010, 08:2117, 08:2329, 08:2533,
\SplitList	07:3268, 198		08:2587, 08:2657, 09:232, 09:304,
\splitmaxdepth	45:469, 45:488, 45:506, 54:2832, 02:394		09:386, 09:537, 09:544, 09:573,
\splittopskip	45:468, 45:487, 45:505, 02:419		09:576, 10:140, 11:79, 11:92, 11:243,
			11:264, 11:469, 11:484, 11:500,
			11:572, 11:729, 11:731, 11:905,
			05:137, 52:131, 52:549, 52:559,

File Key: 01=lt_{dirchk}.dtx, 02=lt_{plain}.dtx, 03=lt_{vers}.dtx, 04=lt_{luatex}.dtx, 05=lt_{expl}.dtx, 06=lt_{defns}.dtx, 07=lt_{cmd}.dtx, 08=lt_{hooks}.dtx, 09=lt_{cmdhooks}.dtx, 10=lt_{sockets}.dtx, 11=lt_{templates}.dtx, 12=lt_{alloc}.dtx, 13=lt_{cntrl}.dtx, 14=lt_{error}.dtx, 15=lt_{para}.dtx, 16=lt_{para}.dtx, 17=lt_{meta}.dtx, 18=lt_{space}.dtx, 19=lt_{logos}.dtx, 20=lt_{files}.dtx, 21=lt_{outenc}.dtx, 22=lt_{counts}.dtx, 23=lt_{length}.dtx, 24=lt_{tfssbas}.dtx, 25=lt_{tfssaxes}.dtx, 26=lt_{tfstrc}.dtx, 27=lt_{tfsscmp}.dtx, 28=lt_{tfssdcl}.dtx, 29=lt_{tfssini}.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=lt_{fntcmd}.dtx, 33=lt_{textcomp}.dtx, 34=lt_{pageno}.dtx, 35=lt_{xref}.dtx, 36=lt_{properties}.dtx, 37=lt_{miscen}.dtx, 38=lt_{math}.dtx, 39=lt_{lists}.dtx, 40=lt_{boxes}.dtx, 41=lt_{tab}.dtx, 42=lt_{pictur}.dtx, 43=lt_{thm}.dtx, 44=lt_{sect}.dtx, 45=lt_{float}.dtx, 46=lt_{idxglo}.dtx, 47=lt_{bibl}.dtx, 48=lt_{marks}.dtx, 49=lt_{page}.dtx, 50=lt_{class}.dtx, 51=lt_{keys}.dtx, 52=lt_{filehook}.dtx, 53=lt_{shipout}.dtx, 54=lt_{output}.dtx, 55=lt_{tagging}.dtx, 56=lt_{thyphen}.dtx, 57=lt_{final}.dtx

- 07:103, 07:105, 07:428, 07:811,
07:1232, 07:1237, 07:1720, 07:1774,
07:1777, 07:1780, 07:1783, 07:1842,
07:2261, 07:2274, 07:2276, 07:2306,
07:2326, 07:2328, 07:2347, 07:2374,
07:2402, 07:2404, 07:2429, 07:2431,
07:2454, 07:2571, 07:2583, 07:2611,
07:2787, 08:346, 08:438, 08:479,
08:536, 08:538, 08:609, 08:611,
08:741, 08:799, 08:832, 08:885,
08:967, 08:974, 08:1002, 08:1009, *174*
- `\str_if_eq_p:nn`
... 07:488, 07:489, 07:490, 07:491,
07:767, 07:958, 07:963, 07:2619,
07:2641, 08:889, 08:1593, 08:1679
- `\str_if_exist:NTF` 10:54
- `\str_if_in:nnTF` 11:383
- `\str_lowercase:n` 57:600
- `\str_map_function:NN` 05:144
- `\str_new:N` 10:35, 07:21
- `\str_replace_all:Nnn` 09:525
- `\str_set:Nn` 10:130,
52:399, 52:400, 07:231, 07:245,
07:246, 07:290, 57:578, 57:586, 57:590
- `\str_tail:n`
..... 07:2261, 07:2672, 07:2698, *173*
- `\str_use:N` 10:67,
10:70, 10:128, 10:147, 10:148, 10:155
- str internal commands:
- `__str_if_eq:nn` 08:23, *228*
- `\strcmp` 50:346, 50:362
- `\stretch` *18:532*
- `\string` *1110*
- `\stripmeaning` *1288*
- `\strut` 38:191, 38:192,
41:29, 06:892, 06:913, 02:523, 227
- `\strutbox` 26:190, 40:490, 40:507, 40:524,
41:186, 41:187, 45:469, 45:477,
45:488, 45:495, 45:506, 45:513, 02:523
- `\subencodingresult` .. 33:1200, 33:1205,
33:1374, 33:1378, 33:1380, 33:1381
- `\subparagraphmark` *44:143*
- `\subsectionmark` *44:143, 1009*
- `\subset` 30:437
- `\supseteq` 30:439
- `\substdefault` *1337*
- `\subsubsectionmark` *44:143*
- `\succ` 30:431
- `\succeq` 30:434
- `\sum` 30:361
- `\sup` 38:24
- `\supereject` *1315*
- `\suppressfloats` *54:2506*
- `\supset` 30:436
- `\supseteq` 30:438
- `\surd` 30:347
- `\SuspendTagging` *1243*
- `\SuspendTagging` 23:22, 55:3, *1243*
- `\swarrow` 30:419
- `\swdefault` *25:2919, 25:2938, 25:3223*
- `\swshape` *25:2919,*
25:2937, 25:3221, 25:3222, 32:30
- `\symbol` 21:180, *29:620*
- `\symletters` ... 33:8, 33:14, 33:578, 33:1100
- `\symoperators` 30:654
- sys commands:
- `\sys_if_engine luatex:TF` 53:26
- ### T
- `\T` 14:23, 21:355,
21:357, 21:359, 21:361, 21:363,
21:365, 21:367, 21:369, 21:371,
21:394, 50:1572, 50:1576, 50:1577
- `\t` 21:302, 21:763, 21:871, 33:121,
33:122, 33:149, 33:153, 33:155,
33:167, 33:170, 33:172, 33:652,
33:820, 33:1087, 33:1089, 33:1282, *1305*
- `tabbing (env.)` *41:71*
- `\tabbing` *41:71*
- `\tabbingsep` 41:130, 41:132, *41:166*
- `\tabcolsep` 41:259, *41:338*
- `\tableofcontents` *825*
- `\tabskip` 38:208, 38:209,
38:369, 38:372, 38:375, 38:377,
38:522, 38:535, 38:538, 38:540,
41:167, 41:192, 02:420, 02:534, 1285
- `tabular (env.)` *41:174*
- `\tabular` *41:174*
- `\tabular*` *41:175*
- `\tabularnewline` ... 41:194, *41:207, 1309*
- tag commands:
- `\tag_get:n` 55:55
- `\tag_if_active:` 55:7
- `\tag_if_active:TF` 55:196, *1357*
- `\tag_resume:n` 55:3, 55:4, 55:6
- `\tag_socket_use:n` ... 55:7, 55:10, *1244*
- `\tag_socket_use:nn` .. 55:7, 55:11, *1244*
- `\tag_socket_use:nnn` . 55:7, 55:12, *1244*
- `\tag_socket_use_expandable:n` ...
..... 55:7, 55:9, *1244*
- `\tag_suspend:n` 55:3, 55:3, 55:5
- tag internal commands:
- `\l__tag_block_flattened_level_-`
`int` *55:35, 55:42*
- `\l__tag_para_bool` 55:43
- `\l__tag_para_flattened_bool` .. 55:41
- `\l__tag_para_main_tag_tl` 55:39
- `\l__tag_para_tag_default_tl` .. 55:40

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpara.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

<code>\l__tag_para_tag_tl</code>	55:40	<code>tagsupport/math/luamml/mtable/innertable/save</code>	
<code>\g__tag_struct_dest_num_prop</code> .	55:53	(socket)	55:144
<code>\tag_resume:n</code>	1243	<code>tagsupport/math/luamml/mtable/smallmatrix/save</code>	
<code>\tag_socket_use:n</code>	1243	(socket)	55:145
<code>\tag_socket_use:nn</code>	1243	<code>tagsupport/math/luamml/mtable/tag/save</code>	
<code>\tag_socket_use:nnn</code>	1243	(socket)	55:147
<code>\tag_socket_use_expandable:n</code>	1243	<code>tagsupport/math/luamml/mtable/tag/set</code>	
<code>\tag_suspend:n</code>	1243	(socket)	55:148
<code>tagsupport/build/column/footins</code>		<code>tagsupport/math/luamml/save/nn</code>	
(socket)	55:110	(socket)	55:126
<code>tagsupport/build/column/outputbox</code>		<code>tagsupport/math/luamml/save/nNn</code>	
(socket)	55:109	(socket)	55:128
<code>tagsupport/build/page/footer</code> (socket)		<code>tagsupport/page@sofar</code> (socket) . . .	55:111
.	55:103	<code>tagsupport/para/begin</code> (socket)	55:46
<code>tagsupport/build/page/header</code> (socket)		<code>tagsupport/para/end</code> (socket)	55:46
.	55:103	<code>tagsupport/para/restore</code> (socket) . .	55:36
<code>tagsupport/caption/begin</code> (socket) .	55:98	<code>tagsupport/refstepcounter</code> (socket)	55:46
<code>tagsupport/caption/end</code> (socket) . . .	55:98	<code>tagsupport/tbl/cell/begin</code> (socket)	55:72
<code>tagsupport/caption/label/begin</code>		<code>tagsupport/tbl/cell/end</code> (socket) . .	55:72
(socket)	55:101	<code>tagsupport/tbl/colspan</code> (socket) . . .	55:82
<code>tagsupport/caption/label/end</code> (socket)		<code>tagsupport/tbl/finalize</code> (socket) . .	55:80
.	55:101	<code>tagsupport/tbl/hmode/begin</code> (socket)	55:84
<code>tagsupport/float/begin</code> (socket) . . .	55:96	<code>tagsupport/tbl/hmode/end</code> (socket) .	55:84
<code>tagsupport/float/end</code> (socket)	55:96	<code>tagsupport/tbl/init</code> (socket)	55:78
<code>tagsupport/float/hmode/begin</code> (socket)		<code>tagsupport/tbl/init/celldata</code> (socket)	
.	55:94	55:79
<code>tagsupport/float/hmode/end</code> (socket)	55:94	<code>tagsupport/tbl/leaders/begin</code> (socket)	
<code>tagsupport/marginpar/begin</code> (socket)	55:70	55:92
<code>tagsupport/marginpar/end</code> (socket) .	55:70	<code>tagsupport/tbl/leaders/end</code> (socket)	55:92
<code>tagsupport/math/luamml/annotate/false</code>		<code>tagsupport/tbl/longtable/finalize</code>	
(socket)	55:130	(socket)	55:88
<code>tagsupport/math/luamml/array/finalize</code>		<code>tagsupport/tbl/longtable/foot</code> (socket)	
(socket)	55:134	55:90
<code>tagsupport/math/luamml/array/finalizecol</code>		<code>tagsupport/tbl/longtable/head</code> (socket)	
(socket)	55:136	55:90
<code>tagsupport/math/luamml/array/initcol</code>		<code>tagsupport/tbl/longtable/init</code> (socket)	
(socket)	55:135	55:88
<code>tagsupport/math/luamml/array/save</code>		<code>tagsupport/tbl/pcell/end</code> (socket) . . .	
(socket)	55:133	55:72 , 55:72
<code>tagsupport/math/luamml/artifact</code>		<code>tagsupport/tbl/restore/celldata</code>	
(socket)	55:158	(socket)	55:81
<code>tagsupport/math/luamml/finph@nt</code>		<code>tagsupport/tbl/row/begin</code> (socket) .	55:72
(socket)	55:152 , 55:155	<code>tagsupport/tbl/row/end</code> (socket) . . .	55:72
<code>tagsupport/math/luamml/hbox</code> (socket) .		<code>tagsupport/tbl/vmode/begin</code> (socket)	55:84
.	55:149	<code>tagsupport/tbl/vmode/end</code> (socket) .	55:84
<code>tagsupport/math/luamml/mtable/aligncol</code>		<code>tagsupport/toc/contentsline/after</code>	
(socket)	55:142	(socket)	55:60
<code>tagsupport/math/luamml/mtable/finalize</code>		<code>tagsupport/toc/contentsline/before</code>	
(socket)	55:140	(socket)	55:60
<code>tagsupport/math/luamml/mtable/finalizecol</code>		<code>tagsupport/toc/leaders/after</code> (socket)	
(socket)	55:138	55:68
<code>tagsupport/math/luamml/mtable/innertable/finph@nt</code>		<code>tagsupport/toc/leaders/before</code> (socket)	
(socket)	55:146	55:68

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=ltemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=ltthyphen.dtx, 57=ltfinal.dtx

- tagsupport/toc/starttoc/after (socket)
 55:64
- tagsupport/toc/starttoc/before
 (socket) 55:64
- \tan 38:15
- \tanh 38:17
- target 36:239, 36:240, 810
- \tau 30:296
- tbl commands:
- \tbl_count_missing_cells:n 55:195, 55:195, 55:296, 1256
 - \tbl_count_table_cols: 55:189, 55:189, 1253
 - \tbl_crcr:n 55:293, 55:293
 - \tbl_gdecr_row_count: 55:253, 55:259
 - \tbl_gincr_row_count: 55:253, 55:256
 - \tbl_gzero_row_count: 55:253, 55:253
 - \tbl_if_row_was_started:TF 55:247, 55:247, 55:250
 - \tbl_inbetween_rows: . 55:262, 55:262
 - \tbl_init_cell_data_for_row: ... 55:243, 55:243
 - \tbl_init_cell_data_for_table . 1247
 - \tbl_init_cell_data_for_table: .. 55:219, 55:219
 - \tbl_restore_outer_cell_data: ... 55:265, 55:265, 1247
 - \tbl_save_outer_table_cols: 55:216, 55:216
 - \tbl_update_cell_data: 55:185, 55:185
 - \tbl_update_cell_data_for_next_-
 row: 55:239, 55:239
 - \tbl_update_multicolumn_cell_-
 data:n 55:282, 55:282
- tbl internal commands:
- \g__tbl_col_int 55:161,
 55:186, 55:197, 55:202, 55:220,
 55:236, 55:241, 55:244, 55:248,
 55:251, 55:263, 55:266, 55:273,
 55:283, 55:286, 55:289, 55:294, 1255
 - \g__tbl_missing_cells_int 55:177, 55:199, 55:206, 55:210
 - \g__tbl_missingcells_int 55:177
 - \g__tbl_row_int 55:161,
 55:221, 55:235, 55:240, 55:254,
 55:257, 55:260, 55:267, 55:272, 1251
 - \l__tbl_saved_col_tl 55:168, 55:220, 55:226, 55:266
 - \l__tbl_saved_row_tl 55:168, 55:221, 55:225, 55:267
 - \l__tbl_saved_span_tl 55:168, 55:222, 55:227, 55:268, 55:274
 - \l__tbl_saved_table_cols_tl 55:168, 55:217, 55:229, 55:231, 55:269
 - \g__tbl_span_tl .. 55:161, 55:186,
 55:187, 55:203, 55:222, 55:237,
 55:245, 55:268, 55:289, 55:291, 1256
 - \g__tbl_table_cols_tl 55:161, 55:191, 55:192, 55:201,
 55:217, 55:269, 55:276, 55:278, 1253
 - \l__tbl_tmpa_seq 55:190, 55:191, 55:194
 - __tbl_trace:n ... 55:179, 55:182,
 55:184, 55:192, 55:207, 55:224, 55:271
- template internal commands:
- __template_assign_boolean: 11:766, 11:766
 - __template_assign_boolean_aux:n 11:766, 11:769, 11:770, 11:772
 - __template_assign_choice: 11:792, 11:792
 - __template_assign_choice_-
 aux:nTF 11:792, 11:794, 11:797, 11:809, 11:814
 - __template_assign_function: ... 11:815, 11:815
 - __template_assign_function_-
 aux:N . 11:815, 11:818, 11:819, 11:821
 - __template_assign_instance: ... 11:832, 11:832
 - __template_assign_instance_-
 aux:N . 11:832, 11:835, 11:836, 11:838
 - __template_assign_variable: ... 11:736, 11:850, 11:850
 - __template_assign_variable:n ... 11:850, 11:852, 11:860
 - __template_assignments_pop: ... 11:915, 11:915, 11:1200
 - __template_assignments_push:n .. 11:674, 11:916, 11:916
 - \l__template_assignments_tl 11:19, 11:675, 11:713, 11:777,
 11:785, 11:812, 11:823, 11:840,
 11:865, 11:872, 11:915, 11:917, 360
 - \c__template_code_root_tl 11:9, 11:54,
 11:402, 11:405, 11:647, 11:649,
 11:650, 11:676, 11:763, 11:919, 360
 - __template_convert_to_assignments: 11:669, 11:711, 11:711, 11:762
 - __template_convert_to_assignments_-
 aux:n 11:711, 11:715, 11:717
 - __template_convert_to_assignments_-
 aux:nn 11:711, 11:720, 11:722, 11:741
 - __template_declare_instance:nxxx 11:652, 11:652, 11:1176
 - __template_declare_instance_-
 aux:nxxx 11:652, 11:658, 11:661, 11:708

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefs.dtx, 07=ltcmd.dtx, 08=lt hooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=lt space.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- `__template_declare_template_-code:nmmn` [11:374](#), [11:386](#), [11:388](#), [11:393](#), [11:400](#)
- `__template_declare_template_-code:nmmn` . [11:374](#), [11:374](#), [11:1168](#)
- `__template_declare_template_-keys:nmmn` .. [11:203](#), [11:203](#), [11:1166](#)
- `__template_declare_type:nn` [11:182](#), [11:186](#), [11:188](#)
- `\l__template_default_tl` .. [11:20](#), [360](#)
- `\c__template_defaults_root_tl` [11:10](#), [11:99](#), [11:100](#), [11:139](#), [11:142](#), [360](#)
- `__template_define_type:nn` [11:182](#), [11:182](#), [11:1164](#)
- `__template_edit_defaults:nnn` ... [11:605](#), [11:605](#), [11:1172](#)
- `__template_edit_instance:nnn` ... [11:693](#), [11:693](#), [11:1180](#)
- `__template_edit_instance_-aux:nmmn` ... [11:700](#), [11:705](#), [11:710](#)
- `__template_edit_instance_-aux:nmmn` [11:693](#)
- `\l__template_error_bool` ... [11:21](#), [11:222](#), [11:228](#), [11:248](#), [11:277](#), [11:290](#), [11:622](#), [11:663](#), [11:665](#), [360](#)
- `__template_execute_if_arg_-agree:nnTF` [11:42](#), [11:42](#), [11:207](#), [11:378](#)
- `__template_execute_if_code_-exist:nnTF` [11:52](#), [11:52](#), [11:654](#), [11:757](#), [11:940](#)
- `__template_execute_if_keys_-exist:nnTF` [11:71](#)
- `__template_execute_if_keytype_-exist:nTF` [11:58](#), [11:58](#), [11:64](#), [11:224](#)
- `__template_execute_if_type_-exist:nTF` [11:65](#), [11:65](#), [11:205](#), [11:376](#)
- `__template_find_global:` [11:732](#), [11:742](#), [11:742](#)
- `__template_find_global_aux:w` ... [11:742](#), [11:747](#), [11:750](#)
- `\l__template_global_bool` [11:22](#), [11:744](#), [11:753](#), [11:768](#), [11:817](#), [11:834](#), [11:856](#), [360](#)
- `__template_if_instance_exist:nn` [11:84](#)
- `__template_if_instance_exist:nnTF` ... [11:84](#), [11:682](#), [11:695](#), [11:911](#), [11:957](#), [11:1194](#), [11:1196](#), [11:1198](#)
- `__template_if_key_value:n` [11:77](#), [11:83](#)
- `__template_if_key_value:nTF` ... [11:77](#), [11:774](#), [11:862](#)
- `__template_if_keys_exist:nnTF` .. [11:71](#), [11:380](#), [11:607](#), [11:922](#), [11:931](#)
- `__template_if_use_template:n` . [11:90](#)
- `__template_if_use_template:nTF` . [11:90](#), [11:899](#)
- `__template_implement_choice_-elt:n` [11:535](#), [11:568](#), [11:600](#)
- `__template_implement_choice_-elt:nnn` [11:536](#), [11:568](#), [11:568](#)
- `__template_implement_choice_-elt_aux:n` [11:568](#), [11:574](#), [11:582](#), [11:585](#)
- `__template_implement_choice_-elt_aux:nnn` [11:568](#), [11:573](#), [11:580](#), [11:592](#)
- `__template_implement_choices:nn` [11:457](#), [11:529](#), [11:529](#)
- `__template_implement_choices_-default:` ... [11:529](#), [11:540](#), [11:547](#)
- `__template_instance_set_eq:nnn` . [11:680](#), [11:680](#), [11:1178](#)
- `\c__template_instances_root_tl` .. [11:11](#), [11:86](#), [11:670](#), [11:672](#), [11:686](#), [11:688](#), [11:689](#), [11:912](#), [360](#)
- `\l__template_key_name_tl` .. [11:23](#), [11:231](#), [11:234](#), [11:261](#), [11:263](#), [11:284](#), [11:299](#), [11:306](#), [11:311](#), [11:353](#), [11:355](#), [11:360](#), [11:423](#), [11:426](#), [11:431](#), [11:475](#), [11:489](#), [11:504](#), [11:513](#), [11:532](#), [11:533](#), [11:538](#), [11:550](#), [11:554](#), [11:557](#), [11:560](#), [11:563](#), [11:564](#), [11:587](#), [11:590](#), [11:595](#), [11:597](#), [11:603](#), [11:627](#), [11:629](#), [11:632](#), [11:734](#), [11:795](#), [11:798](#), [11:800](#), [11:804](#), [376](#)
- `\c__template_key_order_root_tl` .. [11:13](#), [11:115](#), [11:116](#), [11:155](#), [11:158](#), [360](#)
- `\l__template_key_order_seq` [11:29](#), [11:117](#), [11:157](#), [11:160](#), [11:211](#), [11:230](#), [11:263](#), [11:714](#), [361](#)
- `__template_key_to_value:` [11:776](#), [11:864](#), [11:879](#), [11:879](#)
- `__template_key_to_value_auxi:w` . [11:879](#), [11:880](#), [11:881](#)
- `__template_key_to_value_auxii:w` [11:879](#), [11:887](#), [11:892](#)
- `\l__template_keytype_arg_tl` [11:25](#), [11:245](#), [11:258](#), [11:259](#), [11:266](#), [11:323](#), [11:337](#), [11:470](#), [11:531](#), [11:828](#), [11:845](#), [361](#)
- `\l__template_keytype_tl` [11:24](#), [11:224](#), [11:243](#), [11:257](#), [11:264](#), [11:273](#), [11:322](#), [11:333](#),

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lt hooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=lt space.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscomp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- 11:427, 11:429, 11:455, 11:517,
11:637, 11:729, 11:731, 11:735, 361
- \c__template_keytypes_arg_seq ...
..... 11:16, 11:226, 11:346, 360
- \l__template_keytypes_prop
..... 11:28, 11:114, 11:151,
11:154, 11:210, 11:261, 11:416,
11:425, 11:431, 11:557, 11:587,
11:629, 11:719, 11:800, 11:934, 361
- \c__template_keytypes_root_tl ...
..... 11:12, 11:73, 11:107, 11:110,
11:112, 11:113, 11:149, 11:152, 360
- __template_map_var_type:
11:496, 11:499, 11:515, 11:515, 11:854
- __template_parse_keys_elt:n ...
.. 11:213, 11:219, 11:219, 11:272, 367
- __template_parse_keys_elt:nn ...
..... 11:213, 11:270, 11:270
- __template_parse_keys_elt_aux: .
..... 11:219, 11:236, 11:253
- __template_parse_keys_elt_aux:n
..... 11:219, 11:227, 11:241
- __template_parse_values:nnn ...
11:610, 11:614, 11:614, 11:664, 11:761
- __template_parse_values_elt:n ..
..... 11:618, 11:620, 11:620
- __template_parse_values_elt:nn .
..... 11:618, 11:625, 11:625
- __template_parse_values_elt_
aux:n 11:625, 11:631, 11:634
- __template_parse_vars_elt:n ...
..... 11:414, 11:419, 11:419
- __template_parse_vars_elt:nnn ..
..... 11:414, 11:421, 11:421
- __template_parse_vars_elt_
aux:nn 11:421, 11:430, 11:435
- __template_parse_vars_elt_
aux:nnn
11:421, 11:443, 11:447, 11:453, 11:509
- __template_parse_vars_elt_
aux:nw 11:421, 11:437, 11:439
- __template_parse_vars_elt_
key:nn
11:421, 11:462, 11:479, 11:497, 11:510
- __template_quark_if_nil:N ... 11:41
- __template_quark_if_nil:NTF 11:894
- __template_quark_if_nil:nTF . 11:41
- __template_quark_if_nil_p:n . 11:41
- __template_recover_defaults:nn .
..... 11:136, 11:136, 11:410,
11:609, 11:641, 11:656, 11:759, 11:924
- __template_recover_keytypes:nn .
..... 11:136,
11:146, 11:411, 11:616, 11:643, 11:933
- __template_recover_values:nn ...
11:136, 11:162, 11:684, 11:697, 11:959
- __template_recover_vars:nn
..... 11:136, 11:172,
11:645, 11:657, 11:707, 11:760, 11:942
- \c__template_restrict_root_tl .. 360
- __template_show:Nnnn
11:920, 11:925, 11:934, 11:943, 11:947
- __template_show_code:nn
..... 11:918, 11:918, 11:1184
- __template_show_defaults:nn ...
..... 11:920, 11:920, 11:1186
- __template_show_keytypes:nn ...
..... 11:920, 11:929, 11:1188
- __template_show_values:nn
..... 11:955, 11:955, 11:1192
- __template_show_vars:nn
..... 11:920, 11:938, 11:1190
- __template_split_keytype:n
..... 11:221, 11:275, 11:275
- __template_split_keytype_arg:n .
..... 11:316,
11:320, 11:320, 11:349, 11:429,
11:559, 11:589, 11:636, 11:728, 11:802
- __template_split_keytype_arg_
aux:n . 11:320, 11:324, 11:347, 11:350
- __template_split_keytype_arg_
aux:w . 11:320, 11:328, 11:343, 11:351
- __template_split_keytype_aux:w .
..... 11:275, 11:285, 11:296, 11:308
- __template_store_defaults:nn ...
.. 11:96, 11:96, 11:214, 11:611, 11:642
- __template_store_key_implementation:nnn
..... 11:382, 11:408, 11:408
- __template_store_keytypes:nn ...
..... 11:96, 11:104, 11:215, 11:644
- __template_store_value:n
11:354, 11:354, 11:356, 11:357, 11:358
- __template_store_value_aux:Nn ..
..... 11:359, 11:359, 11:362,
11:364, 11:366, 11:368, 11:370, 11:372
- __template_store_value_boolean:n
..... 11:352, 11:352
- __template_store_value_choice:n
..... 11:354, 11:356
- __template_store_value_commalist:n
..... 11:359, 11:373
- __template_store_value_function:n
..... 11:354, 11:357
- __template_store_value_instance:n
..... 11:354, 11:358
- __template_store_value_integer:n
..... 11:359, 11:361

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lt hooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=lt space.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

<code>__template_store_value_length:n</code>	11:360, 11:538, 11:560, 11:667, 11:698, 11:724, 11:925, 11:964, 370
<code>__template_store_value_muskip:n</code>	11:14, 11:123, 11:124, 11:165, 11:168, 360
<code>__template_store_value_real:n</code>	11:359, 11:367
<code>__template_store_value_skip:n</code>	11:359, 11:369
<code>__template_store_value_tokenlist:n</code>	11:359, 11:371, 11:373
<code>__template_store_values:nn</code>	11:96, 11:120, 11:668, 11:685
<code>__template_store_vars:nn</code>	11:96, 11:128, 11:415, 11:646
<code>__template_template_set_eq:nnn</code>	11:639, 11:639, 11:1170
<code>\l__template_tmp_clist</code>	11:32, 11:531, 11:541, 11:543, 11:570, 11:577, 11:579, 361
<code>\l__template_tmp_dim</code>	11:33, 361
<code>\l__template_tmp_int</code>	11:34, 11:190, 11:191, 11:194, 11:196, 11:200, 361
<code>\l__template_tmp_muskip</code>	11:35, 361
<code>\l__template_tmp_skip</code>	11:36, 361
<code>\l__template_tmp_tl</code>	11:37, 11:44, 11:45, 11:49, 11:255, 11:262, 11:278, 11:279, 11:280, 11:286, 11:539, 11:549, 11:550, 11:551, 11:553, 11:554, 11:555, 11:558, 11:559, 11:561, 11:588, 11:589, 11:596, 11:598, 11:630, 11:636, 11:699, 11:701, 11:719, 11:720, 11:801, 11:802, 11:811, 11:812, 11:883, 11:884, 11:890, 361
<code>\g__template_type_prop</code>	11:18, 11:44, 11:67, 11:184, 11:195, 360
<code>__template_use_instance:nn</code>	11:844, 11:897, 11:897, 11:1182
<code>__template_use_instance_aux:nn</code>	11:897, 11:901, 11:909
<code>__template_use_instance_aux:nNnnn</code>	11:897, 11:900, 11:903
<code>__template_use_template:nnn</code>	11:755, 11:755, 11:906, 11:1174
<code>\l__template_value_tl</code>	11:26, 11:724, 11:774, 11:781, 11:787, 11:795, 11:805, 11:829, 11:846, 11:862, 11:868, 11:875, 11:880, 11:885, 11:887, 11:895, 361
<code>\l__template_values_prop</code>	11:30, 11:101, 11:125, 11:141, 11:144, 11:167, 11:170, 11:209, 11:353, 11:355,
<code>\c__template_values_root_tl</code>	11:14, 11:123, 11:124, 11:165, 11:168, 360
<code>\l__template_var_tl</code>	11:27, 11:726, 11:745, 11:747, 11:752, 11:780, 11:788, 11:826, 11:842, 11:867, 11:874, 361
<code>\l__template_vars_prop</code>	11:31, 11:133, 11:177, 11:180, 11:412, 11:474, 11:488, 11:503, 11:532, 11:551, 11:555, 11:598, 11:726, 11:811, 11:884, 11:943, 361
<code>\c__template_vars_root_tl</code>	11:15, 11:131, 11:132, 11:175, 11:178, 360
<code>\tencirc</code>	31:10, 42:125, 42:677
<code>\tencircw</code>	31:10, 42:128
<code>\tenln</code>	31:9, 42:124, 42:126, 42:676, 42:678
<code>\tenlnw</code>	31:9, 42:127, 42:129
<code>\test</code>	1291
<code>\testallgroups</code>	33:1214, 33:1387, 33:1388, 33:1389, 33:1391, 33:1392, 33:1393, 33:1394, 33:1395, 33:1396, 33:1397, 33:1398, 33:1399, 33:1400, 33:1401, 33:1402, 33:1403, 33:1404, 33:1405, 33:1406, 33:1407, 33:1408, 33:1409, 33:1413, 33:1414, 33:1415, 33:1416, 33:1417, 33:1418, 33:1422, 33:1423, 33:1424, 33:1428, 33:1429, 33:1430, 33:1431, 33:1432, 33:1433, 33:1434, 33:1435, 33:1436, 33:1437, 33:1438, 33:1439, 33:1440, 33:1441, 33:1442, 33:1443, 33:1444, 33:1445, 33:1446, 33:1447, 33:1448, 33:1450, 33:1454, 33:1455, 33:1456, 33:1457, 33:1458, 33:1459, 33:1460, 33:1461, 33:1462, 33:1464, 33:1465, 33:1466, 33:1467, 33:1468, 33:1469, 33:1470, 33:1471, 33:1472, 33:1473, 33:1474, 33:1475, 33:1476, 33:1477, 33:1478, 33:1479, 33:1480, 33:1481, 33:1482, 33:1483, 33:1484, 33:1485, 33:1486, 33:1487, 33:1488, 33:1489, 33:1490, 33:1491, 33:1492, 33:1493, 33:1494, 33:1495, 33:1496, 33:1497, 33:1498, 33:1499, 33:1500, 33:1501, 33:1502, 33:1503, 33:1504, 33:1505, 33:1506, 33:1507, 33:1508, 33:1509, 33:1510, 33:1511, 33:1512, 33:1513, 33:1514, 33:1515, 33:1516, 33:1517, 33:1518, 33:1519, 33:1520, 33:1521, 33:1522, 33:1523, 33:1524, 33:1525, 33:1526, 33:1527, 33:1528, 33:1529, 33:1530, 33:1531, 33:1532,

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- 33:1533, 33:1534, 33:1535, 33:1536,
 33:1537, 33:1538, 33:1539, 33:1540,
 33:1541, 33:1542, 33:1543, 33:1544,
 33:1545, 33:1546, 33:1547, 33:1548,
 33:1549, 33:1550, 33:1551, 33:1552,
 33:1553, 33:1554, 33:1555, 33:1556,
 33:1557, 33:1558, 33:1559, 33:1560,
 33:1561, 33:1562, 33:1563, 33:1564,
 33:1565, 33:1566, 33:1567, 33:1568,
 33:1569, 33:1570, 33:1571, 33:1572,
 33:1573, 33:1574, 33:1575, 33:1576,
 33:1577, 33:1578, 33:1579, 33:1598
- `\testallkerneldefinedfamilies`
 33:1386, 33:1595
- `\testFont` 33:1182, 33:1183, 33:1215, 33:1216
- `\testgroup` . 33:1187, 33:1225, 33:1234,
 33:1240, 33:1246, 33:1250, 33:1254,
 33:1264, 33:1268, 33:1336, 33:1340
- `\TeX` 19:1, 19:12, 1336
- `\TeX` and `\LaTeX 2 ϵ` commands:
- `\...-h@k` 1336
- `\..@without@substitution` 638
- `\@` 09:392,
 14:19, 04:20, 18:500, 19:2, 04:1026,
 50:43, 50:58, 50:71, 50:80, 50:118,
 06:833, 06:834, 57:512, 01:49, 1329
- `\@...hook` 234
- `\@?@?` 1286
- `\@@` . 13:15, 13:19, 13:20, 13:21, 13:22,
 13:24, 13:27, 13:28, 13:30, 13:31,
 20:753, 20:773, 26:511, 26:513,
 26:514, 41:238, 41:239, 41:240,
 41:250, 01:315, 01:316, 54:6, 54:7
- `\@@DeclareMathDelimiter` 1297
- `\@@defaultsubs` 24:682
- `\@@enc@update`
 21:201, 24:322, 24:326, 1309
- `\@@end` 20:722, 20:723, 01:206,
 06:23, 37:33, 37:98, 37:174, 37:240,
 56:18, 57:673, 57:694, 01:53, 471
- `\@@endpbox` 41:193, 41:236, 41:386
- `\@@eqnocr`
 38:381, 38:403, 38:413, 38:418, 38:541
- `\@@fileswith@pti@ns`
 50:605, 50:624, 50:1104
- `\@@hyph` 06:24, 06:850
- `\@@hyphenation` 21:225
- `\@@if@newlist` 54:801,
 54:864, 54:877, 54:922, 54:935, 54:981
- `\@@if@definable` 21:35, 06:132
- `\@@input` 20:620,
 20:633, 20:673, 06:22, 37:27,
 37:92, 37:154, 50:1727, 52:172,
 52:188, 52:196, 57:331, 01:52, 1291
- `\@@italiccorr` 32:113, 32:117, 06:25, 1293
- `\@@line` 40:614
- `\@@math@bgroup` 32:131, 32:138
- `\@@math@egroup` 32:128
- `\@@par` . 15:4, 16:140, 16:177, 06:21,
 37:240, 37:531, 37:536, 37:539,
 37:553, 37:557, 37:560, 39:82, 39:85,
 40:334, 40:361, 40:385, 40:406,
 41:199, 44:67, 44:118, 54:255, 422
- `\@@patterns` 21:225
- `\@@protect` 06:266, 06:272, 06:281
- `\@@sqrt` 1315
- `\@@startpbox` . . . 41:193, 41:236, 41:386
- `\@@sverb` 37:646, 840
- `\@@text@case@aux`
 57:580, 57:614, 57:619, 57:624
- `\@@underline` . . . 40:551, 40:554, 40:555
- `\@@unprocessedoptions`
 50:1016, 50:1081, 50:1179
- `\@@warning` 14:215, 1289
- `\@Alph` 22:193, 22:209, 109
- `\@DeclareEncodingSubset` . . 24:191,
 24:193, 24:194, 24:195, 24:196, 24:199
- `\@DeclareMathDelimiter`
 28:983, 28:1002, 1297
- `\@DeclareMathSizes`
 24:269, 24:270, 24:272
- `\@Eshpack`
 . 18:191, 45:201, 45:223, 45:241, 1283
- `\@Ialph` 1299
- `\@IncludeInRelease` 03:71
- `\@IncludeInRelease` 03:71
- `\@LRmoderr` 14:271, 18:239, 18:290, 1356
- `\@M` 18:11, 18:12, 18:13, 18:14,
 18:15, 18:16, 18:17, 18:18, 18:19,
 18:120, 24:751, 24:759, 26:440,
 26:453, 06:37, 06:39, 38:391, 39:225,
 41:56, 44:67, 44:100, 44:118, 44:130,
 44:234, 44:257, 02:21, 54:174,
 54:193, 54:196, 54:256, 02:508, 02:509
- `\@MM` 45:469,
 45:488, 45:506, 02:21, 54:297, 391
- `\@Mi` 12:3, 54:134
- `\@Mii` 12:3, 45:53, 45:122,
 45:194, 45:216, 45:241, 45:311,
 54:293, 54:1434, 54:1583, 54:1745
- `\@Miii` 12:3, 45:55, 45:124, 45:313, 54:296
- `\@Miv` 12:3,
 45:195, 45:201, 45:217, 45:223, 54:270
- `\@Roman` 22:191, 22:197, 1322
- `\@TeXversion` 14:28, 01:310, 2
- `\@abspage@last` . . . 53:105, 53:111,
 53:351, 53:353, 53:355, 53:363,
 53:367, 53:370, 53:409, 53:410, 1145

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpara.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- \@acci ... [29:776](#), [40:387](#), [40:408](#), [1299](#)
- \@accii .. [29:776](#), [40:387](#), [40:408](#), [1299](#)
- \@acciii . [29:776](#), [40:387](#), [40:408](#), [1299](#)
- \@acol [41:168](#), [41:178](#),
[41:260](#), [41:261](#), [41:273](#), [41:274](#),
[41:277](#), [41:294](#), [41:309](#), [41:317](#), [41:327](#)
- \@acolampacol [41:258](#), [41:275](#),
[41:277](#), [41:284](#), [41:292](#), [41:326](#), [41:329](#)
- \@activechar@info [54:774](#), [1296](#)
- \@activechar@warning [1296](#)
- \@addamp [41:251](#), [41:260](#),
[41:261](#), [41:276](#), [41:290](#), [41:327](#), [41:328](#)
- \@addfield [41:43](#), [41:53](#),
[41:86](#), [41:93](#), [41:125](#), [41:140](#), [41:142](#)
- \@addmarginpar [54:329](#), [54:2311](#)
- \@addtobot
[54:1249](#), [54:1336](#), [54:1404](#), [54:1456](#),
[54:1553](#), [54:1602](#), [54:1709](#), [54:1768](#)
- \@addtocurcol [54:326](#), [54:1340](#), [54:2499](#)
- \@addtodblcol [54:1128](#), [54:2062](#)
- \@addtofilelist
... [20:64](#), [20:133](#), [20:188](#), [20:620](#),
[20:742](#), [29:750](#), [29:753](#), [29:760](#),
[29:763](#), [29:770](#), [29:773](#), [50:1726](#),
[52:169](#), [52:188](#), [52:196](#), [57:279](#),
[57:282](#), [57:714](#), [01:85](#), [01:87](#), [1293](#)
- \@addtonextcol [54:1127](#), [54:1805](#), [54:2500](#)
- \@addtopreamble .. [41:311](#), [41:324](#),
[41:330](#), [41:331](#), [41:332](#), [41:334](#), [41:346](#)
- \@addtoreset [22:18](#), [22:46](#),
[22:66](#), [22:79](#), [22:142](#), [22:168](#), [22:171](#)
- \@addtotoporbot
..... [54:1286](#), [54:1450](#), [54:1598](#),
[54:1762](#), [54:1855](#), [54:1936](#), [54:2024](#)
- \@afterheading ... [44:92](#), [44:125](#), [1320](#)
- \@afterindentfalse [44:45](#)
- \@afterindenttrue
..... [44:43](#), [44:124](#), [44:233](#), [44:256](#)
- \@alph [22:192](#), [22:205](#), [45:399](#), [109](#)
- \@ampacol [41:258](#), [41:275](#), [41:286](#), [41:329](#)
- \@arabic [22:54](#), [22:72](#),
[22:164](#), [22:176](#), [22:189](#), [22:195](#), [45:397](#)
- \@argarraycr [41:203](#), [41:204](#)
- \@argdef [06:80](#)
- \@argsbox [40:575](#)
- \@argtabularcr [41:210](#), [41:211](#)
- \@array [41:181](#), [41:182](#)
- \@arrayacol [41:168](#), [41:258](#)
- \@arrayclassiv [41:169](#), [41:331](#)
- \@arrayclassz [41:168](#), [41:275](#)
- \@arraycr [41:170](#), [41:201](#), [41:203](#)
- \@arrayparboxrestore
..... [40:377](#), [40:419](#), [41:384](#)
- \@arrayrule [41:309](#),
[41:311](#), [41:315](#), [41:317](#), [41:319](#), [41:346](#)
- \@arstrut [41:192](#), [41:237](#), [41:343](#)
- \@arstrutbox
... [41:185](#), [41:218](#), [41:343](#), [41:385](#), [1297](#)
- \@author [44:8](#), [44:32](#)
- \@auxout
... [20:217](#), [20:223](#), [20:267](#), [20:285](#),
[20:309](#), [20:342](#), [20:371](#), [20:394](#),
[20:418](#), [20:433](#), [35:105](#), [35:173](#),
[36:55](#), [44:189](#), [44:199](#), [47:7](#), [47:8](#),
[47:39](#), [47:49](#), [47:57](#), [47:67](#), [47:84](#), [53:362](#)
- \@backslashchar .. [14:234](#), [14:236](#),
[30:277](#), [06:231](#), [06:481](#), [06:630](#),
[06:642](#), [06:646](#), [06:647](#), [06:652](#),
[06:696](#), [50:1308](#), [50:1441](#), [50:1530](#), [1299](#)
- \@badcrerr [14:279](#), [1297](#)
- \@badend [14:247](#), [37:390](#), [1346](#)
- \@badlinearg [14:267](#), [42:165](#),
[42:177](#), [42:188](#), [42:189](#), [42:193](#),
[42:242](#), [42:247](#), [42:257](#), [42:262](#), [42:275](#)
- \@badmath [14:251](#), [38:275](#),
[38:277](#), [38:282](#), [38:285](#), [38:294](#),
[38:306](#), [38:311](#), [38:320](#), [38:333](#),
[38:338](#), [38:464](#), [38:476](#), [38:492](#), [38:501](#)
- \@badpoptabs [14:255](#), [41:85](#), [41:151](#)
- \@badrequireerror [50:497](#), [50:1187](#)
- \@badtab [14:258](#), [41:22](#),
[41:87](#), [41:108](#), [41:114](#), [41:121](#), [41:148](#)
- \@begin@tempboxa
[40:27](#), [40:46](#), [40:61](#), [40:246](#), [40:265](#),
[40:334](#), [40:361](#), [40:576](#), [40:584](#), [1288](#)
- \@begindocumenthook
... [20:61](#), [20:127](#), [20:130](#), [20:182](#),
[20:185](#), [47:53](#), [50:1115](#), [50:1136](#), [234](#)
- \@beginndvi
... [54:833](#), [54:901](#), [54:960](#), [54:988](#), [1131](#)
- \@beginndvibox
... [53:449](#), [53:450](#), [54:82](#), [54:989](#), [1126](#)
- \@beginparpenalty .. [18:15](#), [38:467](#),
[38:479](#), [38:505](#), [39:23](#), [39:201](#), [862](#)
- \@begintheorem [43:43](#), [43:48](#)
- \@bezier [42:683](#), [42:682](#)
- \@bibitem [47:3](#), [47:8](#)
- \@biblabel [47:4](#), [47:97](#)
- \@bitor [54:11](#),
[54:1155](#), [54:1175](#), [54:1211](#), [54:1234](#),
[54:1301](#), [54:1386](#), [54:1396](#), [54:1535](#),
[54:1545](#), [54:1688](#), [54:1699](#), [54:1842](#),
[54:1923](#), [54:2009](#), [54:2127](#), [54:2252](#)
- \@botlist ... [54:61](#), [54:382](#), [54:384](#),
[54:429](#), [54:431](#), [54:602](#), [54:608](#),
[54:625](#), [54:1012](#), [54:1021](#), [54:1022](#),

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=ltthyphen.dtx, 57=ltfinal.dtx

- 54:1263, 54:1266, 54:1301, 54:1396,
54:1545, 54:1699, 54:2455, 54:2483
- `\@botnum` .. 45:274, 54:107, 54:1260,
54:1261, 54:1266, 54:1270, 54:1878,
54:1883, 54:1959, 54:1964, 54:2051,
54:2058, 54:2447, 54:2475, 54:2517
- `\@botroom` 45:275, 54:108,
54:1263, 54:1266, 54:2448, 54:2476
- `\@boxfpsbit` . 54:2575, 54:2577, 54:2582
- `\@break@loop` 1297
- `\@break@tfor`
... 13:31, 20:593, 20:610, 32:98, 1297
- `\@bsphack` ... 18:37, 18:126, 18:387,
18:403, 18:421, 18:437, 35:102,
35:172, 36:82, 44:186, 44:187, 45:52,
45:121, 45:310, 46:6, 46:18, 46:23,
46:35, 47:63, 47:80, 54:2382, 805
- `\@caption` 45:12, 45:14
- `\@capytype` 45:5, 45:9, 45:12,
45:40, 45:88, 45:109, 45:157, 54:2539
- `\@car` 19:14, 21:107, 21:128, 06:53
- `\@carcube` 06:55, 06:135, 06:633
- `\@cclv` 02:16,
54:298, 54:302, 54:380, 54:381,
54:410, 54:427, 54:428, 54:457,
54:485, 54:692, 54:696, 57:67, 1131
- `\@cclvi` 04:30, 04:58, 50:1306, 50:1439,
50:1528, 02:21, 02:57, 02:82,
02:93, 02:95, 02:99, 02:159, 02:173
- `\@cdr` ... 20:251, 06:53, 06:776, 06:777
- `\@centercr` .. 37:413, 37:451, 37:456,
37:461, 37:471, 37:475, 37:479, 1336
- `\@centering` 38:360, 38:361,
38:369, 38:372, 38:375, 38:534, 38:538
- `\@cflb` 54:602, 54:608, 54:993
- `\@cflt` 54:601, 54:605, 54:993
- `\@changed@cmd` 21:3, 21:81,
21:243, 24:132, 24:330, 57:476, 1310
- `\@changed@x` . 21:3, 21:231, 21:239, 1309
- `\@changed@x@err` 1317
- `\@changed@x@mouth` 21:231, 21:239, 1309
- `\@charlb` 20:439, 20:447
- `\@charrb` 20:441, 20:447
- `\@chclass`
41:271, 41:272, 41:335, 41:348, 41:353
- `\@check@IncludeInRelease` 03:71
- `\@check@c` 06:189, 06:191
- `\@check@eq` 06:195, 06:196, 06:200
- `\@checkcommand` 1301
- `\@checkend`
... 37:16, 37:82, 37:146, 37:337,
37:352, 37:369, 37:379, 37:389, 1311
- `\@chnum` 41:279,
41:298, 41:335, 41:350, 41:351, 41:352
- `\@circ`
42:621, 42:639, 42:648, 42:653, 42:656
- `\@circle` 42:604, 42:605
- `\@circlefnt` 42:125,
42:128, 42:447, 42:490, 42:519,
42:544, 42:614, 42:630, 42:662, 42:677
- `\@cite` 47:36, 47:95
- `\@cite@ofmt` 47:44, 47:96
- `\@citea` 47:35, 47:37
- `\@citeb` 47:38, 47:39,
47:40, 47:43, 47:44, 47:66, 47:67,
47:68, 47:69, 47:83, 47:84, 47:85, 47:86
- `\@citex` . 47:20, 47:21, 47:29, 47:34, 999
- `\@citex@checkblank` 47:17, 47:18, 47:30
- `\@classi` 41:271, 41:307
- `\@classii` 41:271, 41:321
- `\@classiii` 41:271, 41:326
- `\@classiv` 41:169, 41:180, 41:272
- `\@classoptionslist`
... 50:9, 50:546, 50:561, 50:562,
50:579, 50:810, 50:811, 50:839,
50:840, 50:866, 50:867, 50:1808, 1344
- `\@classv` 41:272, 41:332
- `\@classz` . 41:168, 41:179, 41:271, 1250
- `\@cline` 41:367
- `\@clnht` 42:195, 42:196, 42:204, 42:206,
42:208, 42:218, 42:225, 42:273, 42:671
- `\@clnwd` 42:197,
42:203, 42:207, 42:209, 42:210, 42:671
- `\@cls@pkg` 08:2690,
50:353, 50:354, 50:367, 50:368,
50:955, 50:965, 50:1013, 50:1060,
50:1090, 50:1142, 50:1172, 50:1174,
50:1191, 50:1736, 50:1758, 50:1788
- `\@clsextension` 50:31,
50:156, 50:164, 50:222, 50:383,
50:399, 50:411, 50:493, 50:517,
50:528, 50:546, 50:560, 50:578,
50:677, 50:692, 50:716, 50:809,
50:838, 50:865, 50:959, 50:1006,
50:1033, 50:1094, 50:1107, 50:1147,
50:1162, 50:1613, 51:102, 51:168, 1068
- `\@clubpenalty` .. 20:7, 20:25, 20:95,
20:152, 39:128, 39:227, 44:106, 44:135
- `\@colht` 20:22, 20:92, 20:149, 45:273,
45:275, 45:278, 45:284, 45:285,
45:298, 45:299, 54:112, 54:229,
54:240, 54:249, 54:250, 54:385,
54:397, 54:432, 54:445, 54:474,
54:502, 54:519, 54:525, 54:529,
54:538, 54:543, 54:712, 54:730,
54:734, 54:741, 54:865, 54:923,
54:982, 54:1051, 54:1089, 54:1133,

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=ltthyphen.dtx, 57=ltfinal.dtx

- 54:1158, 54:1177, 54:1217, 54:1239,
54:2142, 54:2268, 54:2640, 57:155
- `\@colnum`
 . 45:276, 54:109, 54:1269, 54:1314,
54:1384, 54:1385, 54:1413, 54:1421,
54:1533, 54:1534, 54:1562, 54:1570,
54:1686, 54:1687, 54:1719, 54:1731,
54:1840, 54:1841, 54:1878, 54:1883,
54:1921, 54:1922, 54:1959, 54:1964,
54:2007, 54:2008, 54:2050, 54:2057,
54:2443, 54:2471, 54:2510, 54:2750
- `\@colroom` 20:23,
20:93, 20:150, 54:113, 54:250,
54:271, 54:272, 54:283, 54:286,
54:385, 54:432, 54:1051, 54:1268,
54:1313, 54:1380, 54:1383, 54:1412,
54:1529, 54:1532, 54:1561, 54:1681,
54:1685, 54:1718, 54:1836, 54:1839,
54:1917, 54:1920, 54:2002, 54:2006,
54:2444, 54:2472, 54:2655, 54:2660,
54:2705, 54:2710, 54:2755, 57:154, 1318
- `\@combinedblfloats`
 . . 54:1024, 54:2814, 54:2856, 54:2893
- `\@combinefloats` . 54:708, 54:992, 1183
- `\@comdblflleft` 54:1024
- `\@comfleleft` ... 54:994, 54:1010, 54:1024
- `\@compatibility` 1292
- `\@cons`
 22:83, 22:92, 06:52, 45:193, 45:215,
45:239, 45:379, 54:235, 54:1162,
54:1181, 54:1197, 54:1221, 54:1223,
54:1243, 54:1245, 54:1416, 54:1484,
54:1565, 54:1629, 54:1724, 54:1797,
54:1871, 54:1952, 54:2041, 54:2144,
54:2167, 54:2270, 54:2295, 54:2312,
54:2313, 54:2756, 02:196, 02:214
- `\@contentsline@destination`
 44:213, 44:215, 44:222, 969
- `\@contfield` 41:50, 41:141, 41:153
- `\@copy@`..... 101
- `\@copy@DeclareRobustCommand`
 06:497, 06:567, 06:588, 06:666, 06:669
- `\@copy@newcommand`
 06:318, 06:498, 06:567,
06:609, 06:639, 06:666, 06:672, 102
- `\@copytexsys` 1291
- `\@ctrerr`
 14:243, 22:208, 22:212, 22:226, 22:234
- `\@curfield` 41:16, 41:41, 41:47,
41:51, 41:52, 41:54, 41:130, 41:131
- `\@curline` 41:16, 41:27,
41:39, 41:44, 41:53, 41:54, 41:55,
41:90, 41:91, 41:103, 41:128, 41:129
- `\@curr@enc` 21:172, 21:174
- `\@curr@file`
 . . . 20:227, 20:228, 20:237, 20:239,
20:263, 20:271, 20:452, 20:471,
20:640, 20:655, 50:940, 50:1253,
50:1258, 50:1264, 50:1270, 50:1274,
50:1285, 50:1294, 50:1321, 50:1384,
50:1389, 50:1395, 50:1418, 50:1427,
50:1453, 52:266, 52:374, 52:376, 1115
- `\@curr@file@reqd`
 52:266, 52:376, 52:380, 1115
- `\@currbox` 45:60, 45:91, 45:95,
45:129, 45:160, 45:164, 45:193,
45:214, 45:215, 45:239, 45:257,
45:259, 45:261, 45:319, 45:322,
45:327, 45:331, 54:211, 54:212,
54:223, 54:224, 54:226, 54:227,
54:235, 54:309, 54:310, 54:1127,
54:1128, 54:1376, 54:1379, 54:1387,
54:1410, 54:1414, 54:1416, 54:1431,
54:1472, 54:1484, 54:1526, 54:1528,
54:1536, 54:1559, 54:1563, 54:1565,
54:1580, 54:1617, 54:1629, 54:1676,
54:1679, 54:1716, 54:1721, 54:1724,
54:1741, 54:1786, 54:1797, 54:1829,
54:1846, 54:1860, 54:1871, 54:1911,
54:1927, 54:1941, 54:1952, 54:1993,
54:2030, 54:2041, 54:2081, 54:2085,
54:2096, 54:2102, 54:2104, 54:2108,
54:2113, 54:2122, 54:2131, 54:2137,
54:2144, 54:2167, 54:2202, 54:2206,
54:2218, 54:2225, 54:2227, 54:2231,
54:2237, 54:2247, 54:2262, 54:2270,
54:2295, 54:2313, 54:2322, 54:2545,
54:2546, 54:2575, 54:2605, 54:2610,
54:2661, 54:2664, 54:2676, 54:2684,
54:2711, 54:2714, 54:2726, 54:2734,
54:2751, 54:2756, 02:275, 02:276, 02:277
- `\@currdir`
 . . . 01:92, 01:114, 01:116, 01:122,
01:124, 01:130, 01:132, 01:137,
01:139, 01:149, 01:162, 01:227,
01:240, 01:253, 50:1232, 50:1258,
50:1367, 50:1389, 50:1418, 50:1501, 9
- `\@current@cmd` 21:43, 24:334, 1309
- `\@currentHpage` ... 36:240, 36:241, 810
- `\@currentHref` 17:34,
17:38, 35:99, 35:107, 35:117, 35:121,
35:147, 36:239, 55:50, 55:54, 810
- `\@currentcounter`
 22:16, 35:114, 35:128,
35:141, 35:149, 35:179, 35:181,
35:191, 36:242, 38:365, 38:519,
40:485, 43:53, 43:55, 45:471, 1347
- `\@currentlabel` ... 35:106, 35:130,

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx,
16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
56=ltthyphen.dtx, 57=ltfinal.dtx

35:143, 35:150, 35:174, 35:182,	50:932, 50:937, 50:942, 50:951,
35:193, 35:204, 35:212 , 36:236,	50:955, 50:957, 50:963, 50:965,
38:364, 38:518, 40:486, 40:503,	50:966, 50:968, 50:974, 50:983,
40:520, 45:472, 45:490, 45:508, 854	50:985, 50:991, 50:994, 50:997,
<code>\@currentlabelname</code>	50:999, 50:1000, 50:1004, 50:1008,
..... 35:98 , 35:107 , 36:238 , 810	50:1015, 50:1017, 50:1018, 50:1023,
<code>\@currenttarget</code> 35:170	50:1026, 50:1030, 50:1034, 50:1039,
<code>\@currenttitle</code> 35:169	50:1051, 50:1075, 50:1076, 50:1078,
<code>\@currenvir</code>	50:1080, 50:1082, 50:1083, 50:1120,
.... 14:249 , 37:3 , 37:251 , 37:272 ,	50:1172, 50:1174, 50:1181, 50:1191,
37:290 , 37:300 , 37:390 , 39:112 ,	50:1736, 50:1758, 50:1788, 51:64,
40:188 , 50:1296 , 50:1308 , 50:1316 ,	51:67, 51:78, 51:109, 51:111, 51:170,
50:1320 , 50:1327 , 50:1429 , 50:1441 ,	51:173, 51:187, 51:189, 51:200,
50:1449 , 50:1453 , 50:1459 , 50:1518 ,	51:258, 08:368, 08:374, 08:429, 1092
50:1530 , 50:1538 , 50:1542 , 50:1548 ,	<code>\@currnamestack</code>
07:1617 , 07:1720 , 07:1735 , 07:1746 , 405	.. 50:33 , 50:136 , 52:546 , 08:426 , 1046
<code>\@currenvline</code> 14:249 , 37:252 ,	<code>\@curroption</code> 1285
37:273 , 37:291 , 37:301 , 37:391 , 40:189	<code>\@curroptions</code> 50:539 ,
<code>\@currtext</code>	50:547 , 50:597 , 50:616 , 50:1181 , 50:1182
. 50:30 , 50:42 , 50:57 , 50:70 , 50:79 ,	<code>\@currpath</code>
50:117 , 50:379 , 50:382 , 50:383 , 50:15 , 50:46 , 50:139 , 50:351 ,
50:398 , 50:399 , 50:410 , 50:411 ,	50:353 , 50:907 , 50:925 , 50:932 ,
50:517 , 50:528 , 50:539 , 50:546 ,	50:941 , 50:983 , 50:984 , 50:1011 , 1051
50:560 , 50:578 , 50:607 , 50:687 ,	<code>\@currpkg@reqd</code> 50:381 ,
50:700 , 50:702 , 50:710 , 50:724 ,	50:935 , 50:937 , 50:946 , 50:979 ,
50:908 , 50:909 , 50:911 , 50:915 ,	50:993 , 50:996 , 50:1011 , 50:1013 , 1067
50:918 , 50:920 , 50:925 , 50:930 ,	<code>\@currsize</code> 29:640 , 1284
50:932 , 50:937 , 50:943 , 50:951 ,	<code>\@currtype</code>
50:957 , 50:959 , 50:963 , 50:968 ,	54:1152 , 54:1153 , 54:1154 , 54:1155 ,
50:974 , 50:983 , 50:986 , 50:991 ,	54:1172 , 54:1173 , 54:1174 , 54:1175 ,
50:994 , 50:997 , 50:999 , 50:1000 ,	54:1301 , 54:1386 , 54:1396 , 54:1535 ,
50:1002 , 50:1006 , 50:1015 , 50:1017 ,	54:1545 , 54:1688 , 54:1699 , 54:1842 ,
50:1018 , 50:1023 , 50:1026 , 50:1029 ,	54:1923 , 54:2009 , 54:2127 , 54:2252 ,
50:1033 , 50:1039 , 50:1052 , 50:1057 ,	54:2545 , 54:2547 , 54:2548 , 54:2551
50:1058 , 50:1063 , 50:1069 , 50:1073 ,	<code>\@curtab</code> 41:11 , 41:26 , 41:86 , 41:87 ,
50:1075 , 50:1076 , 50:1078 , 50:1080 ,	41:88 , 41:94 , 41:95 , 41:98 , 41:102 ,
50:1082 , 50:1083 , 50:1086 , 50:1092 ,	41:103 , 41:107 , 41:146 , 41:147 , 1316
50:1094 , 50:1107 , 50:1120 , 50:1147 ,	<code>\@curtabmar</code> 41:11 , 41:25 , 41:26 , 41:38 ,
50:1150 , 50:1162 , 50:1180 , 50:1181 ,	41:44 , 41:89 , 41:102 , 41:106 , 41:107
51:64 , 51:67 , 51:102 , 51:109 ,	<code>\@d@r</code> 01:145 , 01:146
51:111 , 51:168 , 51:170 , 51:173 , 1066	<code>\@dashbox</code>
<code>\@currextension</code> 1285 42:324 , 42:325 , 42:326 , 42:327 ,
<code>\@currlist</code> 45:193 ,	42:328 , 42:331 , 42:337 , 42:339 ,
45:215 , 45:379 , 54:63 , 54:309 ,	42:349 , 42:351 , 42:352 , 42:353 ,
54:386 , 54:389 , 54:433 , 54:436 , 54:2312	42:354 , 42:358 , 42:362 , 42:365 ,
<code>\@currname</code> 03:71 , 03:116 ,	42:382 , 42:384 , 42:385 , 42:386 ,
03:124 , 20:762 , 20:784 , 20:790 ,	42:387 , 42:390 , 42:394 , 42:396 ,
50:29 , 50:41 , 50:56 , 50:69 , 50:78 ,	42:405 , 42:407 , 42:408 , 42:409 ,
50:116 , 50:351 , 50:353 , 50:365 ,	42:410 , 42:413 , 42:417 , 42:422 , 42:673
50:367 , 50:379 , 50:382 , 50:398 ,	<code>\@dashcnt</code>
50:410 , 50:539 , 50:607 , 50:700 , 42:317 , 42:319 , 42:320 , 42:321 ,
50:702 , 50:710 , 50:724 , 50:906 ,	42:322 , 42:323 , 42:336 , 42:338 ,
50:909 , 50:911 , 50:915 , 50:918 ,	42:341 , 42:343 , 42:344 , 42:345 ,
50:920 , 50:925 , 50:927 , 50:930 ,	42:347 , 42:348 , 42:361 , 42:364 ,

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpara.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=ltthyphen.dtx, 57=ltfinal.dtx

- 42:376, 42:377, 42:378, 42:379,
42:380, 42:381, 42:393, 42:395,
42:398, 42:399, 42:400, 42:401,
42:403, 42:404, 42:416, 42:421, [42:673](#)
- `\@dashdim` 42:316, 42:317,
42:318, 42:319, 42:320, 42:322,
42:325, 42:327, 42:328, 42:329,
42:336, 42:338, 42:340, 42:341,
42:342, 42:343, 42:344, 42:347,
42:351, 42:353, 42:354, 42:355,
42:363, 42:366, 42:375, 42:376,
42:377, 42:378, 42:380, 42:384,
42:386, 42:387, 42:388, 42:393,
42:395, 42:397, 42:398, 42:399,
42:400, 42:403, 42:407, 42:409,
42:410, 42:411, 42:419, 42:424, [42:673](#)
- `\@date` 44:9, [44:33](#)
- `\@dbflt` 45:32, [45:264](#), [1305](#)
- `\@dblarg` .. 44:54, 44:142, 45:12, [06:798](#)
- `\@dbldeferlist` 45:239,
54:66, 54:443, 54:448, 54:450,
54:1090, 54:1097, 54:1098, 54:2252,
54:2295, 54:2459, 54:2488, [1330](#)
- `\@dblfloat` [45:31](#)
- `\@dblfloatplacement`
..... 20:31, 20:101, 20:159,
[45:280](#), 54:399, 54:447, 54:2440,
54:2468, 54:2818, 54:2860, 54:2899
- `\@dblflset` [45:26](#)
- `\@dblfpbot` 45:290, 45:304, [54:2943](#)
- `\@dblfpsep` 45:289, 45:303, [54:2943](#)
- `\@dblfpstop` 45:288, 45:302, [54:2943](#)
- `\@dbltoplist` 54:65, 54:230,
54:233, 54:235, 54:395, 54:396,
54:443, 54:444, 54:1029, 54:1033,
54:1035, 54:1036, 54:2139, 54:2144,
54:2264, 54:2270, 54:2458, 54:2486
- `\@dbltopnum` 45:283,
45:297, 54:105, 54:125, 54:236,
54:238, 54:1040, 54:2078, 54:2079,
54:2143, 54:2146, 54:2174, 54:2179,
54:2199, 54:2200, 54:2269, 54:2273,
54:2302, 54:2307, 54:2451, 54:2479
- `\@dbltoproom` 45:284, 45:286, 45:298,
45:300, 54:106, 54:2081, 54:2084,
54:2085, 54:2094, 54:2095, 54:2098,
54:2101, 54:2104, 54:2108, 54:2112,
54:2116, 54:2121, 54:2141, 54:2202,
54:2205, 54:2206, 54:2215, 54:2216,
54:2217, 54:2220, 54:2224, 54:2227,
54:2231, 54:2236, 54:2240, 54:2245,
54:2246, 54:2267, 54:2452, 54:2480
- `\@dec@text@cmd` [21:3](#)
- `\@declarecommandcopylisthook` ...
.. 06:494, [06:496](#), 06:508, 07:1161, [99](#)
- `\@declaredoptions`
..... 50:8, 50:500, 50:543,
50:581, 50:602, 50:621, 50:1113, [1285](#)
- `\@declareoption` 50:498, 50:499, 50:507
- `\@defaultfamilyhook` [29:430](#), 29:665,
29:682, 29:697, 29:702, 29:717, [1338](#)
- `\@defaultsubs` 24:636, 24:670,
[24:682](#), 37:57, 37:122, 37:161, [1314](#)
- `\@defaultunits`
... 24:277, 24:281, 24:282, 24:283,
24:298, [24:391](#), 26:180, 26:182, 42:13
- `\@defaultunitsset` ... 40:74, 40:85,
[42:8](#), 42:29, 42:30, 42:32, 42:34,
42:60, 42:63, 42:84, 42:85, 42:107,
42:108, 42:164, 42:241, 42:316,
42:318, 42:332, 42:340, 42:342,
42:357, 42:479, 42:480, 42:611,
42:647, 42:689, 42:690, 42:692,
42:693, 42:696, 42:697, 42:699,
42:700, 42:711, 42:712, 42:714,
42:715, 42:717, 42:718, 42:720, 42:721
- `\@defdefault@ds` 50:498, 50:503, 50:508
- `\@deferlist` 54:64,
54:382, 54:391, 54:392, 54:395,
54:400, 54:402, 54:408, 54:429,
54:438, 54:440, 54:1052, 54:1060,
54:1061, 54:1072, 54:1077, 54:1078,
54:1386, 54:1484, 54:1535, 54:1629,
54:1688, 54:1797, 54:1842, 54:1871,
54:1923, 54:1952, 54:2009, 54:2041,
54:2127, 54:2167, 54:2457, 54:2485
- `\@definecounter` 22:12, [22:38](#),
38:349, 39:258, 39:259, 39:260,
39:261, 43:8, 43:16, 45:396, 45:398
- `\@depth` 26:192, 30:569, 30:570, 30:572,
30:573, [06:26](#), 40:550, 40:603,
40:610, 41:187, 41:219, 42:227,
42:300, 42:303, 42:324, 42:333,
42:383, 42:391, 42:726, 42:782, 54:2351
- `\@dir`
01:144, 01:147, 01:149, 01:151, 01:152
- `\@disable@packageload@do`
..... 50:930, [52:477](#)
- `\@dischyp`
.. 40:386, 40:407, [06:835](#), 06:869, [1333](#)
- `\@doclearpage` 54:294, [54:369](#)
- `\@documentclass` [1287](#)
- `\@documentclasshook`
... 50:3, 50:815, 50:843, 50:870, [1292](#)
- `\@doendpe` 37:338, 37:353,
37:370, 37:380, [39:123](#), 39:159, [870](#)

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpara.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscomp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- `\@dofilelist` 20:759,
[20:801](#), [37:42](#), [37:107](#), [37:156](#), [1309](#)
- `\@dofilelist@hash` 20:790, [20:796](#)
- `\@dofilelist@size` 20:784, [20:796](#)
- `\@donoparitem` [39:175](#), [39:189](#)
- `\@dot` [42:604](#), [42:642](#)
- `\@dotsep` [44:240](#), [44:263](#)
- `\@dottedtocline` [44:225](#), [44:251](#), [44:252](#)
- `\@downline` [42:297](#), [42:301](#), [42:306](#)
- `\@downvector` [42:268](#), [42:306](#)
- `\@eha` [03:195](#), [14:219](#),
[14:237](#), [14:239](#), [14:241](#), [14:250](#),
[14:252](#), [14:284](#), [20:224](#), [20:268](#),
[20:286](#), [21:70](#), [21:102](#), [24:29](#), [24:59](#),
[24:103](#), [24:145](#), [24:250](#), [24:316](#),
[24:402](#), [26:107](#), [28:26](#), [28:71](#), [28:100](#),
[28:173](#), [28:204](#), [28:234](#), [28:266](#),
[28:494](#), [28:515](#), [28:567](#), [28:618](#),
[28:663](#), [28:668](#), [28:723](#), [28:841](#),
[28:845](#), [28:849](#), [28:884](#), [28:888](#),
[28:892](#), [28:949](#), [28:959](#), [28:1044](#),
[28:1049](#), [28:1052](#), [28:1084](#), [28:1087](#),
[28:1160](#), [28:1163](#), [28:1166](#), [28:1233](#),
[28:1239](#), [32:146](#), [33:24](#), [33:45](#),
[33:876](#), [33:885](#), [37:250](#), [37:271](#),
[37:289](#), [37:299](#), [06:292](#), [06:331](#),
[06:359](#), [47:71](#), [47:88](#), [54:2376](#), [54:2392](#)
- `\@ehb` [14:219](#), [14:244](#), [14:270](#), [14:272](#),
[14:274](#), [14:276](#), [54:232](#), [54:388](#), [54:435](#)
- `\@ehc` [03:184](#), [14:219](#),
[14:279](#), [14:282](#), [14:288](#), [14:290](#),
[06:128](#), [06:155](#), [37:655](#), [37:672](#),
[37:686](#), [37:701](#), [37:714](#), [38:422](#),
[39:251](#), [44:31](#), [06:481](#), [06:522](#), [50:1788](#)
- `\@ehd` [03:102](#), [03:159](#),
[14:219](#), [14:246](#), [14:254](#), [14:257](#),
[14:259](#), [14:265](#), [28:119](#), [05:88](#),
[41:100](#), [41:109](#), [45:6](#), [50:757](#), [50:1013](#)
- `\@elt` 20:440, [21:1598](#), [21:1600](#),
[22:22](#), [22:37](#), [22:103](#), [22:106](#), [29:142](#),
[29:153](#), [29:155](#), [29:156](#), [29:181](#),
[29:506](#), [29:508](#), [29:509](#), [29:519](#),
[29:743](#), [30:17](#), [30:25](#), [06:52](#), [54:4](#),
[54:7](#), [54:11](#), [54:23](#), [54:26](#), [54:27](#),
[54:28](#), [54:29](#), [54:34](#), [54:35](#), [54:36](#),
[54:37](#), [54:38](#), [54:39](#), [54:40](#), [54:41](#),
[54:43](#), [54:47](#), [54:53](#), [54:54](#), [54:55](#),
[54:56](#), [54:488](#), [54:705](#), [54:994](#),
[54:1005](#), [54:1010](#), [54:1020](#), [54:1032](#),
[54:1034](#), [54:1062](#), [54:1079](#), [54:1099](#),
[54:1118](#), [54:1131](#), [54:1138](#), [54:1189](#),
[54:1192](#), [54:1201](#), [54:2418](#), [54:2430](#), [700](#)
- `\@empty` [13:14](#), [1302](#)
- `\@emptycol` [54:196](#),
[54:243](#), [54:246](#), [54:275](#), [54:279](#), [1296](#)
- `\@enc@info`
... [21:7](#), [21:12](#), [21:103](#), [21:211](#), [21:221](#)
- `\@end@check@IncludeInRelease` ...
..... [03:143](#), [03:145](#)
- `\@end@tempboxa`
..... [40:36](#), [40:55](#), [40:64](#), [40:258](#),
[40:270](#), [40:352](#), [40:374](#), [40:582](#), [40:592](#)
- `\@enddocument@kernel@warnings` ...
.. [37:43](#), [37:52](#), [37:108](#), [37:117](#), [37:176](#)
- `\@enddocumenthook`
..... [37:145](#), [50:1115](#), [50:1137](#)
- `\@endfloatbox`
..... [45:190](#), [45:211](#), [45:236](#), [45:248](#)
- `\@endparenv` [39:120](#), [39:123](#), [870](#)
- `\@endparpenalty` ... [18:16](#), [38:468](#),
[38:480](#), [38:506](#), [39:23](#), [39:124](#), [862](#)
- `\@endpbox` [41:193](#),
[41:236](#), [41:266](#), [41:333](#), [41:384](#), [41:387](#)
- `\@endpfalse` [37:256](#), [37:276](#), [37:294](#),
[37:304](#), [39:129](#), [39:133](#), [39:137](#),
[39:138](#), [39:140](#), [40:138](#), [40:191](#), [870](#)
- `\@endptrue`
.. [39:124](#), [39:126](#), [39:136](#), [39:140](#), [871](#)
- `\@endpreamblehook` [455](#)
- `\@endtheorem` [43:13](#), [43:19](#), [43:25](#), [43:48](#)
- `\@enlargepage` [54:2361](#), [54:2366](#), [54:2368](#)
- `\@ensuredmath` [38:433](#), [38:435](#)
- `\@enumctr` [39:265](#), [39:268](#), [39:269](#)
- `\@enumdepth`
.. [39:257](#), [39:263](#), [39:264](#), [39:265](#), [875](#)
- `\@enumspacing` [875](#)
- `\@eqcnt` [38:357](#), [38:419](#),
[38:424](#), [38:521](#), [38:536](#), [38:537](#), [38:539](#)
- `\@eqncr`
[38:370](#), [38:388](#), [38:425](#), [38:426](#), [38:523](#)
- `\@eqnnum` [38:351](#),
[38:352](#), [38:423](#), [38:454](#), [38:513](#), [1283](#)
- `\@eqnset` [38:357](#), [38:535](#), [1285](#)
- `\@eqnswfalse` [38:387](#)
- `\@eqnswtrue`
..... [38:359](#), [38:366](#), [38:424](#), [38:520](#)
- `\@eqpen`
[38:357](#), [38:391](#), [38:393](#), [38:404](#), [38:414](#)
- `\@er@ext` [1286](#)
- `\@err@` [14:37](#), [14:41](#),
[14:44](#), [14:52](#), [14:64](#), [14:68](#), [14:71](#), [14:79](#)
- `\@esphack`
... [18:39](#), [18:132](#), [18:392](#), [18:409](#),
[18:426](#), [18:443](#), [35:109](#), [35:175](#),
[36:86](#), [44:186](#), [44:187](#), [45:385](#), [46:17](#),
[46:19](#), [46:34](#), [47:74](#), [47:90](#), [54:2384](#), [805](#)

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=ltthyphen.dtx, 57=ltfinal.dtx

<code>\@evenfoot</code>	<code>\@expl@@@shipout@add@foreground@picture@@n</code> 53:417, 53:508
.. 49:12, 49:15, 54:820, 54:891, 54:950	
<code>\@evenhead</code>	<code>\@expl@char@generate@@nn</code>
.. 49:12, 49:15, 54:819, 54:890, 54:949 05:145, 06:814, 1341
<code>\@execute@begin@hook</code>	<code>\@expl@cs@(thing)@spec@@N</code> 1340
..... 37:253, 37:257, 37:260, 828	<code>\@expl@cs@argument@spec@@N</code> ... 1355
<code>\@executeoption</code>	<code>\@expl@cs@parameter@spec@@N</code>
..... 1286	... 05:140, 05:141, 05:142, 05:153,
<code>\@expandtwoargs</code>	06:657, 06:710, 06:721, 06:726, 1355
..... 03:188, 06:217, 50:224,	<code>\@expl@cs@prefix@spec@@N</code>
50:350, 50:545, 50:581, 50:635, 50:644 05:138, 05:152, 06:655, 06:725
<code>\@expast</code>	<code>\@expl@@@filehook@clear@replacement@flag@@\@expl@cs@replacement@spec@@N</code> ...
..... 41:239, 41:267 50:449,
<code>\@expl@@@filehook@clear@replacement@flag@@\@expl@cs@replacement@spec@@N</code> ...	50:472, 52:295, 52:318, 52:339, 52:519
..... 50:449, 05:143, 05:154, 06:615,
<code>\@expl@@@filehook@drop@extension@@N</code>	06:650, 06:657, 06:711, 06:722, 06:726
..... 52:292, 52:293, 52:315,	<code>\@expl@cs@to@str@@N</code>
52:316, 52:336, 52:337, 52:521, 1114	... 05:133, 05:136, 05:148, 05:150,
<code>\@expl@@@filehook@file@pop@@</code> ...	06:489, 06:581, 06:601, 06:603,
..... 50:948, 52:157, 52:525, 52:535	06:604, 06:617, 06:630, 06:642,
<code>\@expl@@@filehook@file@pop@assign@@nmmn</code>	06:646, 06:647, 06:652, 06:696,
..... 52:162, 52:527, 1110	06:705, 06:709, 06:717, 06:719, 1340
<code>\@expl@@@filehook@file@push@@</code> ...	<code>\@expl@finalise@setup@@</code>
..... 50:931, 52:151, 52:523 05:30, 57:268, 57:269
<code>\@expl@@@filehook@if@file@replaced@@TF</code>	<code>\@expl@pop@filename@@</code>
.. 52:289, 52:312, 52:333, 52:517, 1115	05:26, 50:88, 50:92, 50:98, 50:109, 1340
<code>\@expl@@@filehook@if@no@extension@@nTF</code>	<code>\@expl@push@filename@@</code>
..... 52:285, 05:24, 50:37, 50:39, 50:52,
52:308, 52:329, 52:507, 52:509, 52:534	50:54, 50:64, 50:76, 50:96, 50:102, 1044
<code>\@expl@@@filehook@normalize@file@name@@w</code>	<code>\@expl@push@filename@aux@@</code>
..... 52:291, 52:314, 52:335, 52:515 08:2795, 05:25, 50:37,
<code>\@expl@@@filehook@resolve@file@subst@@w</code>	50:48, 50:60, 50:64, 50:82, 50:102, 303
..... 50:447,	<code>\@expl@str@if@eq@@nnTF</code> ... 05:137,
50:470, 52:288, 52:311, 52:332, 52:513	05:151, 06:453, 06:455, 50:350, 1340
<code>\@expl@@@filehook@set@curr@file@@nNN</code>	<code>\@expl@str@map@function@@NN</code>
..... 50:446, 50:469, 05:144, 05:155, 06:811, 109
50:900, 52:374, 52:380, 52:511, 1115	<code>\@expl@str@map@function@@NN and</code>
<code>\@expl@@@hook@curr@name@pop@@</code> ...	<code>\@expl@char@generate@@nn</code> ... 1341
..... 08:2897, 50:89	<code>\@expl@sys@load@backend@@</code>
<code>\@expl@@@hook@curr@name@push@@n</code> 1342 20:17, 05:21, 05:23
<code>\@expl@@@initialize@all@@</code>	<code>\@expl@text@uppercase@@n</code> 1348
..... 08:2897, 37:261, 08:1470	<code>\@extra@page@added</code> 37:68, 37:133, 53:387
<code>\@expl@@@mark@update@dbcol@structures@@</code>	<code>\@failedlist</code>
..... 48:501, 48:514, 54:466	54:1116,
<code>\@expl@@@mark@update@singlecol@structures@@</code>	54:1139, 54:1155, 54:1162, 54:1175,
..... 48:499, 48:513, 54:469	54:1181, 54:1197, 54:1211, 54:1234
<code>\@expl@@@shipout@add@background@box@@n</code>	<code>\@fcolmadefalse</code>
..... 53:417, 53:502	54:1107
<code>\@expl@@@shipout@add@background@picture@@n</code>	<code>\@fcolmadetrue</code>
..... 53:417, 53:506	54:1195
<code>\@expl@@@shipout@add@firstpage@material@@Nn</code>	<code>\@file-subst@<file></code>
..... 53:417, 53:499	1111
<code>\@expl@@@shipout@add@foreground@box@@n</code>	<code>\@filef@und</code> . 20:491, 20:503, 20:504,
..... 53:417, 53:504	20:507, 20:513, 20:545, 20:567,
	20:590, 20:607, 20:620, 20:673,
	52:155, 52:172, 52:188, 52:196, 1335
	<code>\@filehook@file@push</code>
	1066
	<code>\@filehook@set@CurrentFile</code>
	20:315, 20:376, 50:933, 52:152, 52:363

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=ltthyphen.dtx, 57=ltfinal.dtx

- \@filelist .. 20:63, 20:132, 20:187,
20:741, 20:742, 20:761, 29:750,
29:760, 29:770, 57:279, 57:698, 57:714
- \@filesfalse 20:200,
50:1228, 50:1229, 50:1363, 50:1364
- \@fileswithOptions 50:497,
50:605, 50:624, 50:801, 50:802,
50:806, 50:808, 50:836, 50:837,
50:863, 50:864, 50:891, 50:1104, 1285
- \@fileswith@ptions 50:786,
50:787, 50:794, 50:795, 50:799, 50:803
- \@fileswithoptions
..... 50:677, 50:684, 50:692, 50:775
- \@fileswtrue 20:5,
50:1211, 50:1214, 50:1271, 50:1275,
50:1346, 50:1349, 50:1404, 50:1408
- \@finalstrut
... 40:490, 40:507, 40:524, 40:593,
41:385, 45:477, 45:495, 45:513, 1299
- \@firstampfalse 41:254, 41:277, 41:294
- \@firstamptrue 41:262
- \@firstcolfirstmark
..... 54:2839, 54:2840, 54:2844
- \@firstcoltopmark ... 54:2837, 54:2845
- \@firstcolumnfalse
..... 54:2801, 54:2830, 54:2871
- \@firstcolumntrue
..... 20:28, 20:98, 20:156,
54:96, 54:205, 54:2805, 54:2848, 54:2877
- \@firstoffive 35:23, 35:25, 35:40, 35:57
- \@firstofone
... 04:12, 04:100, 04:108, 20:126,
20:181, 20:459, 04:166, 21:86,
21:171, 26:347, 28:54, 28:82, 28:147,
28:185, 28:215, 28:246, 28:994,
05:34, 05:61, 05:76, 37:144, 38:431,
06:212, 41:372, 45:10, 06:477,
06:482, 06:483, 06:486, 06:487,
47:38, 06:518, 47:66, 47:83, 06:523,
06:524, 06:527, 06:528, 50:949,
50:978, 52:352, 52:369, 57:359, 1340
- \@firstoftwo
... 03:85, 17:17, 20:506, 20:512,
20:546, 20:591, 20:608, 21:151,
22:244, 22:249, 24:219, 24:239,
28:998, 05:35, 29:511, 05:43, 33:798,
33:848, 33:864, 35:42, 35:78, 06:212,
06:454, 06:456, 06:478, 06:519,
49:16, 06:584, 50:161, 50:195,
50:207, 50:230, 50:248, 50:307,
50:337, 06:635, 06:750, 50:1797,
06:760, 06:770, 06:797, 06:821,
54:612, 54:621, 54:628, 01:71, 469
- \@firsttab 41:2, 41:74,
41:75, 41:76, 41:106, 41:118, 1297
- \@flcheckspace 54:1263, 54:1299, 54:2646
- \@flfail 54:1139, 54:1190,
54:1211, 54:1221, 54:1234, 54:1243
- \@float 45:26, 45:32
- \@floatboxreset 45:101, 45:170, 45:174
- \@floatpenalty
. 45:3, 45:53, 45:55, 45:58, 45:122,
45:124, 45:127, 45:191, 45:194,
45:199, 45:201, 45:212, 45:216,
45:221, 45:223, 45:237, 45:241,
45:311, 45:313, 45:317, 45:321, 45:379
- \@floatplacement ... 20:31, 20:101,
20:159, 45:271, 54:147, 54:207,
54:251, 54:477, 54:2441, 54:2469, 1287
- \@flsetnum 54:1260, 54:1296, 54:1384,
54:1533, 54:1686, 54:1840, 54:1921,
54:2007, 54:2078, 54:2199, 54:2614
- \@flsettextmin
..... 54:1359, 54:1510, 54:1656,
54:1825, 54:1907, 54:1989, 54:2630
- \@flstop 54:2506
- \@flsucceed 54:1132,
54:1140, 54:1189, 54:1223, 54:1245
- \@fltovf . 14:275, 45:93, 45:162, 45:322
- \@flupdates . 54:1266, 54:1311, 54:2747
- \@flushglue . 12:17, 37:452, 37:456,
37:462, 37:472, 37:475, 37:480,
37:530, 37:552, 39:76, 40:393, 40:414
- \@fnsymbol 22:194, 22:213
- \@font@aliasinfo 26:540
- \@font@info 24:134, 24:172,
24:178, 24:201, 24:202, 24:463,
24:480, 24:718, 25:2866, 26:31,
26:39, 26:47, 26:75, 26:88, 26:155,
26:201, 26:215, 26:226, 26:240,
26:256, 26:262, 26:275, 26:282,
26:289, 26:294, 26:304, 26:316,
26:328, 26:492, 26:504, 26:509,
26:516, 26:546, 26:559, 26:567,
28:280, 28:293, 28:304, 28:309,
28:314, 28:328, 28:345, 28:354,
28:369, 28:384, 28:442, 28:498,
28:597, 28:603, 28:647, 28:660,
28:743, 28:832, 28:875, 28:940,
28:1034, 28:1201, 28:1230, 33:63, 57:478
- \@font@series@contextfalse
..... 29:503, 29:539
- \@font@series@contexttrue
..... 29:522, 29:526, 29:538
- \@font@shape@subst@warning
..... 25:2833, 25:2836,
25:2841, 25:2899, 25:3178, 25:3181, 630

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpara.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

`\@font@warning` 24:4, 24:632, 24:637, 24:664, 24:671, 25:2844, 26:20, 26:34, 26:42, 26:50, 26:62, 26:78, 26:477, 26:491, 26:503, 26:508, 26:515, 26:558, 26:566, 27:30, 37:54, 37:119, 37:158, 57:300
`\@fontenc@load@list`
 .. 21:1599, 29:743, 30:17, 30:25, 1338
`\@fontswitch` 32:126, 32:128, 1297
`\@footnotemark` 45:454, 45:460, 45:522, 45:528, 45:529, 45:555
`\@footnotetext` 40:455, 45:454, 45:460, 45:461, 45:538, 45:544
`\@for` 13:16, 20:233, 20:305, 20:367, 20:414, 20:761, 47:36, 47:65, 47:82, 50:232, 50:249, 50:543, 50:562, 50:579, 50:597, 50:602, 50:616, 50:621, 50:657, 50:667, 50:1182, 50:1219, 50:1354, 999
`\@forced@seriesfalse`
 25:2782, 25:2795, 26:141
`\@forced@seriestrue` . 25:2786, 25:2797
`\@forloop` 13:19, 13:20
`\@fornoop` 13:15, 13:23, 13:29
`\@fortmp`
 13:17, 13:18, 13:26, 50:655, 50:657, 50:1218, 50:1219, 50:1353, 50:1354
`\@fpbot` 45:290, 45:304, 54:1137, 54:2937
`\@fpmin` 45:278, 45:287, 45:301, 54:111, 54:1194, 54:2449, 54:2477, 54:2764, 54:2781
`\@fps` 45:41, 45:42, 45:44, 45:47, 45:64, 45:110, 45:111, 45:113, 45:116, 45:133, 54:2537, 54:2539, 54:2542
`\@fpsaddddefault`
 . 45:45, 45:48, 45:114, 45:117, 54:2534
`\@fpsep` ... 45:289, 45:303, 54:1135, 54:1144, 54:1216, 54:1238, 54:2937
`\@fpstype` 54:1257, 54:1278, 54:1279, 54:1293, 54:1324, 54:1325, 54:1349, 54:1351, 54:1354, 54:1356, 54:1408, 54:1464, 54:1465, 54:1500, 54:1502, 54:1505, 54:1507, 54:1557, 54:1609, 54:1610, 54:1644, 54:1647, 54:1650, 54:1653, 54:1714, 54:1776, 54:1777, 54:1815, 54:1817, 54:1820, 54:1822, 54:1897, 54:1899, 54:1902, 54:1904, 54:1977, 54:1980, 54:1983, 54:1986, 54:2075, 54:2090, 54:2092, 54:2110, 54:2119, 54:2155, 54:2156, 54:2196, 54:2211, 54:2213, 54:2233, 54:2243, 54:2282, 54:2283, 54:2520, 54:2546, 54:2548, 54:2550, 54:2553, 54:2554, 54:2555, 54:2557, 54:2558, 54:2562, 54:2563, 54:2565, 54:2566, 54:2600, 54:2602, 54:2604, 54:2616, 54:2618, 54:2632, 54:2634, 54:2669, 54:2672, 54:2683, 54:2719, 54:2722, 54:2733
`\@fptop` 45:288, 45:302, 54:1134, 54:2937
`\@frameb@x`
 . 40:218, 40:257, 40:269, 40:273, 1303
`\@framebox` 40:225, 40:232, 40:236
`\@framepicbox` .. 40:225, 40:232, 40:294
`\@freelist` .. 45:60, 45:129, 45:319, 45:320, 54:25, 54:30, 54:44, 54:52, 54:211, 54:489, 54:706, 54:1006, 54:1021, 54:1035, 54:1140, 54:2312, 54:2313, 02:196, 02:214, 02:275, 1331
`\@generic@error` 1302
`\@generic@message` 1302
`\@getcirc` 42:437, 42:484, 42:513, 42:540, 42:612, 42:628
`\@getfpsbit` 54:1254, 54:1290, 54:2072, 54:2193, 54:2573
`\@getlarrow` 42:266, 42:274, 42:276
`\@getlinechar` 42:190, 42:229
`\@getpen` ... 18:35, 18:38, 18:47, 18:118
`\@getrarrow` 42:267, 42:274, 42:283
`\@glossaryfile` ... 46:21, 46:22, 46:31
`\@gnewline` 18:96, 18:102, 18:109, 18:112
`\@gobble` 03:163, 03:187, 13:6, 13:9, 14:101, 14:127, 14:147, 14:155, 14:180, 14:189, 14:202, 04:11, 17:15, 18:76, 04:98, 18:559, 20:64, 20:133, 20:188, 20:466, 20:468, 20:741, 21:47, 24:633, 24:666, 26:346, 27:26, 28:29, 28:31, 28:446, 28:457, 28:541, 28:608, 28:609, 28:638, 28:644, 28:652, 28:657, 28:675, 28:689, 28:699, 28:708, 28:721, 28:738, 28:747, 28:821, 28:823, 28:827, 28:835, 28:869, 28:878, 28:930, 28:932, 28:943, 28:1027, 28:1037, 28:1118, 28:1123, 28:1192, 28:1223, 29:140, 29:181, 05:33, 29:519, 05:65, 29:753, 29:763, 29:773, 05:88, 33:825, 06:111, 37:281, 06:133, 06:208, 06:231, 06:248, 06:252, 06:289, 06:295, 06:298, 06:308, 06:328, 06:334, 06:337, 06:346, 06:356, 06:362, 06:365, 06:374, 06:392, 06:396, 06:398, 06:399, 06:401, 06:409, 06:413, 06:415, 44:143, 44:144, 44:145, 44:146, 44:147, 44:200, 45:7, 47:11, 47:45, 47:46, 50:745, 50:928, 50:1203, 50:1267, 50:1294, 50:1398, 50:1427, 50:1511, 50:1516, 50:1590,

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- 50:1742, 50:1754, 52:261, 54:897,
54:898, 54:899, 54:956, 54:957,
54:958, 54:1201, 54:2432, 54:2765,
54:2782, 57:282, 57:411, 57:714, 1298
- \@gobble@AddToHook@args
..... 08:2913, 08:2914
- \@gobble@IncludeInRelease 03:71
- \@gobble@RemoveFromHook@arg
..... 08:2916, 08:2917
- \@gobble@om
... 44:180, 44:190, 44:192, 44:203, 89
- \@gobble@som
... 44:180, 44:191, 44:197, 44:204, 89
- \@gobble@with@sphack@om
... 44:180, 44:205, 54:827, 54:829, 89
- \@gobble@with@sphack@som
..... 44:180, 44:206, 54:828, 89
- \@gobblecr 18:557, 18:558
- \@gobblefour 28:25, 28:443,
28:599, 28:601, 28:605, 28:607,
28:617, 28:621, 28:745, 28:797,
06:208, 50:1296, 50:1429, 50:1518
- \@gobblethree
... 06:208, 50:128, 06:644, 06:656, 1046
- \@gobbletwo 13:12,
20:32, 20:102, 20:160, 24:638,
24:672, 28:133, 05:123, 05:128,
37:23, 37:55, 37:89, 37:120, 37:151,
37:159, 06:175, 06:176, 06:208,
49:11, 49:13, 50:1266, 50:1397,
50:1510, 52:260, 52:489, 57:306, 1322
- \@gtempa 20:685,
20:686, 20:692, 20:693, 20:694,
20:719, 20:720, 20:722, 20:723,
20:724, 06:126, 06:127, 06:181,
06:183, 41:3, 41:5, 41:6, 41:7,
41:8, 06:575, 06:583, 50:349, 50:351,
50:364, 50:365, 50:384, 50:386,
50:400, 50:402, 50:412, 50:414, 1293
- \@halfwidth 42:2,
42:126, 42:129, 42:131, 42:227,
42:299, 42:302, 42:324, 42:333,
42:349, 42:361, 42:364, 42:383,
42:391, 42:405, 42:416, 42:421,
42:679, 42:705, 42:724, 42:725,
42:726, 42:765, 42:780, 42:781, 42:782
- \@halignto 41:170, 41:174, 41:177, 41:191
- \@hangfrom 44:66, 44:117, 44:138
- \@height
... 18:397, 18:405, 18:431, 18:439,
21:313, 21:315, 26:191, 30:351,
30:569, 30:570, 30:572, 30:573,
06:26, 40:202, 40:207, 40:280,
40:290, 40:550, 40:603, 40:610,
41:186, 41:219, 41:359, 41:376,
42:227, 42:300, 42:303, 42:324,
42:333, 42:351, 42:359, 42:383,
42:391, 42:407, 42:414, 42:589,
42:599, 42:725, 42:781, 54:2351, 02:502
- \@highpenalty 18:119, 57:3
- \@hightab 41:11,
41:21, 41:23, 41:74, 41:86, 41:95,
41:96, 41:111, 41:146, 41:147, 1316
- \@hline . 42:167, 42:179, 42:226, 42:265
- \@holdpg 54:120, 54:298,
54:300, 54:301, 54:306, 54:307, 54:308
- \@hspace . 18:512, 18:513, 18:529, 1284
- \@hspacer 18:512, 18:528, 1284
- \@hvector 42:244, 42:259, 42:265
- \@ialph 1299
- \@icentercr 37:430, 37:431
- \@iden 06:215
- \@if 06:171, 06:172, 06:174
- \@if.. 1299
- \@if@DeclareRobustCommand
..... 09:143, 06:497,
06:554, 06:565, 06:566, 06:570,
06:664, 06:665, 06:668, 06:691, 101
- \@if@bottomfloats@TF 54:624,
54:635, 54:644, 54:659, 54:763, 1170
- \@if@flushbottom@TF .. 54:610, 54:761
- \@if@footnotes@TF 54:617,
54:633, 54:651, 54:657, 54:762, 1170
- \@if@newcommand .. 09:144, 09:155,
06:316, 06:498, 06:555, 06:566,
06:608, 06:620, 06:626, 06:692, 102
- \@if@newlist 1326
- \@if@pti@ns
50:224, 50:227, 50:229, 50:246, 50:247
- \@if@options
50:221, 50:222, 50:223, 50:963, 50:1058
- \@if@short@command 1299
- \@ifatmargin 41:55, 41:106
- \@ifbothcounters
..... 22:111, 22:125, 22:133,
22:141, 22:157, 22:159, 22:168, 22:170
- \@ifclasslater . 50:163, 50:171, 50:179
- \@ifclassloaded 50:155, 50:268, 50:277
- \@ifclasswith .. 50:221, 50:270, 50:279
- \@ifdefinable
..... 21:32, 21:35, 22:11, 23:3,
29:619, 06:84, 06:86, 06:130, 06:132,
06:250, 40:154, 43:7, 43:15, 43:22, 1287
- \@iffileonpath
20:487, 20:541, 20:563, 20:580, 20:601
- \@ifl@aded 50:155, 50:156,
50:157, 50:909, 50:1039, 50:1057, 1066

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpara.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=ltthyphen.dtx, 57=ltfinal.dtx

<code>\@ifl@t@r</code> .. 21:1560, 05:201, 05:207, 50:169, 50:172, 50:177, 50:180, 50:184, 50:188, 50:190, 50:201, 50:202, 50:766	43:21, 06:490, 47:40, 47:68, 47:85, 06:530, 06:531, 49:3, 49:7, 50:128, 50:153, 50:451, 50:457, 50:474, 50:486, 50:564, 50:598, 50:617, 50:911, <u>06:735</u> , 52:484, 07:3281, <i>1298</i>
<code>\@ifl@ter</code> .. 21:1605, 21:1606, 50:163, 50:164, <u>50:183</u> , 50:951, 50:1086	<code>\@ignore...</code> .. <i>1319</i>
<code>\@ifl@ter@@</code> .. 21:1605, 21:1606	<code>\@ignorefalse</code> 24:433, <u>37:4</u> , 37:255, 37:275, 37:293, 37:303, 37:339, 37:356, 37:372, 37:381, 45:384, <i>1283</i>
<code>\@ifnch</code> .. 05:49, 05:50, 06:782, <u>06:784</u> , 06:796	<code>\@ignoretrue</code> .. 18:201, 18:214, 24:428, <u>37:4</u> , 37:7, 38:348, 38:351, 38:384, 38:544, <i>1319</i>
<code>\@ifnextchar</code> .. 18:94, 18:558, 20:633, 22:13, 26:412, 05:45, 37:429, 38:355, 39:174, 40:9, 40:11, 40:18, 40:20, 40:26, 40:68, 40:160, 40:161, 40:167, 40:168, 40:175, 40:179, 40:224, 40:225, 40:231, 40:232, 40:237, 40:295, 40:303, 40:311, 40:318, 40:322, 40:425, 40:429, 40:433, 40:534, 40:539, 40:562, 40:569, 40:574, 41:57, 41:181, 41:203, 41:210, 42:23, 42:132, 42:143, 42:453, 43:3, 43:5, 43:32, 43:38, 45:27, 45:264, 45:324, 45:452, 45:519, 45:536, 47:3, 47:17, 47:29, 50:356, 50:370, 50:761, 50:785, 50:793, 50:800, 50:1212, 50:1215, 50:1347, 50:1350, <u>06:778</u> , 06:783, 06:797, 54:207, 54:2508, 01:82, <i>1327</i>	<code>\@iiminipage</code> .. 40:427, 40:431, 40:434, 40:435, <u>40:436</u>
<code>\@iforloop</code> .. 13:21, <u>13:22</u>	<code>\@iiiparbox</code> .. 40:305, 40:313, 40:320, 40:323, 40:324, <u>40:325</u> , 40:472
<code>\@ifpackage@later</code> .. 50:163, 50:170, 50:178, <i>1328</i>	<code>\@iiminipage</code> .. 40:430, <u>40:432</u>
<code>\@ifpackage@loaded</code> .. 50:155, 50:267, 50:276, 54:2492, <i>1041</i>	<code>\@iininput</code> .. 20:633, <u>20:634</u> , <i>469</i>
<code>\@ifpackage@with</code> .. 50:221, 50:269, 50:278, <i>1041</i>	<code>\@iiparbox</code> .. 40:319, <u>40:321</u>
<code>\@ifframebox</code> .. 40:238, 40:239, <u>40:240</u>	<code>\@iirsbox</code> .. 40:574, <u>40:583</u>
<code>\@ifframepicbox</code> .. 40:295, <u>40:296</u>	<code>\@imakebox</code> .. 40:26, <u>40:41</u> , 40:177
<code>\@ifstar</code> 18:60, 18:72, 18:377, 18:512, 22:155, 22:166, 24:269, 27:121, 06:73, 37:418, 37:425, 37:725, 37:734, 38:390, 41:56, 41:202, 41:209, 42:142, 42:604, 44:52, 44:142, 50:498, 50:540, <u>06:797</u> , 54:2356	<code>\@imakepicbox</code> .. 40:68, <u>40:69</u> , 40:182, 40:297
<code>\@ifundefin@d@i</code> .. 06:740, 06:741, 06:758, 06:761	<code>\@iminipage</code> .. 40:426, <u>40:428</u>
<code>\@ifundefin@d@ii</code> 06:740, 06:743, 06:746	<code>\@in@minipage@envtrue</code> .. 40:451
<code>\@ifundefined</code> .. 22:3, 22:7, 22:18, 22:100, 22:114, 22:116, 24:101, 24:200, 24:249, 26:425, 28:488, 05:132, 33:827, 35:85, 37:216, 37:233, 37:249, 37:270, 37:288, 06:127, 37:298, 06:134, 06:154, 06:161, 06:183, 06:194, 06:289, 06:295, 06:328, 06:334, 06:356, 06:362, 06:392, 06:409,	<code>\@include</code> .. 20:228, 20:271, 20:287, <u>20:291</u> , <i>459</i>
	<code>\@includeinreleasefalse</code> .. 03:74, 03:79, 03:133, 03:141, 06:873
	<code>\@includeinreleasetrue</code> .. 03:123
	<code>\@index</code> .. 46:18, <u>46:19</u> , 46:35
	<code>\@indexfile</code> .. 46:4, 46:5, 46:14
	<code>\@inlabel</code> .. <i>861</i>
	<code>\@inlabelfalse</code> .. <u>39:28</u> , 39:104, 39:215, 54:161, 54:188, <i>1294</i>
	<code>\@inlabeltrue</code> .. <u>39:28</u> , 39:209
	<code>\@inmatherr</code> .. 14:285, 39:112, 39:173, 42:604, <i>1301</i>
	<code>\@inmathwarn</code> .. <u>21:3</u>
	<code>\@inpenc@test</code> .. 57:356, 57:423
	<code>\@ininput</code> .. 20:34, 20:104, 20:162, 20:299, 20:361, 20:408, <u>20:672</u> , 44:152, 57:717, <i>1100</i>
	<code>\@ininput@</code> .. 20:319, 20:346, 20:379, 20:398, 20:423, <u>20:674</u> , 24:490, 47:51, <i>1299</i>
	<code>\@ininput@file@exists@with@hooks</code> .. <u>52:143</u> , <i>1342</i>
	<code>\@ininputcheck</code> .. 20:482, 20:483, 20:490, 20:536, 20:537, 20:544, 20:558, 20:559, 20:566, 20:588, 20:589, 20:592, 20:605, 20:606, 20:609, 01:175, 01:176, 01:179, 01:187, 05:38, 05:39, 05:42, 06:38,

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- 06:45, 50:1255, 50:1256, 50:1290,
50:1386, 50:1387, 50:1423, 50:1498,
50:1499, 50:1506, 01:54, 02:307, 1344
- \@insertfalse
54:1347, 54:1498, 54:1642, 54:1813,
54:1895, 54:1975, 54:2070, 54:2191
- \@inserttrue
54:1273, 54:1318, 54:1436,
54:1585, 54:1748, 54:2149, 54:2276
- \@invalidchar 14:290
- \@iparbox 40:304, 40:312, 40:317
- \@irsbox 40:562, 40:569, 40:574, 40:575
- \@isavebox 40:175, 40:176
- \@isavepicbox 40:180, 40:181
- \@ishortstack 42:133, 42:141
- \@istackcr 42:143, 42:144
- \@itabcr 41:57, 41:58
- \@item 39:174, 39:187
- \@itemdepth
.. 39:272, 39:274, 39:275, 39:276, 875
- \@itemfudge 41:38, 41:44, 41:82
- \@itemitem 39:276, 39:279
- \@itemlabel . 39:44, 39:96, 39:174, 1288
- \@itempenalty 18:17, 39:23, 39:206, 862
- \@itemspacing 875
- \@iwhiledim 13:7
- \@iwhilenum 13:3
- \@iwhilesw 13:10
- \@ixpt 24:815
- \@ixstackcr 42:142
- \@kernel@... 1044
- \@kernel@Ref 35:159, 35:164
- \@kernel@after@<hook> 220
- \@kernel@after@begindocument ...
09:78, 20:58, 20:76, 05:7, 51:238, 1344
- \@kernel@after@begindocument@before
..... 20:16, 20:76, 25:3251, 455
- \@kernel@after@enddocument
..... 05:1, 37:15, 37:81, 53:352
- \@kernel@after@enddocument@afterlastpage
... 05:1, 37:19, 37:85, 53:358, 08:455
- \@kernel@after@para@after
..... 16:8, 16:106, 414
- \@kernel@after@para@end
..... 16:8, 16:99, 414
- \@kernel@after@shipout@background
..... 53:72, 53:160, 1343
- \@kernel@after@shipout@lastpage .
..... 53:114,
53:119, 53:160, 53:378, 53:383, 1146
- \@kernel@before@<hook> 220
- \@kernel@before@begindocument ...
..... 20:56, 20:76, 05:7, 53:407, 1344
- \@kernel@before@enddocument
..... 37:13, 37:79, 37:180, 824
- \@kernel@before@insertmark
..... 48:237, 48:264, 1023
- \@kernel@before@para@before
..... 16:8, 16:21, 16:54, 414
- \@kernel@before@para@begin
..... 16:8, 16:30, 16:62, 413
- \@kernel@before@shipout@background
..... 53:68, 53:70, 53:160, 1134
- \@kernel@currentdata 35:171
- \@kernel@currpathstack
... 50:45, 50:47, 50:91, 50:123, 1046
- \@kernel@eqno .. 38:443, 38:445, 38:451
- \@kernel@leqno . 38:444, 38:446, 38:452
- \@kernel@make@file@csname
..... 52:287, 52:290,
52:310, 52:313, 52:331, 52:334, 52:363
- \@kernel@new@label@record@testdef
..... 36:149, 36:279, 37:25
- \@kernel@pageref 35:65, 35:69
- \@kernel@pageref@exp 35:70
- \@kernel@ref 35:64, 35:67, 35:159
- \@kernel@ref@exp 35:72
- \@kernel@refstepcounter
..... 35:136, 35:153, 43:31, 1356
- \@kernel@rename@newcommand
..... 06:297, 06:314,
06:351, 06:379, 06:384, 06:397, 93
- \@kernel@reserved@label@data ...
..... 35:100, 35:107
- \@kernel@sRef 35:161, 35:164
- \@kernel@spageref
... 35:26, 35:43, 35:60, 35:65, 35:69
- \@kernel@sref . 35:13, 35:25, 35:30,
35:42, 35:47, 35:59, 35:64, 35:67, 35:161
- \@killglue
.. 42:59, 42:73, 42:103, 42:115, 42:123
- \@kludgeins . 54:317, 54:318, 54:319,
54:321, 54:374, 54:375, 54:421,
54:422, 54:492, 54:514, 54:515,
54:516, 54:520, 54:535, 54:539,
54:549, 54:709, 54:731, 54:740,
54:747, 54:769, 54:770, 54:2352, 54:2383
- \@labels 39:27, 39:177,
39:178, 39:220, 39:237, 39:238, 861
- \@largefloatcheck
.. 45:192, 45:213, 45:238, 45:256, 1297
- \@lastchclass 41:262, 41:272,
41:273, 41:275, 41:283, 41:308,
41:322, 41:326, 41:335, 41:348, 41:349
- \@latex@error
.. 08:2940, 03:100, 03:159, 03:184,
03:195, 14:163, 14:217, 14:233,

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=ltthyphen.dtx, 57=ltfinal.dtx

14:239, 14:241, 14:244, 14:246,	50:1394, 50:1417, 50:1500, 50:1507,
14:248, 14:252, 14:254, 14:256,	50:1574, 50:1658, 53:79, 53:98,
14:259, 14:263, 14:268, 14:272,	53:369, 54:241, 54:273, 54:2327, 54:2606
14:274, 14:276, 14:278, 14:279,	<code>\@latexbug</code> 14:277 , 54:331 , 54:2313
14:281, 14:284, 14:288, 14:290,	<code>\@latexerr</code> 14:215 , 54:435 , 1298
17:10, 20:224, 20:268, 20:286,	<code>\@latexinfo</code> 1296
21:68, 21:102, 24:6, 24:29, 24:59,	<code>\@latexrelease@catcode@null</code>
24:103, 24:145, 24:250, 24:316, 07:6 , 07:3282
24:402, 26:106, 27:100, 27:111,	<code>\@lbibitem</code> 47:3 , 47:4
28:24, 28:69, 28:98, 28:118, 28:171,	<code>\@ldots</code> 30:515 , 30:517
28:202, 28:232, 28:264, 28:380,	<code>\@leftcolumn</code>
28:396, 28:494, 28:515, 28:567, 54:119 , 54:2802 , 54:2808 ,
28:617, 28:621, 28:663, 28:668,	54:2831 , 54:2851 , 54:2872 , 54:2881
28:723, 28:791, 28:797, 28:841,	<code>\@leftmark</code> 49:16 , 49:119
28:845, 28:849, 28:884, 28:888,	<code>\@let@token</code> 18:457 ,
28:892, 28:949, 28:959, 28:1044,	18:458 , 18:465 , 05:49 , 05:51 , 32:83 ,
28:1049, 28:1052, 28:1084, 28:1087,	32:96 , 38:256 , 38:258 , 38:261 ,
28:1160, 28:1163, 28:1166, 28:1233,	06:782 , 06:785 , 06:788 , 06:796 , 1297
28:1239, 29:50, 29:61, 29:83, 29:94,	<code>\@lign</code> 38:208 , 38:210
29:601, 29:721, 05:80, 32:143,	<code>\@linechar</code> 42:190 , 42:191 , 42:192 ,
37:250, 37:271, 37:289, 37:299,	42:196 , 42:197 , 42:199 , 42:204 ,
06:128, 06:155, 37:655, 37:672,	42:206 , 42:207 , 42:208 , 42:209 ,
37:686, 37:701, 37:713, 38:422,	42:211 , 42:215 , 42:216 , 42:219 ,
39:250, 06:290, 41:100, 41:109,	42:220 , 42:225 , 42:272 , 42:669 , 1327
06:329, 06:357, 44:31, 45:6, 45:83,	<code>\@linefnt</code> 42:124 , 42:127 , 42:190 ,
06:481, 47:71, 47:88, 06:522, 50:688,	42:265 , 42:273 , 42:304 , 42:307 , 42:676
50:739, 50:752, 50:781, 50:964,	<code>\@linelen</code> 42:164 , 42:165 , 42:176 ,
50:1012, 50:1059, 50:1171, 50:1188,	42:177 , 42:203 , 42:210 , 42:219 ,
50:1196, 50:1201, 50:1223, 50:1280,	42:221 , 42:226 , 42:227 , 42:228 ,
50:1358, 50:1413, 50:1757, 50:1787,	42:241 , 42:242 , 42:256 , 42:257 ,
54:232, 54:388, 54:2374, 54:2391	42:300 , 42:303 , 42:305 , 42:306 , 42:670
<code>\@latex@info</code> 14:163 ,	<code>\@list</code> 862
14:208 , 33:40 , 06:236 , 06:310 ,	<code>\@listctr</code> 39:233 , 39:256 , 47:9 , 861
06:348 , 06:376 , 06:835 , 52:487 , 403	<code>\@listdepth</code> 39:23 ,
<code>\@latex@info@no@line</code>	39:35 , 39:38 , 39:43 , 39:99 , 40:456 , 861
. 14:163 , 14:209 , 53:89 , 54:775 , 1301	<code>\@listfiles</code>
<code>\@latex@note</code> 14:190 , 24:34 20:62 , 20:131 , 20:186 , 20:753 , 20:772
<code>\@latex@note@no@line</code>	<code>\@listi</code> 862
14:190 , 50:1237 , 50:1257 , 50:1263 , 1345	<code>\@listii</code> 862
<code>\@latex@warning</code>	<code>\@listvi</code> 862
. 14:163 , 14:215 , 20:344 ,	<code>\@lnbk</code> 1321
20:396 , 21:73 , 22:56 , 22:74 , 35:18 ,	<code>\@loadwithoptions</code>
35:35 , 35:52 , 36:211 , 36:221 , 36:227 , 50:694 , 50:716 , 50:726 , 50:735
40:51 , 40:252 , 40:347 , 40:447 ,	<code>\@lowpenalty</code> 18:118 , 57:3
42:449 , 45:260 , 47:42 , 47:69 , 47:86 ,	<code>\@ltab</code> 41:71 , 41:106
50:1319 , 50:1326 , 50:1452 , 50:1458 ,	<code>\@ltxnomath</code> 1296
50:1541 , 50:1547 , 53:174 , 54:2540	<code>\@m</code> 18:303 , 18:336 , 18:504 ,
<code>\@latex@warning@no@line</code>	18:509 , 20:45 , 20:115 , 20:173 , 39:80 ,
. 14:163 , 14:216 , 20:19 , 20:89 ,	42:213 , 42:217 , 47:37 , 02:21 , 02:429 ,
20:146 , 20:739 , 05:98 , 35:8 , 35:88 ,	02:431 , 02:432 , 02:498 , 02:499 , 1313
35:89 , 36:133 , 36:134 , 37:62 , 37:69 ,	<code>\@mainaux</code>
37:127 , 37:134 , 37:166 , 06:202 ,	20:3 , 20:37 , 20:38 , 20:107 , 20:108 ,
44:32 , 50:352 , 50:366 , 50:767 ,	20:165 , 20:166 , 20:217 , 20:299 ,
50:952 , 50:1087 , 50:1239 , 50:1388 ,	

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpara.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- 20:342, 20:361, 20:394, 20:408,
 20:433, 37:22, 37:88, 37:150, 1307
 \@make@normalcolbox
 54:495, 54:501, 54:750
 \@make@specialcolbox
 54:493, 54:510, 54:751
 \@makebox 40:11, 40:20, 40:25
 \@makecaption 45:24, 1248
 \@makecol ... 54:259, 54:411, 54:458,
 54:482, 54:483, 54:688, 54:690, 1178
 \@makecol@handlesplitfootnotes ..
 54:585, 54:599
 \@makefcolumn
 ... 54:391, 54:392, 54:400, 54:402,
 54:438, 54:440, 54:448, 54:450,
 54:2760, 54:2762, 54:2778, 54:2779
 \@makefnmark 45:400, 45:532
 \@makefntext 40:489,
 40:506, 40:523, 45:476, 45:494, 45:512
 \@makeother 01:110,
 24:503, 24:504, 24:505, 24:506,
 24:507, 24:508, 24:509, 24:510,
 24:511, 24:512, 24:513, 37:541,
 37:562, 37:707, 37:722, 37:732,
 50:438, 50:439, 50:1301, 50:1434,
 50:1523, 06:800, 06:801, 01:60, 01:81
 \@makepicbox
 ... 40:10, 40:19, 40:67, 42:366, 42:424
 \@makespecialcolbox
 54:710, 54:723, 1286
 \@marbox
 ... 45:320, 45:322, 45:326, 45:330,
 45:331, 45:379, 54:2312, 54:2322,
 54:2325, 54:2333, 54:2335, 54:2336,
 54:2338, 54:2339, 54:2340, 54:2349
 \@marginparreset 45:343, 45:361, 45:370
 \@markright . 49:52, 49:75, 49:98, 49:121
 \@math@level 24:404,
 24:422, 24:429, 24:434, 28:288, 546
 \@maxdepth 20:60,
 20:129, 20:184, 54:87, 54:497,
 54:577, 54:695, 54:720, 57:152, 1293
 \@maxtab 41:2, 41:94, 1297
 \@medpenalty 18:119, 57:3
 \@meta@family@list
 29:117, 29:142, 29:154, 29:243, 29:507
 \@midlist
 ... 54:62, 54:489, 54:490, 54:706,
 54:707, 54:1301, 54:1303, 54:1416,
 54:1565, 54:1724, 54:2456, 54:2484
 \@minipage... 1319
 \@minipagefalse
 39:212, 40:420, 40:422,
 40:469, 45:187, 45:250, 45:345, 45:363
 \@minipagerestore 40:457, 40:459
 \@minipagetrue 40:421, 45:186
 \@minus 06:26, 54:2930,
 54:2931, 54:2932, 54:2935, 54:2936
 \@missing@onefilewithoptions ...
 50:926, 50:982, 50:1100
 \@missingfile@area
 20:646, 20:687, 20:700, 50:984
 \@missingfile@base
 20:646, 20:688, 20:701, 50:985
 \@missingfile@ext
 20:646, 20:689, 20:702, 50:986
 \@missingfileerror 20:641, 20:656,
 20:666, 20:675, 50:983, 50:1080, 470
 \@mkboth 49:11, 49:13
 \@mklab 39:45, 39:171
 \@mkpream . 41:189, 41:234, 41:262, 1253
 \@mparbottom 45:387,
 45:388, 54:116, 54:476, 54:2323,
 54:2331, 54:2332, 54:2333, 54:2334
 \@mpargs 40:440, 40:472
 \@mparswitchfalse 54:100
 \@mpfn
 40:454, 45:452, 45:457, 45:541, 45:545
 \@mpfootins 40:446, 40:463,
 40:464, 40:467, 40:473, 40:480,
 40:481, 40:498, 40:499, 40:515, 40:516
 \@mpfootnotetext 40:455, 40:475
 \@mplistdepth 40:456, 40:473
 \@multicnt 41:370,
 41:372, 41:373, 41:374, 41:381,
 41:382, 41:383, 42:103, 42:104,
 42:106, 42:115, 42:116, 42:118,
 42:666, 42:703, 42:705, 42:706,
 42:707, 42:709, 42:710, 42:716,
 42:722, 42:733, 42:737, 42:763,
 42:765, 42:767, 42:769, 42:770,
 42:774, 42:778, 42:789, 42:793, 1316
 \@multiplelabels ... 20:33, 20:103,
 20:161, 35:87, 35:93, 36:132, 37:60,
 37:66, 37:125, 37:131, 37:164, 37:170
 \@multiput 42:86, 42:95, 42:98
 \@multispan 41:371, 41:375, 41:379
 \@mypkg@name 1091
 \@mypkg@other@name 1091
 \@namedef 20:446,
 24:136, 24:137, 24:161, 24:203, 25:6,
 25:1436, 25:2913, 26:419, 28:417,
 28:421, 28:430, 33:57, 33:830,
 06:50, 35:90, 37:287, 37:347, 37:364,
 37:378, 37:584, 37:593, 38:426,
 38:427, 41:175, 43:12, 43:13, 43:18,
 43:19, 43:23, 43:24, 43:25, 50:1214,
 50:1349, 52:482, 57:480, 57:481, 1285

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx,
 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
 56=ltthyphen.dtx, 57=ltfinal.dtx

- `\@nameuse` ... 20:336, 20:392, 20:431,
 20:445, [06:51](#), 43:23, 49:5, 50:920,
 52:486, 54:811, 54:885, 54:943, [1301](#)
`\@nbitem` 39:199, [39:252](#)
`\@ne` [02:16](#), [391](#)
`\@needsf@rmat` .. 50:762, 50:765, 50:770
`\@needsformat` .. 50:750, 50:760, 50:764
`\@negargfalse` 42:186
`\@negargtrue` 42:185
`\@newcommand` [06:79](#), [06:80](#)
`\@newctr` 22:13, [22:17](#), 43:8
`\@newenv` [06:150](#), [06:151](#), [06:160](#)
`\@newenva` [06:149](#), [06:148](#)
`\@newenvb` [06:151](#), [06:150](#)
`\@newfontswitch` [1295](#)
`\@newl@bel`
[35:84](#), [37:24](#), [37:90](#), [37:152](#), [47:10](#), [1313](#)
`\@newline` 18:95, [18:97](#)
`\@newlistfalse` [39:29](#), [39:33](#),
[39:108](#), [39:213](#), 54:802, 54:878, 54:936
`\@newlisttrue` [39:29](#), [39:33](#), 39:87
`\@next` 45:60, 45:129,
 45:319, 45:320, 54:5, 54:211, 54:309,
 54:1151, 54:1171, 54:2312, 02:275
`\@nextchar`
[41:269](#), [41:270](#), [41:330](#), [41:331](#), [41:332](#)
`\@nil` 03:13, 03:19, 03:83,
 03:104, 03:105, 03:117, 03:118,
 03:165, 03:166, 03:167, 13:13,
 13:19, 13:27, 19:14, 20:247, 20:248,
 20:249, 21:107, 01:145, 21:128,
 01:146, 21:1018, 21:1022, 21:1085,
 21:1097, 21:1099, 24:455, 24:466,
 24:582, 24:597, 24:701, 24:704,
 24:705, 24:713, 25:2819, 25:2821,
 25:2853, 25:2855, 25:3165, 25:3167,
 25:3191, 25:3193, 26:351, 26:352,
 26:354, 26:367, 26:373, 26:377,
 26:378, 26:414, 26:435, 26:440,
 26:520, 26:534, 27:26, 27:44, 27:53,
 27:57, 28:41, 28:587, 28:595, 28:628,
 28:1244, 28:1246, 32:58, 32:62,
 06:53, 06:54, 06:58, 06:63, 06:135,
 41:367, 41:368, 06:451, 06:452,
 50:90, 50:91, 50:97, 50:105, 50:108,
 50:115, 50:140, 50:191, 50:192,
 50:203, 50:204, 50:214, 50:215,
 50:217, 06:633, 50:447, 50:470,
 50:514, 50:520, 50:636, 50:656,
 50:660, 50:666, 50:670, 50:877,
 50:886, 50:903, 50:904, 50:1631,
 50:1636, 50:1639, 50:1641, 50:1642,
 50:1657, 50:1677, 50:1694, 50:1749,
 50:1771, 50:1795, 06:776, 06:777,
 52:168, 52:176, 52:369, 52:411, 52:413
`\@nmbulistfalse` 39:33, 39:46, 39:91, [1283](#)
`\@nmbulisttrue` 39:256
`\@nnil` [13:13](#), 13:20,
 13:21, 13:22, 13:28, 20:248, 20:249,
 24:277, 24:281, 24:282, 24:283,
 24:298, 26:180, 26:182, 26:346,
 26:348, 26:360, 26:362, 26:367,
 26:381, 26:383, 26:390, 26:401,
 26:402, 26:404, 26:435, 26:440,
 42:13, 06:429, 06:435, 50:822,
 50:823, 50:830, 50:850, 50:851, 50:858
`\@no@font@optfalse` [27:17](#), [27:129](#)
`\@no@lnbk` 18:9, 18:10, [18:41](#), [1321](#)
`\@no@pgbk` 18:7, 18:8, [18:33](#), [1314](#)
`\@nobreak` [1004](#)
`\@nobreak...` [1319](#)
`\@nobreakfalse`
 ... 18:121, 18:123, 39:224, 44:94,
 44:129, 44:157, 45:182, 54:163,
 54:190, 54:1425, 54:1574, 54:1735, [1320](#)
`\@nobreaktrue` .. 18:122, 44:126, 45:181
`\@nocnterr` [14:240](#), [1294](#)
`\@nocounterr` [14:240](#), 22:4,
 22:8, 22:18, 22:114, 22:116, 43:21, [1295](#)
`\@nodocument` [14:245](#),
 16:143, 20:68, 20:137, 20:192,
 20:296, 20:358, 37:241, 45:39,
 45:108, 54:154, 54:181, 54:210, [1321](#)
`\@noitemargfalse` [39:32](#), 39:231
`\@noitemargtrue` [39:32](#), 39:174
`\@noitemerr`
 ... [14:280](#), 18:264, 18:281, 18:351,
 18:373, 39:69, 39:81, 39:107, [1356](#)
`\@noitemerror` [1307](#)
`\@noligs` 37:542,
 37:563, 37:723, 37:733, [37:744](#), [1319](#)
`\@nolnbnk` [1321](#)
`\@nolnerr` [14:238](#),
 18:43, 18:114, 37:417, 37:424, [1297](#)
`\@nomath` [24:3](#), 24:400, 29:552, 29:587,
 29:593, 29:614, 29:616, 29:638, [1281](#)
`\@noparitemfalse` [39:30](#), 39:176
`\@noparitemtrue` [39:30](#), 39:66
`\@noparlistfalse` [39:31](#), 39:70
`\@noparlisttrue` [39:31](#), 39:67
`\@nopgbk` [1314](#)
`\@normalcr` .. [18:53](#), 18:93, 40:419, [1336](#)
`\@normalsize` 50:4, 50:5, [1295](#)
`\@normalsizecheck` [1294](#)
`\@noskipsec` [861](#)
`\@noskipsecfalse` 20:54,
 20:124, 20:179, 44:98, 54:156, 54:183

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
 06=ltdefns.dtx, 07=ltcmd, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpara.dtx,
 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
 26=ltfssstrc.dtx, 27=ltfsscscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
 56=ltthyphen.dtx, 57=ltfinal.dtx

- `\@noskipsectrue` [44:38](#), [44:95](#)
- `\@notdefinable` [14:232](#),
[06:136](#), [06:137](#), [06:141](#), [06:478](#), [06:519](#)
- `\@notprerr` [14:283](#), [20:66](#), [20:135](#), [20:190](#)
- `\@nthm` [43:3](#), [43:4](#)
- `\@nxttabmar`
. [41:11](#), [41:21](#), [41:23](#), [41:25](#), [41:75](#),
[41:111](#), [41:112](#), [41:118](#), [41:119](#), [1316](#)
- `\@obsoletefile` [20:738](#)
- `\@oddfont` [49:11](#), [49:14](#),
[49:15](#), [54:122](#), [54:816](#), [54:888](#), [54:947](#)
- `\@oddhead` [49:11](#),
[49:14](#), [54:121](#), [54:815](#), [54:888](#), [54:947](#)
- `\@onefilewithoptions` [50:814](#), [50:818](#),
[50:824](#), [50:842](#), [50:846](#), [50:852](#),
[50:869](#), [50:873](#), [50:879](#), [50:895](#),
[50:897](#), [50:980](#), [50:1047](#), [50:1600](#), [1107](#)
- `\@onefilewithoptions@clashchk`
. [50:912](#), [50:962](#)
- `\@onelevel@sanitize`
. [45:42](#), [45:111](#), [06:802](#), [1305](#)
- `\@onfilewithoptions` [1070](#)
- `\@onlypreamble`
. [08:2821](#), [20:197](#), [20:206](#), [20:243](#),
[20:740](#), [20:800](#), [21:41](#), [21:42](#), [21:79](#),
[21:80](#), [21:84](#), [21:143](#), [21:163](#), [21:207](#),
[21:208](#), [21:224](#), [24:18](#), [24:116](#),
[24:118](#), [24:124](#), [24:140](#), [24:168](#),
[24:183](#), [24:266](#), [24:271](#), [24:313](#),
[24:609](#), [26:420](#), [27:28](#), [27:36](#), [27:42](#),
[27:79](#), [27:83](#), [27:88](#), [27:93](#), [27:98](#),
[27:108](#), [27:126](#), [27:127](#), [27:128](#),
[27:134](#), [27:138](#), [27:142](#), [28:18](#),
[28:20](#), [28:45](#), [28:47](#), [28:108](#), [28:117](#),
[28:137](#), [28:410](#), [28:411](#), [28:424](#),
[28:470](#), [28:518](#), [28:530](#), [28:532](#),
[28:545](#), [28:570](#), [28:627](#), [28:629](#),
[28:671](#), [28:710](#), [28:726](#), [28:803](#),
[28:897](#), [28:906](#), [28:962](#), [28:965](#),
[28:968](#), [28:988](#), [28:1001](#), [28:1055](#),
[28:1090](#), [28:1101](#), [28:1115](#), [28:1169](#),
[28:1189](#), [28:1193](#), [28:1257](#), [32:140](#),
[32:141](#), [33:831](#), [35:92](#), [06:66](#), [06:188](#),
[06:190](#), [06:199](#), [06:207](#), [46:12](#),
[46:29](#), [47:64](#), [47:81](#), [48:16](#), [48:386](#),
[50:10](#), [50:13](#), [50:85](#), [50:112](#), [50:120](#),
[50:122](#), [50:154](#), [50:357](#), [50:420](#),
[50:426](#), [50:491](#), [50:494](#), [50:495](#),
[50:506](#), [50:507](#), [50:508](#), [50:535](#),
[50:541](#), [50:554](#), [50:591](#), [50:609](#),
[50:629](#), [50:649](#), [50:673](#), [50:678](#),
[50:682](#), [50:685](#), [50:693](#), [50:714](#),
[50:717](#), [50:727](#), [50:746](#), [50:759](#),
[50:764](#), [50:770](#), [50:798](#), [50:803](#),
[50:891](#), [50:980](#), [50:1105](#), [50:1114](#),
[50:1122](#), [50:1123](#), [50:1141](#), [50:1169](#),
[50:1178](#), [50:1185](#), [50:1186](#), [50:1194](#),
[50:1199](#), [50:1204](#), [50:1580](#), [50:1581](#),
[50:1582](#), [50:1583](#), [50:1585](#), [51:202](#), [1287](#)
- `\@opargbegintheorem` [43:45](#), [43:48](#)
- `\@opcol` [54:260](#), [54:268](#), [54:392](#),
[54:411](#), [54:440](#), [54:458](#), [54:463](#), [1030](#)
- `\@options` [50:628](#)
- `\@othm` [43:3](#), [43:20](#)
- `\@outerparskip`
. [39:1](#), [39:88](#), [39:117](#), [39:183](#), [39:253](#)
- `\@outputbox` [48:404](#),
[48:407](#), [48:411](#), [48:436](#), [48:439](#),
[48:443](#), [54:118](#), [54:485](#), [54:502](#),
[54:504](#), [54:505](#), [54:521](#), [54:524](#),
[54:529](#), [54:530](#), [54:537](#), [54:543](#),
[54:545](#), [54:558](#), [54:576](#), [54:578](#),
[54:579](#), [54:692](#), [54:694](#), [54:712](#),
[54:714](#), [54:715](#), [54:724](#), [54:726](#),
[54:727](#), [54:732](#), [54:734](#), [54:735](#),
[54:741](#), [54:743](#), [54:851](#), [54:914](#),
[54:973](#), [54:997](#), [54:1003](#), [54:1013](#),
[54:1014](#), [54:1037](#), [54:1044](#), [54:1130](#),
[54:1133](#), [54:1136](#), [54:1142](#), [54:1143](#),
[54:2802](#), [54:2806](#), [54:2812](#), [54:2831](#),
[54:2834](#), [54:2835](#), [54:2849](#), [54:2855](#),
[54:2872](#), [54:2878](#), [54:2887](#), [1181](#)
- `\@outputbox@append`
. [54:518](#), [54:561](#), [54:566](#), [54:575](#),
[54:587](#), [54:634](#), [54:636](#), [54:645](#),
[54:652](#), [54:658](#), [54:660](#), [54:755](#), [1170](#)
- `\@outputbox@appendfootnotes`
. [54:583](#),
[54:639](#), [54:643](#), [54:654](#), [54:663](#),
[54:667](#), [54:671](#), [54:677](#), [54:756](#), [1170](#)
- `\@outputbox@attachbottomfloats`
. [54:600](#), [54:759](#), [1170](#)
- `\@outputbox@attachfloats`
. [54:600](#), [54:640](#),
[54:647](#), [54:650](#), [54:662](#), [54:666](#),
[54:672](#), [54:678](#), [54:757](#), [54:992](#), [1170](#)
- `\@outputbox@attachtopyfloats`
. [54:600](#), [54:758](#), [1170](#)
- `\@outputbox@depth` [54:500](#),
[54:504](#), [54:506](#), [54:518](#), [54:578](#), [1182](#)
- `\@outputbox@reinsertbskip`
. [54:559](#), [54:565](#),
[54:637](#), [54:646](#), [54:653](#), [54:661](#),
[54:668](#), [54:673](#), [54:676](#), [54:753](#), [1170](#)
- `\@outputbox@removebskip`
. [54:486](#), [54:556](#), [54:752](#), [1181](#)
- `\@outputdblcol`
. [54:467](#), [54:2797](#), [54:2799](#),

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=ltthyphen.dtx, 57=ltfinal.dtx

- 54:2827, 54:2828, 54:2868, 54:2869
- `\@outputpage` 54:401, 54:450, 54:470,
54:789, 54:2815, 54:2820, 54:2858,
54:2862, 54:2894, 54:2902, 1186
- `\@oval` 42:453, 42:470
- `\@ovbtrue` 42:475, 42:503, 42:533
- `\@ovdx` 42:431, 42:486, 42:488, 42:494,
42:496, 42:515, 42:517, 42:525,
42:527, 42:542, 42:550, 42:552,
42:588, 42:591, 42:598, 42:600,
42:692, 42:693, 42:694, 42:695,
42:711, 42:712, 42:713, 42:716,
42:732, 42:752, 42:753, 42:754,
42:755, 42:771, 42:772, 42:774, 42:788
- `\@ovdy` 42:431, 42:487, 42:489, 42:495,
42:496, 42:516, 42:518, 42:526,
42:527, 42:543, 42:551, 42:552,
42:563, 42:568, 42:576, 42:580,
42:699, 42:700, 42:701, 42:702,
42:717, 42:718, 42:719, 42:722,
42:736, 42:759, 42:760, 42:761,
42:762, 42:775, 42:776, 42:778, 42:792
- `\@ovhlinefalse` 42:476, 42:504
- `\@ovhlinetrue` 42:456, 42:460,
42:464, 42:482, 42:488, 42:510, 42:517
- `\@ovhorz` 42:493, 42:494,
42:524, 42:525, 42:549, 42:550, 42:583
- `\@ovltrtrue` 42:475, 42:503, 42:533
- `\@ovri` 40:33, 42:431, 42:485, 42:514,
42:541, 42:563, 42:576, 42:592, 42:601
- `\@ovro` 42:431, 42:485, 42:494,
42:495, 42:514, 42:525, 42:526,
42:541, 42:550, 42:551, 42:562,
42:568, 42:575, 42:580, 42:587,
42:597, 42:613, 42:620, 42:629, 42:638
- `\@ovrtrue` 42:475, 42:503, 42:533
- `\@ovttrue` 42:475, 42:503, 42:533
- `\@ovvert` 42:491, 42:492,
42:520, 42:522, 42:545, 42:547, 42:556
- `\@ovvlinefalse` 42:476, 42:504
- `\@ovvlinetrue`
42:459, 42:481, 42:489, 42:509, 42:518
- `\@ovxx` 42:431, 42:479,
42:481, 42:482, 42:486, 42:492,
42:493, 42:507, 42:509, 42:510,
42:515, 42:522, 42:524, 42:536,
42:538, 42:542, 42:547, 42:549,
42:587, 42:597, 42:689, 42:690,
42:691, 42:695, 42:704, 42:705,
42:714, 42:715, 42:716, 42:731,
42:749, 42:750, 42:751, 42:755,
42:764, 42:765, 42:773, 42:774, 42:787
- `\@ovyy` 42:431,
42:480, 42:481, 42:482, 42:487,
42:494, 42:508, 42:509, 42:510,
42:516, 42:525, 42:537, 42:538,
42:543, 42:550, 42:560, 42:573,
42:696, 42:697, 42:698, 42:702,
42:704, 42:720, 42:721, 42:722,
42:735, 42:756, 42:757, 42:758,
42:762, 42:764, 42:777, 42:778, 42:791
- `\@p@pfilename` 50:90,
50:97, 50:105, 50:108, 50:115, 50:120
- `\@p@pfilepath` ... 50:91, 50:138, 50:147
- `\@p@pfilepath@aux` 50:139, 50:140, 50:148
- `\@p@pagedp` ... 54:115, 54:306, 54:311,
54:1365, 54:1663, 54:2341, 54:2351
- `\@pageht`
... 54:114, 54:307, 54:311, 54:313,
54:314, 54:315, 54:319, 54:1364,
54:1515, 54:1662, 54:2324, 54:2331
- `\@par` 15:3, 15:5
- `\@parboxrestore`
..... 40:334, 40:361, 40:419,
40:453, 40:484, 40:502, 40:519,
45:19, 45:100, 45:169, 45:342,
45:360, 45:470, 45:489, 45:507,
54:217, 54:803, 54:879, 54:937, 1320
- `\@parboxto` 40:325, 1328
- `\@parmoderr` 14:273, 45:58, 45:127, 45:316
- `\@parse@version` 03:83,
03:104, 03:105, 03:117, 03:118,
03:165, 03:166, 03:167, 50:197,
50:203, 50:204, 50:214, 50:1642,
50:1657, 50:1694, 50:1771, 50:1795
- `\@parse@version@`
..... 50:191, 50:192, 50:197, 50:209
- `\@parse@version@dash`
..... 50:215, 50:217, 1333
- `\@partaux` 20:3, 20:223,
20:267, 20:285, 20:309, 20:311,
20:312, 20:332, 20:371, 20:373,
20:374, 20:388, 20:418, 20:420,
20:421, 20:427, 20:439, 20:441, 20:444
- `\@partlist`
... 20:232, 20:236, 20:237, 20:239,
20:263, 20:282, 20:305, 20:367, 20:414
- `\@partswfalse` 20:6
- `\@partswtrue` ... 20:231, 20:261, 20:281
- `\@pass@options`
... 50:443, 50:445, 50:465, 50:466,
50:468, 50:481, 50:482, 50:484,
50:491, 50:492, 50:493, 50:991, 50:1073
- `\@pboxswfalse` .. 40:332, 40:359, 40:438
- `\@pboxswtrue` 40:342, 40:369
- `\@pdef` 1290
- `\@penup` 38:199, 38:200

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpara.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- \@percentchar ... 50:1293, 50:1295,
50:1297, 50:1299, 50:1426, 50:1428,
50:1430, 50:1432, 50:1515, 50:1517,
50:1519, 50:1521, 50:1569, 01:90
- \@pgbk 1314
- \@picbox
. 42:6, 42:31, 42:43, 42:51, 42:52, 1282
- \@picht 42:6, 42:29, 42:42, 42:51
- \@picture 42:23, 42:24
- \@picture@warn
..... 42:223, 42:441, 42:445, 42:449
- \@pkgetension 1290
- \@pkgextension 50:31, 50:155, 50:163,
50:221, 50:492, 50:684, 50:687,
50:726, 50:735, 50:825, 50:853,
50:880, 50:1002, 50:1029, 50:1150,
50:1180, 50:1614, 52:482, 52:492, 1068
- \@plus 18:532,
06:26, 44:16, 44:231, 44:254,
49:127, 54:2930, 54:2931, 54:2932,
54:2935, 54:2936, 54:2940, 54:2941,
54:2942, 54:2946, 54:2947, 54:2948
- \@pnumwidth 44:243, 44:266
- \@popfilename
..... 50:33, 50:960, 50:1095, 1045
- \@pr@videpackage
..... 50:356, 50:370, 50:373,
50:394, 50:396, 50:407, 50:409, 50:420
- \@preamble
... 41:190, 41:192, 41:200, 41:237,
41:256, 41:258, 41:259, 41:263,
41:278, 41:296, 41:297, 41:334, 55:190
- \@preamblecmds 20:67, 20:136,
20:191, 06:66, 50:1809, 50:1810, 1290
- \@preammerr 14:260, 41:199, 41:274, 41:355
- \@process@pti@ns
..... 50:553, 50:571, 50:588,
50:595, 50:596, 50:609, 50:613, 50:615
- \@process@ptions 50:540, 50:542, 50:554
- \@protect@... 1301
- \@protected@testopt
..... 06:89, 06:101, 06:628, 06:640
- \@protecteddef 1296
- \@providesfile
..... 50:430, 57:710, 01:82, 01:83
- \@ptionlist
... 50:152, 50:224, 50:539, 50:968,
50:974, 50:1063, 50:1069, 50:1181
- \@pushfilename
..... 50:33, 50:905, 50:1050, 1044
- \@put 42:452,
42:496, 42:527, 42:552, 42:620, 42:638
- \@qend 14:236, 06:136, 06:776
- \@qrelax 06:137, 06:776
- \@raw@classoptionslist
..... 50:11, 50:812, 51:100, 51:142
- \@rc@ifdefinable
..... 21:32, 06:130, 06:132, 06:250
- \@reargdef 06:122, 1290
- \@refundefined 20:55, 20:125, 20:180,
35:3, 37:58, 37:123, 37:162, 1316
- \@reinserts 54:325, 54:328, 54:767
- \@remove@eq@value 50:509, 50:576, 50:636
- \@remove@tlig 21:1018, 21:1026
- \@remove@tlig@ 21:1018, 21:1019
- \@remove@tlig@@ 21:1019, 21:1022
- \@remove@tlig@@@ 21:1032, 21:1051
- \@removeelement . 13:32, 50:635, 50:644
- \@removefromreset 22:96,
22:123, 22:124, 22:134, 22:157, 22:160
- \@renewfontswitch 1293
- \@reqcolroom
..... 54:1364, 54:1365, 54:1368,
54:1370, 54:1371, 54:1376, 54:1377,
54:1381, 54:1383, 54:1411, 54:1412,
54:1515, 54:1518, 54:1520, 54:1521,
54:1526, 54:1530, 54:1532, 54:1560,
54:1561, 54:1662, 54:1663, 54:1667,
54:1670, 54:1671, 54:1676, 54:1683,
54:1685, 54:1717, 54:1718, 54:1829,
54:1831, 54:1833, 54:1834, 54:1837,
54:1839, 54:1911, 54:1913, 54:1915,
54:1918, 54:1920, 54:1993, 54:1996,
54:1999, 54:2004, 54:2006, 54:2520,
54:2651, 54:2652, 54:2657, 54:2660,
54:2702, 54:2707, 54:2710, 1359
- \@reserveda 620
- \@reset@ptions ... 50:924, 50:961,
50:989, 50:1054, 50:1096, 50:1106
- \@resetactivechars
..... 54:774, 54:800, 54:876, 54:934
- \@reset@hfps 54:1480,
54:1625, 54:1793, 54:2597, 1297
- \@resetprotect 1311
- \@restorepar . 15:5, 18:388, 18:404,
18:422, 18:438, 39:127, 39:137, 1313
- \@reversemarginfalse .. 45:388, 54:99
- \@reversemargintrue 45:387
- \@rightmark 49:16, 49:120
- \@rightskip 37:456,
37:475, 37:493, 39:75, 40:392, 40:413
- \@rjfieldfalse 41:34, 41:77
- \@rjfieldtrue 41:125
- \@rmfamilyhook
..... 29:430, 29:452, 29:464, 29:491
- \@roman 22:190, 22:196
- \@rsbox 40:562, 40:569, 40:573
- \@rtab 41:71, 41:86

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpara.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- [\@rule](#) 40:534, 40:539, [40:543](#)
- [\@sanitize](#)
..... 46:7, 46:18, 46:24, 46:35, [06:800](#)
- [\@savebox](#) 40:161, 40:168, [40:174](#)
- [\@savemarbox](#)
..... 45:326, 45:327, 45:330, [45:333](#)
- [\@savepicbox](#) ... 40:161, 40:168, [40:178](#)
- [\@savsf](#) 17:28, 17:30,
17:50, 17:52, [18:124](#), 18:130, 18:139,
18:158, 18:172, 18:184, 18:198, 18:212
- [\@savsk](#) [18:124](#), 18:129, 18:140,
18:159, 18:173, 18:185, 18:199, 18:213
- [\@scolelt](#) 54:1062, [54:1127](#)
- [\@sdblcolelt](#) 54:1079, 54:1099, [54:1128](#)
- [\@secntformat](#) ... 44:60, [44:111](#), [1284](#)
- [\@secondoffive](#) 35:24, 35:27, 35:41, 35:58
- [\@secondoftwo](#) 03:87, 03:185,
17:5, 20:509, 20:539, 20:585, 20:602,
21:149, 22:243, 22:248, 24:221,
24:241, 05:36, 29:513, 05:40, 33:800,
33:850, 33:866, 35:44, 35:80, [06:212](#),
06:457, 49:17, 06:586, 50:159,
50:193, 50:205, 50:238, 50:256,
50:305, 50:335, 06:637, 06:745,
06:752, 50:1800, 06:772, 06:819,
54:614, 54:619, 54:626, 57:716, 01:72
- [\@secpenalty](#) 18:18, [44:36](#), 44:50
- [\@sect](#) 44:54, [44:55](#)
- [\@seqncr](#) [38:425](#)
- [\@set@curr@file@aux](#) [52:363](#), [1351](#)
- [\@setckpt](#) 20:439,
[20:446](#), 37:23, 37:89, 37:151, [1284](#)
- [\@setfloattypecounts](#) ... 54:1348,
54:1499, 54:1643, 54:1814, 54:1896,
54:1976, 54:2071, 54:2192, [54:2544](#)
- [\@setfontsize](#) [29:638](#)
- [\@setfps](#) [45:34](#)
- [\@setfpsbit](#) ... 45:73, 45:75, 45:77,
45:85, 45:143, 45:146, 45:149, [54:2588](#)
- [\@setmarks](#) .. 54:2841, 54:2843, 54:2857
- [\@setminipage](#) 40:458,
45:21, 45:177, [45:185](#), 45:376, [1308](#)
- [\@setminpage](#) [1321](#)
- [\@setnobreak](#) [45:179](#), [45:375](#), [1308](#)
- [\@setpar](#) [15:3](#), 39:78
- [\@setprotect](#) [1311](#)
- [\@setref](#) [35:10](#)
- [\@setsize](#) [29:638](#)
- [\@settab](#) 41:71, [41:93](#)
- [\@settodim](#) [23:17](#)
- [\@settopoint](#) [23:35](#)
- [\@setupverbvisiblespace](#) .. 37:585,
37:597, [37:601](#), 37:662, 37:676, 37:690
- [\@setupverbvisibletab](#) 37:613, [37:634](#)
- [\@sffamilyhook](#)
..... [29:430](#), 29:457, 29:465, 29:492
- [\@sharp](#) 41:196, 41:235, 41:265, 41:280,
41:281, 41:301, 41:303, 41:305, 41:333
- [\@shipoutsetup](#) [54:789](#)
- [\@shortstack](#) 42:132, 42:133
- [\@show@...](#) [101](#)
- [\@show@DeclareRobustCommand](#) ...
..... 06:554, 06:568, [06:613](#), 06:667, 06:670
- [\@show@DeclareRobustCommand@env](#) .
..... 06:691, [06:693](#)
- [\@show@environment](#) ... 06:685, 06:686
- [\@show@environment@begin](#)
..... 06:694, 06:700, 06:708, 06:713
- [\@show@environment@end](#)
..... 06:698, 06:707, [06:712](#), [105](#)
- [\@show@environment@end@aux](#)
..... 06:716, 06:718
- [\@show@macro](#) [06:613](#), 06:703
- [\@show@newcommand](#) . 06:555, 06:568,
06:621, [06:648](#), 06:667, 06:673, [102](#)
- [\@show@newcommand@aux](#)
..... 06:648, 06:677, 06:695, [103](#)
- [\@show@newcommand@env](#) [06:693](#), 06:692
- [\@show@nonstop](#) ... 06:703, [06:724](#), [106](#)
- [\@show@normalenv](#) 06:689, [06:712](#)
- [\@show@tokens](#)
..... 06:653, [06:658](#), 06:702, 06:719
- [\@show@timeout](#)
..... 06:697, 06:702, [06:724](#), [106](#)
- [\@showcommandlisthook](#)
..... 06:551, [06:553](#), 06:561, 07:1472, [100](#)
- [\@showenvironmentlisthook](#)
..... 06:688, [06:690](#), 07:1474, [1351](#)
- [\@skipping@modulefalse](#)
..... 03:154, 03:173, 03:200
- [\@skipping@moduletrue](#) 03:153, 03:177
- [\@sline](#) . 42:167, 42:179, [42:184](#), 42:269
- [\@slowromancap](#) 22:197, [22:198](#)
- [\@spaces](#) [14:218](#)
- [\@specialoutput](#) [54:254](#)
- [\@specialpagefalse](#)
..... 54:95, 54:810, 54:885, 54:943
- [\@specialpagetrue](#) 49:9
- [\@specialstyle](#)
..... 49:9, 54:811, 54:885, 54:943
- [\@sptoken](#) 06:785, [06:795](#)
- [\@sqrt](#) [38:355](#)
- [\@ssect](#) 44:53, [44:112](#)
- [\@stackcr](#) 42:139, [42:142](#)
- [\@star@or@long](#) [06:72](#), 06:77, 06:124,
06:146, 06:152, 06:178, 06:187, 06:233
- [\@startcolumn](#) . 54:261, 54:268, [54:1049](#)

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpara.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

<code>\@startdblcolumn</code>	40:589, 40:590, 40:591, 42:304,
..... 54:1049 , 54:2819 , 54:2822 ,	42:305, 42:447, 42:448, 42:485,
54:2861 , 54:2863 , 54:2900 , 54:2906	42:490, 42:495, 42:496, 42:514,
<code>\@startfield</code>	42:519, 42:526, 42:527, 42:541,
41:46 , 41:92 , 41:104 , 41:125 , 41:133	42:544, 42:551, 42:552, 42:613,
<code>\@startline</code>	42:614, 42:619, 42:620, 42:629,
41:20 , 41:57 , 41:62 , 41:68 , 41:83 , 41:154	42:630, 42:637, 42:638, 42:723,
<code>\@startpagehook</code>	42:741, 42:779, 42:797, 44:138,
1287	
<code>\@startpbox</code>	44:139, 45:322, 45:380, 54:303,
41:236 , 41:266 , 41:332 , 41:384 , 41:386	54:375, 54:380, 54:381, 54:422,
<code>\@startsection</code>	54:427, 54:428, 54:549, 54:747,
44:39	
<code>\@starttoc</code>	54:836, 54:848, 54:849, 54:904,
44:149 , 1246	
<code>\@stopfield</code> 41:32 , 41:48 , 41:86 , 41:93 ,	54:911, 54:912, 54:963, 54:970,
41:125 , 41:127 , 41:140 , 41:142 , 41:154	54:971, 54:995, 54:999, 54:1011,
<code>\@stopline</code>	54:1017, 54:1024, 54:1025, 54:1026,
41:30 , 41:56 , 41:85	54:1027, 54:1031, 54:1039, 1283
<code>\@stpelt</code>	
22:22 , 22:25	
<code>\@string@makeletter</code>	06:806 , 109
<code>\@strip@args</code>	21:94
<code>\@strip@tex@ext</code>	20:227 ,
20:237 , 20:239 , 20:244 , 20:274 , 459	
<code>\@strip@tex@ext@aux</code> ..	20:244 , 20:275
<code>\@svector</code>	42:244 , 42:259 , 42:269
<code>\@sverb</code>	
37:646 , 37:725 , 37:734 , 37:737 , 1284	
<code>\@svsec</code> 44:57 , 44:60 , 44:66 , 44:78 , 1284	
<code>\@svsechd</code>	44:76 , 44:101 , 44:121
<code>\@swaptwoargs</code>	20:619 ,
20:621 , 50:934 , 52:153 , 52:167 , 52:187	
<code>\@sxverbatim</code> ...	37:518 , 37:586 , 37:593
<code>\@tabacckludge</code>	
..... 21:243 , 21:245 , 21:474 ,	
21:475 , 33:157 , 33:164 , 33:166 , 1317	
<code>\@tabacol</code>	41:178 , 41:258
<code>\@tabarray</code>	41:170 , 41:180 , 41:181
<code>\@tabclassiv</code>	41:180 , 41:330
<code>\@tabclassz</code>	41:179 , 41:282
<code>\@tabcr</code>	41:56 , 41:73
<code>\@tabfbox</code>	41:16 , 41:80 , 41:82
<code>\@tablab</code>	41:72 , 41:126
<code>\@tabminus</code>	41:72 , 41:117
<code>\@tabplus</code>	41:72 , 41:110
<code>\@tabpush</code>	41:11 ,
41:77 , 41:85 , 41:140 , 41:143 , 41:145	
<code>\@tabrj</code>	41:72 , 41:124
<code>\@tabular</code>	41:174 , 41:177 , 41:178
<code>\@tabularcr</code>	41:180 , 41:208
<code>\@tempa</code>	1308
<code>\@tempboxa</code>	12:13 , 21:87 , 23:21 ,
23:22 , 23:23 , 23:28 , 23:29 , 39:236 ,	
39:242 , 39:243 , 39:245 , 40:29 , 40:30 ,	
40:31 , 40:32 , 40:37 , 40:38 , 40:39 ,	
40:40 , 40:214 , 40:248 , 40:267 ,	
40:276 , 40:286 , 40:441 , 40:472 ,	
40:579 , 40:580 , 40:581 , 40:588 ,	
<code>\@tempcnta</code>	12:7 ,
28:970 , 28:971 , 28:972 , 28:973 ,	
28:977 , 41:242 , 41:243 , 41:244 ,	
41:245 , 42:187 , 42:188 , 42:214 ,	
42:215 , 42:216 , 42:229 , 42:230 ,	
42:231 , 42:232 , 42:239 , 42:240 ,	
42:254 , 42:255 , 42:270 , 42:271 ,	
42:276 , 42:278 , 42:279 , 42:280 ,	
42:281 , 42:282 , 42:285 , 42:287 ,	
42:288 , 42:289 , 42:290 , 42:291 ,	
42:292 , 42:293 , 42:294 , 42:295 ,	
42:296 , 42:335 , 42:336 , 42:337 ,	
42:338 , 42:339 , 42:360 , 42:361 ,	
42:362 , 42:363 , 42:364 , 42:365 ,	
42:392 , 42:393 , 42:394 , 42:395 ,	
42:396 , 42:414 , 42:416 , 42:418 ,	
42:419 , 42:421 , 42:423 , 42:438 ,	
42:439 , 42:440 , 42:442 , 42:444 ,	
42:446 , 42:448 , 42:561 , 42:566 ,	
42:574 , 42:578 , 42:615 , 42:616 ,	
42:617 , 42:618 , 42:631 , 42:632 ,	
42:634 , 42:635 , 42:657 , 42:658 ,	
42:659 , 42:660 , 42:661 , 42:662 ,	
42:710 , 42:730 , 42:770 , 42:786 ,	
45:62 , 45:68 , 45:70 , 45:79 , 45:80 ,	
45:90 , 45:91 , 45:131 , 45:137 , 45:139 ,	
45:152 , 45:153 , 45:159 , 45:160 ,	
50:1241 , 50:1243 , 50:1244 , 50:1245 ,	
50:1372 , 50:1374 , 50:1375 , 50:1376 ,	
50:1486 , 50:1488 , 50:1489 , 50:1490 ,	
54:12 , 54:14 , 54:16 , 54:1208 ,	
54:1209 , 54:1210 , 54:1211 , 54:1231 ,	
54:1232 , 54:1233 , 54:1234 , 54:1256 ,	
54:1259 , 54:1292 , 54:1295 , 54:1407 ,	
54:1556 , 54:1713 , 54:2074 , 54:2077 ,	
54:2195 , 54:2198 , 54:2313 , 54:2315 ,	
54:2318 , 54:2320 , 54:2322 , 54:2344 ,	
54:2578 , 54:2579 , 54:2583 , 54:2589 ,	

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- 54:2593, 57:227, 57:232, 57:233,
57:234, 57:364, 57:366, 57:367,
57:368, 57:375, 57:377, 57:378,
57:379, 57:385, 57:387, 57:388,
57:389, 57:396, 57:398, 57:399,
57:400, 57:426, 57:428, 57:429,
57:430, 57:452, 57:454, 57:455,
57:456, 57:462, 57:464, 57:465,
57:466, 57:490, 57:495, 57:496, 57:497
- `\@tempcntb` . . . [12:7](#), 28:971, 28:975,
28:977, 42:279, 42:280, 42:281,
42:283, 42:284, 42:285, 42:561,
42:562, 42:566, 42:567, 42:574,
42:575, 42:578, 42:579, 45:88, 45:89,
45:90, 45:157, 45:158, 45:159, 54:13,
54:16, 54:17, 54:2589, 54:2590,
54:2591, 57:228, 57:232, 57:491, 57:495
- `\@tempdima` [12:10](#),
24:282, 24:287, 38:186, 38:189,
38:195, 40:47, 40:48, 40:62, 40:63,
40:247, 40:248, 40:266, 40:267,
40:274, 40:275, 40:276, 40:278,
40:333, 40:334, 40:360, 40:361,
40:439, 40:443, 40:546, 40:549,
40:550, 40:577, 40:579, 40:585,
40:588, 41:35, 41:36, 41:37, 41:88,
41:89, 41:90, 41:91, 41:218, 41:219,
42:210, 42:211, 42:213, 42:214,
42:215, 42:216, 42:217, 42:218,
42:437, 42:438, 42:439, 42:448,
42:486, 42:487, 42:491, 42:492,
42:515, 42:516, 42:520, 42:522,
42:542, 42:543, 42:545, 42:547,
42:616, 42:618, 42:633, 42:635,
42:636, 42:656, 42:657, 42:658,
44:236, 44:237, 44:259, 44:260,
44:273, 45:196, 45:198, 45:218,
45:220, 45:258, 45:259, 45:260,
54:227, 54:228, 54:229, 54:519,
54:521, 54:522, 54:527, 54:531,
54:535, 54:540, 54:544, 54:730,
54:732, 54:733, 54:736, 54:740,
54:742, 54:1191, 54:1194, 54:1214,
54:1224, 54:1236, 54:1246, 54:2137,
54:2138, 54:2141, 54:2142, 54:2262,
54:2263, 54:2267, 54:2268, 54:2323,
54:2324, 54:2325, 54:2326, 54:2329,
54:2332, 54:2335, 54:2337, 54:2751,
54:2752, 54:2754, 54:2755, [1284](#)
- `\@tempdimb` [12:10](#), 24:283,
24:288, 24:721, 24:725, 26:180,
26:181, 26:438, 26:461, 26:462,
26:471, 26:472, 26:476, 26:494,
26:497, 26:500, 26:502, 40:336,
40:337, 40:363, 40:364, 40:547,
40:550, 40:578, 40:580, 40:586,
40:589, 42:211, 42:212, 42:357,
42:359, 42:362, 42:365, 42:481,
42:483, 42:484, 42:509, 42:512,
42:513, 42:538, 42:539, 42:540,
42:611, 42:612, 42:621, 42:627,
42:628, 42:639, 42:647, 42:648,
42:653, 54:1214, 54:1215, 54:1216,
54:1217, 54:1224, 54:1236, 54:1237,
54:1238, 54:1239, 54:1246, [1284](#)
- `\@tempdimc` . . [12:10](#), 26:455, 26:456,
26:458, 26:459, 26:461, 26:462,
40:74, 40:75, 40:85, 40:86, 40:548,
40:549, 40:550, 42:30, 42:31, 42:32,
42:33, 42:34, 42:35, 42:60, 42:61,
42:63, 42:64, 42:332, 42:333, 42:334
- `\@tempdimx` [1284](#)
- `\@tempskipa`
[12:14](#), 18:45, 18:48, 18:49, 18:300,
18:307, 18:309, 18:312, 18:333,
18:340, 18:342, 18:345, 26:182,
26:183, 39:116, 39:117, 39:118,
39:181, 39:183, 39:184, 39:185,
39:253, 39:254, 39:255, 44:42, 44:44,
44:45, 44:50, 44:62, 44:63, 44:88,
44:89, 44:91, 44:103, 44:104, 44:113,
44:114, 54:562, 54:563, 54:566,
54:2372, 54:2373, 54:2375, 54:2383
- `\@tempskipb` [12:14](#),
18:221, 18:223, 18:225, 18:228,
18:230, 18:244, 18:259, 18:276,
18:298, 18:300, 18:301, 18:305,
18:307, 18:309, 18:310, 18:331,
18:333, 18:334, 18:338, 18:340,
18:342, 18:343, 18:365, 18:368, [442](#)
- `\@tempswa` [1280](#)
- `\@tempswafalse` . . . 20:303, 20:365,
20:412, 21:1559, 24:95, 24:771,
28:482, 28:557, 28:631, 28:712,
28:1232, 28:1238, 37:26, 37:91,
37:153, 37:533, 37:554, 47:17, 47:29,
50:1214, 50:1230, 50:1349, 50:1365,
52:286, 52:309, 52:330, 54:1262,
54:1298, 54:2080, 54:2201, 02:262, 01:62
- `\@tempswatru` 20:301, 20:306,
20:363, 20:368, 20:410, 20:415,
21:1562, 21:1563, 24:98, 24:772,
24:773, 24:776, 24:779, 28:485,
28:560, 28:634, 28:715, 28:1195,
36:152, 37:196, 37:538, 37:559,
47:17, 47:29, 50:1211, 50:1346,
52:286, 52:309, 52:330, 54:2082,
54:2105, 54:2203, 54:2228, 54:2662,

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpara.dtx,
16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
56=ltthyphen.dtx, 57=ltfinal.dtx

- 54:2679, 54:2712, 54:2729, 02:268, 01:63
- `\@temptokena` [12:16](#), 37:208,
37:212, 37:221, 37:234, 37:235,
49:42, 49:46, 49:53, 49:56, 49:68,
49:69, 49:76, 49:77, 49:91, 49:92,
49:99, 49:100, 49:122, 49:123, [826](#)
- `\@test...` [1319](#)
- `\@test@opt` [1315](#)
- `\@testdef` .. [37:24](#), [37:90](#), [37:152](#), [37:194](#)
- `\@testfalse` 54:8, 54:10, 54:11
- `\@testfp`
54:1156, 54:1176, 54:1212, 54:1235,
[54:2581](#), 54:2765, 54:2782, [1307](#)
- `\@testopt`
18:7, 18:8, 18:9, 18:10, 06:33, 06:79,
[06:99](#), 06:103, 06:148, 38:396, [1315](#)
- `\@testpach` 41:270, [41:348](#)
- `\@testtrue`
.... 54:9, 54:17, 54:354, 54:1159,
54:1178, 54:1218, 54:1240, 54:2585
- `\@testwrongwidth`
[54:343](#), 54:1157, 54:1213, 54:1387,
54:1536, 54:1846, 54:1927, 54:2131
- `\@text@composite`
..... [21:94](#), 21:1092, 21:1097, [1309](#)
- `\@text@composite@x` [21:94](#)
- `\@textbottom` 49:127, 49:129,
54:507, 54:532, 54:546, 54:557,
54:611, 54:717, 54:737, 54:744, [54:772](#)
- `\@textfloatsheight` 54:476,
54:1361, 54:1363, 54:1414, 54:1415,
54:1420, 54:1512, 54:1514, 54:1563,
54:1564, 54:1569, 54:1659, 54:1661,
54:1721, 54:1723, 54:1729, [54:2520](#)
- `\@textmin` 45:285,
45:286, 45:299, 45:300, 54:110,
54:1363, 54:1367, 54:1370, 54:1371,
54:1514, 54:1517, 54:1520, 54:1521,
54:1661, 54:1666, 54:1670, 54:1671,
54:1833, 54:1915, 54:1999, 54:2098,
54:2100, 54:2116, 54:2220, 54:2222,
54:2240, 54:2638, 54:2640, 54:2642
- `\@textsubscript` 45:424, [45:432](#)
- `\@textsuperscript`
..... 45:401, 45:403, [45:404](#), [1312](#)
- `\@texttop` 49:127,
49:129, 54:503, 54:713, 54:725, [54:772](#)
- `\@tf@r` 13:25, 13:26, [1293](#)
- `\@tfor` [13:25](#), 20:586, 20:603,
20:755, 32:88, 40:78, 40:97, 41:268,
42:477, 42:505, 42:534, 45:63, 45:132
- `\@tforloop` 13:27, 13:28, 13:30
- `\@thanks` 44:11, [44:34](#)
- `\@thefnmark` 40:487, 40:504,
40:521, 45:400, 45:401, 45:453,
45:458, 45:473, 45:491, 45:509,
45:521, 45:526, 45:537, 45:542, 45:553
- `\@thefoot` 54:122,
54:816, 54:820, 54:858, 54:888,
54:891, 54:918, 54:947, 54:950, 54:977
- `\@thehead` 54:121,
54:815, 54:819, 54:843, 54:888,
54:890, 54:908, 54:947, 54:949, 54:967
- `\@themargin` 54:70,
54:817, 54:821, 54:835, 54:889,
54:891, 54:903, 54:948, 54:950, 54:962
- `\@themark` 49:41, 49:42, 49:52, 49:53,
49:67, 49:68, 49:75, 49:76, 49:90,
49:91, 49:98, 49:99, 49:121, 49:123
- `\@thirdofthree` 21:215, [06:216](#)
- `\@thm` 43:12, 43:18, 43:24, [43:26](#)
- `\@thmcounter` 43:11, 43:17, [43:46](#)
- `\@thmcountersep` 43:10, [43:46](#)
- `\@title` 44:7, [44:31](#)
- `\@tocrmarg` 44:232, 44:255
- `\@toodeep` . [14:253](#), 39:36, 39:263, 39:274
- `\@toplist` 54:60, 54:382,
54:383, 54:429, 54:430, 54:601,
54:605, 54:996, 54:1006, 54:1007,
54:1299, 54:1311, 54:2454, 54:2482
- `\@topnewpage` [54:197](#), [1296](#)
- `\@topnum` .. 45:271, 54:103, 54:1296,
54:1297, 54:1311, 54:1315, 54:1878,
54:1883, 54:1959, 54:1964, 54:2051,
54:2058, 54:2445, 54:2473, 54:2514
- `\@toproom` 45:273, 54:104,
54:1299, 54:1311, 54:2446, 54:2474
- `\@topsep` ... [39:1](#), 39:71, 39:73, 39:202
- `\@topsepadd`
.... [39:1](#), 39:59, 39:61, 39:71, 39:124
- `\@totalleftmargin`
37:529, 37:551, [39:9](#), 39:53, 39:54,
40:391, 40:412, 41:35, 41:76, 41:81, [861](#)
- `\@trivlist` 39:48, [39:57](#), 39:92
- `\@tryfcolumn` ... 54:1052, [54:1072](#),
54:1090, [54:1106](#), 54:2766, 54:2783
- `\@trylist` 54:1115,
54:1118, 54:1151, 54:1171, 54:1193
- `\@ttfamilyhook`
..... [29:430](#), 29:462, 29:466, 29:493
- `\@twoclasseserror` 50:675, [50:1200](#)
- `\@twocolumnfalse` 54:97, 54:145
- `\@twocolumntrue` 54:204
- `\@twoloadclasserror`
..... 50:959, 50:1094, [50:1195](#)
- `\@twosidefalse` 54:98
- `\@typein` 06:32, 06:33, 06:40, 06:48, [1315](#)

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

<code>\@typeset@protect</code>	29:118, 29:119, 05:4, 29:232, 29:233, 29:234, 29:235, 29:236, 29:237, 29:238, 29:239, 29:240, 29:241, 29:243, 29:244, 29:245, 01:226,
..... 21:44, 21:50, 21:230, 21:238, 29:639, 06:102, 37:280, 37:317, 06:255, 06:262, 06:264, 57:358, 1307	05:13, 05:14, 29:305, 05:15, 29:306, 05:16, 29:473, 29:491, 29:492, 29:493, 29:535, 29:536, 29:537, 29:538, 29:539, 29:579, 29:580, 29:581, 29:582, 29:583, 29:584, 29:595, 29:717, 30:15, 30:55, 30:62, 30:77, 05:150, 32:37, 32:38, 05:151, 32:39, 05:152, 05:153, 05:154, 05:155, 32:122, 33:123, 05:172, 05:173, 05:174, 05:175, 33:331, 05:195, 05:196, 33:581, 33:584, 33:585, 33:615, 33:616, 33:617, 33:620, 33:621, 33:622, 33:623, 33:624, 33:625, 33:630, 33:631, 33:632, 33:633, 33:638, 33:639, 33:640, 33:641, 33:644, 33:645, 33:646, 33:648, 33:649, 33:654, 33:655, 33:656, 33:657, 33:658, 33:659, 33:660, 33:661, 33:662, 33:663, 33:664, 33:665, 33:666, 33:667, 33:668, 33:669, 33:671, 33:672, 33:674, 33:675, 33:676, 33:677, 33:678, 33:679, 33:680, 33:681, 33:682, 33:683, 33:685, 33:686, 33:687, 33:688, 33:689, 33:690, 33:691, 33:692, 33:693, 33:695, 33:696, 33:697, 33:698, 33:699, 33:700, 33:701, 33:702, 33:703, 33:704, 33:705, 33:706, 33:707, 33:708, 33:709, 33:710, 33:711, 33:712, 33:713, 33:714, 33:715, 33:716, 33:717, 33:718, 33:719, 33:720, 33:721, 33:726, 33:727, 33:729, 33:730, 33:731, 33:732, 33:733, 33:734, 33:735, 33:736, 33:738, 33:739, 33:741, 33:742, 33:744, 33:745, 33:746, 33:747, 33:749, 33:750, 33:751, 33:753, 33:755, 33:756, 33:757, 33:758, 33:759, 33:760, 33:761, 33:763, 33:764, 33:765, 33:766, 33:767, 33:768, 33:769, 33:770, 33:771, 06:34, 35:169, 35:170, 35:171, 35:191, 35:207, 35:208, 36:267, 36:268, 36:270, 36:271, 36:272, 36:274, 36:275, 36:276, 36:278, 36:279, 37:176, 37:228, 37:229, 37:230, 37:231, 37:306, 37:385, 37:405, 37:406, 37:407, 37:408, 37:515, 37:577, 37:595,
<code>\@uclclist</code>	
21:1528, 21:1529, 21:1586, 57:571, 1325	
<code>\@undefind</code>	
42:57, 42:71, 42:82, 42:91, 42:162, 42:174, 42:237, 42:252, 42:313, 42:371	
<code>\@undefined</code>	
01:92, 02:582, 01:93, 01:94, 02:650, 02:690, 02:705, 03:53, 03:62, 10:203, 10:204, 10:205, 10:207, 10:208, 10:209, 10:211, 10:212, 10:214, 10:215, 10:216, 10:217, 10:218, 10:219, 10:220, 10:221, 10:222, 01:115, 14:28, 01:123, 04:2, 04:15, 04:16, 04:17, 04:29, 04:30, 16:170, 16:171, 16:172, 16:173, 16:174, 17:61, 18:90, 01:131, 04:74, 04:84, 18:497, 18:546, 18:552, 20:61, 20:62, 20:130, 20:131, 20:185, 20:186, 20:274, 01:138, 20:275, 20:477, 20:478, 20:479, 20:484, 20:538, 20:551, 20:560, 20:575, 20:630, 04:173, 04:189, 04:197, 04:205, 21:213, 21:215, 21:353, 21:355, 21:357, 21:359, 21:361, 21:363, 21:365, 21:367, 21:369, 21:371, 21:390, 21:392, 21:394, 21:479, 04:237, 04:238, 04:239, 04:240, 04:241, 04:242, 04:243, 04:244, 04:245, 04:246, 04:247, 04:248, 04:249, 04:250, 04:251, 04:252, 04:253, 04:254, 04:255, 21:721, 21:724, 04:261, 22:113, 22:257, 24:531, 24:571, 24:633, 24:666, 24:730, 24:737, 01:189, 01:193, 25:2776, 25:2799, 25:2808, 25:2897, 25:2898, 25:2899, 25:2900, 25:2901, 25:2902, 25:2903, 25:2904, 25:2905, 25:2906, 25:2917, 25:2922, 25:2927, 25:2934, 25:2935, 25:2936, 25:2937, 25:2938, 25:2939, 25:2940, 25:3154, 25:3234, 25:3236, 25:3237, 25:3239, 25:3240, 25:3241, 25:3243, 26:553, 26:554, 27:4, 27:5, 27:6, 27:7, 27:8, 27:9, 27:10, 27:11, 27:12, 27:13, 27:14, 27:15, 27:16, 27:17, 27:18, 27:19, 27:20, 28:142, 28:525, 28:577, 01:219, 28:828, 28:935, 29:36, 29:59, 29:72, 29:92, 29:105, 29:106, 29:107, 29:108, 29:109, 29:110, 29:111, 29:112, 29:113, 29:114, 29:115, 29:117,	

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpara.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=ltthyphen.dtx, 57=ltfinal.dtx

37:596, 37:597, 37:598, 37:631,	02:149, 02:184, 02:189, 02:222,
37:643, 37:692, 38:236, 38:246,	02:223, 02:246, 02:256, 01:52, 01:53,
38:247, 38:283, 38:286, 38:329,	02:291, 02:353, 02:363, 02:365,
38:342, 38:436, 06:235, 40:21,	02:475, 02:476, 02:486, 02:487, <i>1334</i>
40:169, 40:233, 40:314, 40:540,	<code>\undefinedfonterror</code> <i>1297</i>
40:570, 06:319, 06:320, 06:351,	<code>\unexpandable@protect</code>
42:18, 06:379, 06:383, 06:384, 20:211, <u>06:232</u> ,
42:466, 42:467, 06:401, 06:415,	06:267, 06:273, 06:278, 41:264, <i>1306</i>
44:203, 44:204, 44:205, 44:206,	<code>\unknownoptionerror</code>
44:222, 44:270, 45:5, 06:462, 06:463, 50:1110, <u>50:1170</u> , 50:1183
06:464, 06:465, 06:466, 06:467,	<code>\unknownversion</code> <i>1286</i>
06:468, 01:310, 01:311, 45:429,	<code>\unprocessedoptions</code> 50:625,
45:448, 06:504, 45:562, 06:505,	50:734, 50:981, 50:1020, 50:1021,
06:506, 06:507, 06:508, 06:509,	50:1081, 50:1085, 50:1185, 51:88, <i>1068</i>
47:30, 47:53, 06:540, 06:541, 06:542,	<code>\unused</code> 14:15, 14:32, 14:59,
06:560, 06:561, 50:4, 50:25, 50:146,	06:10, 06:17, 50:1578, <u>02:307</u> , <i>1344</i>
50:147, 50:148, 50:209, 50:416,	<code>\unusedoptionlist</code>
50:526, 50:576, 06:668, 06:669,	20:18, 20:20, 20:88, 20:90, 20:145,
06:670, 06:671, 06:672, 06:673,	20:147, <u>50:12</u> , 50:518, 50:519,
06:675, 06:676, 06:677, 50:981,	50:529, 50:530, 50:637, 50:645,
50:1019, 50:1021, 50:1027, 50:1084,	51:130, 51:131, 51:137, 51:164, <i>1285</i>
50:1099, 50:1100, 50:1115, 50:1254,	<code>\upline</code> 42:297, <u>42:298</u> , 42:304
50:1332, 50:1335, 50:1385, 50:1464,	<code>\upordown</code>
50:1467, 50:1477, 50:1478, 50:1479,	42:195, 42:196, 42:204, 42:225, 42:273
50:1480, 50:1481, 50:1482, 50:1483,	<code>\upvector</code> 42:268, <u>42:304</u>
06:732, 50:1554, 50:1557, 50:1573,	<code>\use@option</code> . 50:549, 50:565, 50:583,
50:1593, 50:1600, 06:747, 06:754,	50:599, 50:601, 50:618, 50:620, <u>50:630</u>
52:18, 52:19, 52:20, 52:21, 52:147,	<code>\use@text@encoding</code>
52:189, 52:190, 52:197, 52:198, <u>21:168</u> , 33:16, 33:1102
52:199, 52:320, 52:341, 52:355,	<code>\vbsphack</code> <u>18:220</u>
52:359, 52:360, 06:828, 06:829,	<code>\@verb</code> 37:725, 37:734, <u>37:737</u>
06:830, 52:492, 52:498, 52:499,	<code>\@verbatim</code> <u>37:523</u> , 37:569, 37:584, 37:593
52:500, 06:839, 53:438, 53:439,	<code>\@verbvisiblespacebox</code> ... <u>37:583</u> ,
53:440, 53:441, 53:442, 53:444,	37:598, 37:610, 37:611, 37:624, 37:625
53:445, 53:446, 53:453, 53:454,	<code>\@verreq</code> 30:468, 30:469
53:456, 53:458, 53:459, 53:460,	<code>\@viiipt</code> <u>24:814</u>
53:463, 53:486, 54:32, 54:366,	<code>\@viipt</code> <u>24:813</u>
54:367, 54:750, 54:751, 54:752,	<code>\@vipt</code> <u>24:812</u>
54:753, 54:755, 54:756, 54:757,	<code>\@vline</code> 42:166, 42:178, <u>42:297</u>
54:758, 54:759, 54:761, 54:762,	<code>\@vobeyspaces</code> <u>37:499</u> , 37:569,
54:763, 54:2424, 54:2425, 54:2426,	37:586, 37:662, 37:676, 37:690, 37:737
54:2427, 54:2532, 02:65, 57:10,	<code>\@vobeytabs</code> <u>37:499</u>
57:18, 57:39, 57:54, 57:73, 57:82,	<code>\@vpt</code> <u>24:811</u>
57:89, 57:98, 57:108, 57:160, 57:161,	<code>\@vspace</code> <u>18:377</u>
57:270, 57:283, 57:296, 57:316,	<code>\@vspace@calcify</code>
57:341, 57:351, 57:352, 57:353,	... <u>18:80</u> , 18:102, 18:242, 18:257,
57:382, 57:384, 57:423, 57:424,	18:384, 18:389, 18:399, 18:407,
57:443, 57:444, 57:445, 57:446,	37:435, 38:404, 41:62, 41:224, 42:148
57:447, 57:448, 57:449, 57:450,	<code>\@vspacer</code> <u>18:377</u>
57:451, 57:468, 57:484, 57:485,	<code>\@vtryfc</code> 54:1121, <u>54:1129</u>
57:486, 57:529, 57:530, 57:664,	<code>\@vvector</code> 42:243, 42:258, <u>42:268</u>
57:699, 57:700, 57:701, 57:702,	<code>\@warning</code> <u>14:215</u>
57:703, 02:81, 02:105, 02:106,	<code>\@wckptelt</code> 20:440, <u>20:443</u>
02:121, 02:122, 02:127, 02:136,	<code>\@whiledim</code> <u>13:7</u> , 42:123, 42:203

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpara.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=ltthyphen.dtx, 57=ltfinal.dtx

- `\@whilenoop` [13:3](#), [1314](#)
- `\@whilenum` [13:3](#),
[41:244](#), [42:104](#), [42:116](#), [42:336](#),
[42:338](#), [42:361](#), [42:364](#), [42:393](#),
[42:395](#), [42:416](#), [42:421](#), [42:730](#), [42:786](#)
- `\@whilesw` [13:10](#),
[54:262](#), [54:392](#), [54:401](#), [54:439](#),
[54:449](#), [54:2820](#), [54:2862](#), [54:2901](#)
- `\@whileswnoop` [13:10](#), [1314](#)
- `\@wholewidth`
... [40:199](#), [40:201](#), [40:202](#), [40:204](#),
[40:206](#), [40:207](#), [40:208](#), [40:209](#), [42:2](#),
[42:126](#), [42:129](#), [42:131](#), [42:299](#),
[42:302](#), [42:350](#), [42:359](#), [42:406](#),
[42:413](#), [42:564](#), [42:577](#), [42:589](#),
[42:599](#), [42:678](#), [42:679](#), [42:727](#), [42:783](#)
- `\@width` [18:528](#), [21:309](#),
[21:314](#), [26:193](#), [30:629](#), [06:26](#),
[40:204](#), [40:206](#), [40:282](#), [40:289](#),
[40:550](#), [40:603](#), [40:610](#), [41:188](#),
[41:219](#), [41:347](#), [41:366](#), [42:227](#),
[42:299](#), [42:302](#), [42:325](#), [42:333](#),
[42:350](#), [42:359](#), [42:384](#), [42:392](#),
[42:406](#), [42:413](#), [42:564](#), [42:577](#),
[42:727](#), [42:783](#), [45:395](#), [54:2351](#),
[54:2810](#), [54:2853](#), [54:2884](#), [02:505](#), [1310](#)
- `\@wrglossary` [46:25](#), [46:30](#), [1306](#)
- `\@wrindex` [46:8](#), [46:13](#), [1306](#)
- `\@writeckpt`
..... [20:330](#), [20:386](#), [20:425](#), [20:437](#)
- `\@writefile` [20:32](#), [20:102](#),
[20:160](#), [37:215](#), [44:193](#), [44:201](#), [825](#)
- `\@writesetup` [54:789](#)
- `\@wrong@font@char`
... [21:179](#), [24:634](#), [24:668](#), [24:681](#), [1317](#)
- `\@wtryfc` [54:1131](#), [54:1141](#)
- `\@x@protect` [06:105](#), [06:254](#)
- `\@x@sf` [45:531](#), [45:533](#)
- `\@xDeclareMathDelimiter`
..... [28:1000](#), [28:1056](#)
- `\@xaddvskip`
..... [18:220](#), [18:245](#), [18:260](#), [18:277](#)
- `\@xarg` [42:163](#),
[42:166](#), [42:175](#), [42:178](#), [42:185](#),
[42:189](#), [42:190](#), [42:226](#), [42:228](#),
[42:238](#), [42:239](#), [42:243](#), [42:253](#),
[42:254](#), [42:258](#), [42:266](#), [42:274](#), [42:663](#)
- `\@xargarraycr` .. [41:205](#), [41:214](#), [41:218](#)
- `\@xargdef` [06:80](#)
- `\@xarraycr` [41:202](#), [41:203](#)
- `\@xbitor` [54:11](#), [54:13](#)
- `\@xcentercr` [37:418](#), [37:425](#), [37:429](#)
- `\@xdblarg` [06:798](#)
- `\@xdblfloat` [45:264](#), [1305](#)
- `\@xdim` . [42:84](#), [42:93](#), [42:105](#), [42:107](#),
[42:117](#), [42:119](#), [42:667](#), [42:731](#),
[42:732](#), [42:733](#), [42:734](#), [42:740](#),
[42:787](#), [42:788](#), [42:789](#), [42:790](#), [42:796](#)
- `\@xeqncr` [38:388](#)
- `\@exexnoop` [41:238](#), [41:248](#)
- `\@exexpast` [41:239](#), [41:240](#)
- `\@xfloat` [45:28](#), [45:29](#), [45:34](#), [45:266](#), [1306](#)
- `\@xfootnote` [45:452](#), [45:455](#), [1283](#)
- `\@xfootnotemark` . [45:519](#), [45:523](#), [1283](#)
- `\@xfootnotenext` [45:536](#), [45:539](#)
- `\@xfootnotetext` [1283](#)
- `\@xhline` [41:360](#), [41:361](#)
- `\@xifnch` [06:786](#), [06:796](#)
- `\@xiipt` . [24:818](#), [30:175](#), [30:177](#), [30:178](#)
- `\@xipt` [24:817](#), [30:174](#)
- `\@xivpt` [24:819](#), [30:176](#), [30:178](#)
- `\@xmpar` [45:324](#), [45:325](#)
- `\@xnewline`
..... [18:61](#), [18:62](#), [18:73](#), [18:74](#), [18:94](#)
- `\@xnext` [54:6](#), [54:7](#)
- `\@xnthm` [43:5](#), [43:6](#)
- `\@xobeysp` [18:467](#), [18:492](#),
[37:504](#), [37:514](#), [37:518](#), [37:607](#),
[37:611](#), [37:621](#), [37:625](#), [37:638](#), [447](#)
- `\@xobeytab` [18:488](#), [37:506](#), [37:638](#)
- `\@xprocess@options` [50:540](#),
[50:557](#), [50:559](#), [50:575](#), [50:577](#), [50:591](#)
- `\@xpt` ... [24:816](#), [30:173](#), [30:176](#), [30:177](#)
- `\@xsect` [44:86](#), [44:87](#), [44:123](#)
- `\@xtabcr` [41:56](#), [41:57](#)
- `\@xtabularcr` [41:209](#), [41:210](#)
- `\@xthm` [43:32](#), [43:38](#), [43:41](#)
- `\@xtryfc` [54:1118](#), [54:1146](#)
- `\@xtypein` [06:33](#), [06:35](#), [06:42](#)
- `\@xverbatim` [37:518](#), [37:569](#)
- `\@xviipt` [24:820](#), [30:177](#), [30:179](#)
- `\@xxDeclareMathDelimiter`
..... [28:985](#), [28:989](#)
- `\@xxpt` [24:821](#), [30:178](#), [30:179](#)
- `\@xxvpt` [24:822](#), [30:179](#)
- `\@xxxii` [12:2](#), [21:445](#), [21:447](#),
[45:89](#), [45:158](#), [54:1153](#), [54:1154](#),
[54:1173](#), [54:1174](#), [54:1209](#), [54:1210](#),
[54:1232](#), [54:1233](#), [54:2547](#), [1317](#)
- `\@xympar` [45:328](#), [45:332](#), [45:378](#)
- `\@yarg` [42:163](#), [42:167](#),
[42:175](#), [42:179](#), [42:185](#), [42:186](#),
[42:195](#), [42:238](#), [42:244](#), [42:253](#),
[42:259](#), [42:268](#), [42:270](#), [42:297](#), [42:663](#)
- `\@yargarraycr` .. [41:206](#), [41:216](#), [41:220](#)
- `\@yargdf` [06:107](#)
- `\@yargdef`
... [06:84](#), [06:94](#), [06:107](#), [06:123](#), [1319](#)

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=ltthyphen.dtx, 57=ltfinal.dtx

<code>\@ydim</code>	42:85, 42:94, 42:105, 42:108, 42:117, 42:119, 42:667, 42:735, 42:736, 42:737, 42:738, 42:739, 42:791, 42:792, 42:793, 42:794, 42:795	<code>\bfseries@rm@kernel</code>	29:109, 29:132, 29:208, 29:235, 702
<code>\@yeqncr</code>	38:388	<code>\bfseries@sf</code>	29:107, 29:129, 29:212, 29:215, 29:233, 29:260, 29:273, 29:316, 29:321, 29:357, 696
<code>\@ympar</code>	45:324, 45:329	<code>\bfseries@sf@kernel</code>	29:110, 29:133, 29:212, 29:236
<code>\@ynthm</code>	43:5, 43:14	<code>\bfseries@tt</code>	29:108, 29:129, 29:216, 29:219, 29:234, 29:261, 29:274, 29:317, 29:322, 29:358, 702
<code>\@ythm</code>	43:32, 43:38, 43:41	<code>\bfseries@tt@kernel</code>	29:111, 29:134, 29:216, 29:237
<code>\@ytryfc</code>	54:1164, 54:1183, 54:1187	<code>\bm@b</code>	40:37
<code>\@yyarg</code>	42:185, 42:186, 42:187, 42:190, 42:274, 42:663	<code>\bm@c</code>	40:37, 40:50, 40:251, 40:346
<code>\@ztryfc</code>	54:1192, 54:1203	<code>\bm@l</code>	40:37
<code>\@@end</code>	225	<code>\bm@r</code>	40:37
<code>\accent@spacefactor</code>	21:88, 21:91, 21:92, 1325	<code>\bm@s</code>	40:37
<code>\active@math@prime</code>	38:253, 38:254, 54:787, 1325	<code>\bm@t</code>	40:37
<code>\add@accent</code>	21:83, 21:85, 1318	<code>\botmark</code>	1010
<code>\add@percent@to@temptokena</code>	37:202, 37:218, 37:220, 37:229, 826	<code>\bx@</code>	1224
<code>\add@unicode@accent</code>	21:1061, 21:1075	<code>\bx@A</code>	54:26, 54:53
<code>\addto@hook</code>	24:153, 24:155, 24:810, 28:454, 28:590, 28:594, 28:611, 28:735, 28:741, 28:749, 28:765, 28:768, 28:771, 28:1204, 28:1211, 28:1214, 1281	<code>\bx@AA</code>	54:36
<code>\AddToHook</code>	309	<code>\bx@B</code>	54:26, 54:53
<code>\AddToHookNext</code>	309	<code>\bx@BB</code>	54:36
<code>\AddToHookNextWithArguments</code>	309	<code>\bx@C</code>	54:26, 54:53
<code>\AddToHookWithArguments</code>	309	<code>\bx@CC</code>	54:36
<code>\alloc@</code>	04:22, 04:26, 04:38, 24:15, 02:90, 02:91, 02:92, 02:93, 02:94, 02:95, 02:96, 02:97, 02:98, 02:99, 02:226, 1340	<code>\bx@D</code>	54:26, 54:53
<code>\alpha@elt</code>	28:46, 28:458, 28:685, 28:787, 28:1203, 28:1204	<code>\bx@DD</code>	54:36
<code>\alpha@list</code>	28:42, 28:44, 28:467, 28:673, 28:685, 28:730, 28:785, 28:786, 28:1199, 28:1205, 28:1206	<code>\bx@E</code>	54:26, 54:53
<code>\apptocmd</code>	310	<code>\bx@EE</code>	54:36
<code>\atveryend@DEPRECATED</code>	52:580, 52:582	<code>\bx@F</code>	54:27, 54:54
<code>\best@size</code>	26:439, 26:463, 26:469, 26:475	<code>\bx@FF</code>	54:37
<code>\bf@def@ult</code>	29:398	<code>\bx@G</code>	54:27, 54:54
<code>\bfdef@ult</code>	29:201, 29:272, 29:273, 29:274, 29:315, 29:316, 29:317, 29:402, 29:443, 622	<code>\bx@GG</code>	54:37
<code>\bfdefault@previous</code>	29:268, 29:271, 29:311, 29:314, 30:117, 30:127, 1341	<code>\bx@H</code>	54:27, 54:54
<code>\bfseries@..</code>	699	<code>\bx@HH</code>	54:37
<code>\bfseries@previous</code>	1341	<code>\bx@I</code>	54:27, 54:54
<code>\bfseries@rm</code>	29:106, 29:129, 29:208, 29:211, 29:232, 29:259, 29:272, 29:315, 29:320, 29:356, 702	<code>\bx@II</code>	54:37
		<code>\bx@J</code>	54:27, 54:54
		<code>\bx@JJ</code>	54:37
		<code>\bx@K</code>	54:28, 54:55
		<code>\bx@KK</code>	54:38
		<code>\bx@L</code>	54:28, 54:55
		<code>\bx@LL</code>	54:38
		<code>\bx@M</code>	54:28, 54:55
		<code>\bx@MM</code>	54:38
		<code>\bx@N</code>	54:28, 54:55
		<code>\bx@NN</code>	54:38
		<code>\bx@O</code>	54:29, 54:56
		<code>\bx@OO</code>	54:39
		<code>\bx@P</code>	54:29, 54:56
		<code>\bx@PP</code>	54:39
		<code>\bx@Q</code>	54:29, 54:56
		<code>\bx@QQ</code>	54:39

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=ltthyphen.dtx, 57=ltfinal.dtx

<code>\bx@R</code>	54:29, 54:56	<code>\cf@encoding</code>
<code>\bx@RR</code>	54:39	21:52, 21:59, 21:62, 21:69, 21:172, 24:319, 24:329, 24:339, <u>24:389</u>
<code>\bx@S</code>	54:34	<code>\ch@ck</code>	50:1259, 50:1278, 50:1390, 50:1411, 50:1503, 02:206, 02:207, 02:208, 02:209, 02:210, 02:236, 02:248, 02:249, 02:250, 02:251, 02:279, 02:281, 02:293, 02:294, 02:295, 02:296, <u>02:302</u>
<code>\bx@SS</code>	54:40	<code>\char@if@alph</code>	<u>06:806</u>
<code>\bx@T</code>	54:34	<code>\chardef@text@cmd</code>	21:3
<code>\bx@TT</code>	54:40	<code>\check@command</code>	06:187, <u>06:189</u>
<code>\bx@U</code>	54:34	<code>\check@icl</code>	32:9, <u>32:44</u> , 32:49, 32:55, 32:63, 32:70, 32:72
<code>\bx@UU</code>	54:40	<code>\check@icr</code>	32:9, <u>32:44</u> , 32:50, 32:56, 32:64, 32:73, 32:78, <u>1320</u>
<code>\bx@V</code>	54:34	<code>\check@mathfonts</code>	19:5, 21:322, 21:348, 21:380, 21:1276, 24:430, 24:435, 24:442, 24:444, <u>26:251</u> , 28:336, 28:419, 33:636, <u>675</u>
<code>\bx@VV</code>	54:40	<code>\check@nocorr</code>	<u>1303</u>
<code>\bx@W</code>	54:35	<code>\check@nocorr@</code>	32:46, <u>1322</u>
<code>\bx@WW</code>	54:41	<code>\check@orange</code>	26:380, <u>26:381</u>
<code>\bx@X</code>	54:35	<code>\check@single</code>	26:379, <u>26:401</u>
<code>\bx@XX</code>	54:41	<code>\cl@ckpt</code>	20:440, <u>22:37</u>
<code>\bx@Y</code>	54:35	<code>\cl@page</code>	34:4
<code>\bx@YY</code>	54:41	<code>\col@number</code> <u>54:93</u> , 54:146, 54:206, 54:218, <u>1296</u>
<code>\bx@Z</code>	54:35	<code>\color@begingroup</code>	24:740, 24:801, 38:104, 38:134, 40:29, <u>40:110</u> , 40:215, 40:442, 40:488, 40:505, 40:522, 41:47, 41:51, 45:475, 45:493, 45:511, 54:590, 54:698, <u>1293</u>
<code>\bx@ZZ</code>	54:21, 54:41, 54:51	<code>\color@endbox</code>	40:110, 45:253, 45:348, 45:366, 54:222, 54:844, 54:859, 54:909, 54:919, 54:968, 54:978
<code>\c@</code>	22:113	<code>\color@endgroup</code> 24:745, 24:807, 38:104, 38:134, 40:29, 40:119, <u>40:134</u> , 40:173, 40:194, 40:217, 40:470, 40:492, 40:509, 40:525, 41:49, 45:479, 45:497, 45:514, 54:595, 54:702, <u>871</u>
<code>\c@*</code>	22:15, <u>1358</u>	<code>\color@hbox</code>	<u>40:110</u> , 54:841, 54:856, 54:906, 54:916, 54:965, 54:975, <u>1307</u>
<code>\c@bottomnumber</code>	45:269, 45:274, <u>54:2916</u>	<code>\color@setgroup</code> <u>40:110</u> , 40:173, 40:192, <u>1298</u>
<code>\c@dbltopnumber</code>	<code>\color@vbox</code>	<u>40:110</u> , 45:96, 45:165, 45:339, 45:357, 45:381, 54:213, <u>1308</u>
.....	45:268, 45:283, 45:297, <u>54:2923</u>	<code>\conditionally@traceoff</code> <u>14:291</u> , 07:288, 07:300, <u>407</u>
<code>\c@enumi</code>	39:258	<code>\conditionally@traceon</code>	<u>14:291</u> , 07:324
<code>\c@enumii</code>	39:258	<code>\contionally@traceon</code> <u>407</u>
<code>\c@enumiv</code>	39:258, <u>1283</u>	<code>\copy@kernel@robust@command</code> <u>06:588</u> , <u>06:675</u>
<code>\c@equation</code>	38:349, 38:383, 38:543		
<code>\c@errorcontextlines</code>	14:212, <u>1287</u>		
<code>\c@footnote</code>	44:12, 45:397, 45:525		
<code>\c@localmathalphabets</code>		
..	<u>28:138</u> , 28:151, 28:283, 28:331, <u>668</u>		
<code>\c@mpfootnote</code>	40:454, 45:399		
<code>\c@ncel</code>	30:472, 30:473		
<code>\c@page</code>	34:3, 34:6, 34:7, 48:425, 48:485, 54:136, 54:2318		
<code>\c@secnumdepth</code>		
.....	44:56, 44:71, 44:81, <u>44:140</u>		
<code>\c@tocdepth</code>	44:140, 44:230, 44:253		
<code>\c@topnumber</code>	45:267, 45:271, <u>54:2912</u>		
<code>\c@totalnumber</code>	45:270, 45:276, <u>54:2919</u>		
<code>\c@totalpages</code>	53:348, 53:441		
<code>\calculate@math@sizes</code>	<u>24:717</u> , 26:220		
<code>\catcodetable@atletter</code>	<u>04:93</u> , 04:246		
<code>\catcodetable@initex</code>	<u>04:93</u> , 04:243		
<code>\catcodetable@latex</code>	<u>04:93</u> , 04:245		
<code>\catcodetable@string</code>	<u>04:93</u> , 04:244		
<code>\cdp@elt</code>	24:97, <u>24:117</u> , 24:128, 24:129, 24:150, 24:153, 24:155, 28:368, 28:484, 28:559, 28:633, 28:714, 57:472, 57:473		
<code>\cdp@list</code>	24:99, <u>24:115</u> , 24:129, 24:157, 24:158, 28:386, 28:486, 28:561, 28:635, 28:716, 57:473		

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=ltthyphen.dtx, 57=ltfinal.dtx

<code>\count@</code>	03:14, 03:15, 03:16, 03:17, 03:18, 03:20, 01:163, 01:164, 01:165, 01:170, 24:775, 24:781, 24:783, 26:23, 26:303, 26:305, 26:327, 26:328, 28:451, 28:453, 28:457, 28:816, 28:817, 28:818, 28:864, 28:865, 28:866, 28:925, 28:926, 28:927, 28:973, 28:974, 28:975, 28:1013, 28:1014, 28:1015, 28:1021, 28:1022, 28:1023, 28:1067, 28:1068, 28:1069, 28:1075, 28:1076, 28:1077, 28:1136, 28:1137, 28:1138, 28:1144, 28:1145, 28:1146, 32:115, 32:118, 06:169, 06:173, 06:287, 06:312, 42:729, 42:730, 42:731, 42:734, 42:735, 42:738, 42:742, 42:785, 42:786, 42:787, 42:790, 42:791, 42:794, 42:798, 50:1302, 50:1304, 50:1305, 50:1306, 50:1435, 50:1437, 50:1438, 50:1439, 50:1524, 50:1526, 50:1527, 50:1528, <u>02:41</u> , 57:239, 57:240, 57:247, 57:249, 57:534, 57:535, 57:542, 57:544, 02:191, 02:192, 02:197, 02:199, 02:205, 02:206, 02:207, 02:208, 02:209, 02:210, 02:211, 01:50, 02:505, 02:506, <i>1253</i>	<code>\DeclareEncodingSubset@aux</code>	24:187, 24:189
<code>\counterwithin@s</code>	22:166, 22:167, 22:185	<code>\DeclareFontEncoding@</code>	24:123, 24:125, 24:140, 57:383, 57:403, 57:469, <i>1334</i>
<code>\counterwithin@x</code>	22:166, 22:169, 22:186	<code>\DeclareFontEncoding@saved</code>	57:383, <u>57:403</u> , 57:485
<code>\counterwithout@s</code>	22:155, 22:156, 22:182	<code>\DeclareFontShape@</code>	24:22, <u>24:23</u>
<code>\counterwithout@x</code>	22:155, 22:158, 22:183	<code>\DeclareRobustCommand</code>	<i>319</i>
<code>\curr@fontshape</code>	21:198, 24:89, <u>24:460</u> , 24:468, 24:472, 24:474, 24:616, 24:622, 24:625, 24:634, 24:641, 24:643, 24:651, 24:657, 24:660, 24:668, 24:675, 24:677, 25:2842, 26:93, 26:101, 26:143, 26:170, 26:478, 26:498, 26:530, 26:546, 26:561, 28:390, 28:395, 29:557, 29:566, <i>1281</i>	<code>\DeclareSymbolFont@cmdropped</code>	28:476, 28:481, 28:524, 28:525
<code>\curr@math@size</code>	24:449, 26:257, 26:263, 26:268, 26:285	<code>\DeclareSymbolFontAlphabet@</code>	28:1191, <u>28:1194</u>
<code>\declare@command@copy</code>	<i>1340</i>	<code>\DeclareUnicodeAccent@</code>	21:1068, 21:1070, 21:1074
<code>\declare@commandcopy</code>	06:476, 06:480, 06:485, <u>06:488</u> , 06:507, <i>98</i>	<code>\def</code>	<i>314</i>
<code>\declare@commandcopy@do</code>	<u>06:488</u> , 06:534, 06:535	<code>\default@ds</code>	50:505, <u>50:536</u> , 50:600, 50:619, 50:1108, 50:1110, <i>1285</i>
<code>\declare@commandcopy@let</code> .	06:495, <u>06:499</u> , 06:509, 06:610, 06:611, <i>99</i>	<code>\default@family</code>	24:130, 24:162, 24:584, 24:598, 24:601, 24:626, 24:661, 57:474, <i>1280</i>
<code>\declare@environmentcopy</code>	06:517, 06:521, 06:526, 06:529	<code>\default@M</code>	24:137, 24:177, 24:180, <u>24:184</u> , 57:481, <i>1285</i>
<code>\declare@file@substitution</code>	<u>52:247</u> , 52:567, 53:517, <i>1104</i>	<code>\default@mextra</code>	27:10, <u>27:89</u>
<code>\declare@robustcommand</code>	<u>06:233</u>	<code>\default@series</code>	24:130, 24:163, 24:585, 24:599, 24:602, 24:623, 24:658, 57:474
		<code>\default@shape</code>	24:131, 24:164, 24:586, 24:600, 24:603, 24:621, 24:656, 57:475
		<code>\default@T</code>	24:171, 24:174, <u>24:184</u> , 24:335, <i>1285</i>
		<code>\define@mathalphabet</code>	27:18, <u>27:131</u> , <i>1280</i>
		<code>\define@mathgroup</code> .	27:19, <u>27:135</u> , <i>1280</i>
		<code>\define@newfont</code>	24:452, <u>24:461</u>
		<code>\delayed@f@adjustment</code>	24:353, 25:2783, 25:2784, <u>25:2785</u> , 25:2787, 25:2788, 25:2799, 25:3136, 25:3137, 25:3139, 25:3140, 26:123, 26:127, 26:135, 26:139, 29:663, <i>718</i>
		<code>\delayed@merge@font@series</code>	25:2784, <u>25:2848</u> , <u>25:2863</u> , 25:2902, 26:134, 26:137
		<code>\delayed@merge@font@shape</code>	25:3137, <u>25:3186</u> , 25:3241, 26:133, 26:136
		<code>\development@branch@name</code>	<u>03:11</u> , 03:39, 03:53, 03:54, 03:55, 03:62, 03:63, 03:64
		<code>\dimen@</code>	14:28, 14:29, 18:396, 18:401, 18:430, 18:435, 21:449, 21:450, 21:452, 21:453, 21:818, 21:819, 24:277, 24:279, 24:285, 24:298, 24:301, 24:305,

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- 24:720, 24:721, 24:722, 24:726,
26:452, 26:453, 26:454, 26:455,
26:459, 33:66, 33:68, 33:892, 33:894,
38:72, 38:73, 38:199, 38:200, 38:201,
38:202, 40:587, 40:590, 41:176,
41:177, 02:41, 54:714, 54:716,
54:726, 54:728, 02:502, 02:503,
02:539, 02:540, 02:542, 02:544, 1319
- `\dimen@i` 02:41
- `\dimen@ii` 24:281, 24:286, 02:41
- `\disable@package@load`
..... 52:477, 53:475, 1104
- `\disply` 38:204, 38:208, 38:209
- `\do@add@percent@to@temptokena` ...
..... 37:206, 37:212, 37:230
- `\do@emfont@update` 29:558, 29:562, 29:582
- `\do@noligs` 37:739, 37:744
- `\do@subst@correction`
..... 24:85, 26:483, 26:538
- `\document@default@language`
..... 20:51, 20:52,
20:121, 20:122, 54:799, 54:875, 57:286
- `\document@select@group`
..... 28:140, 28:146, 28:403, 668
- `\dont@add@percent@to@temptokena` .
..... 37:205, 37:207, 37:231
- `\dorestore@version` ... 28:115, 28:120
- `\ds@` 50:538, 50:1112, 1285
- `\dt@pfalse` 38:205
- `\dt@ptrue` 38:204
- `\e@alloc` 04:15, 04:49,
04:79, 04:89, 04:178, 04:182, 04:186,
04:194, 04:202, 04:210, 02:51, 02:52,
02:53, 02:55, 02:56, 02:63, 02:64,
02:66, 02:68, 57:12, 57:47, 02:79,
02:82, 02:84, 02:138, 02:230, 1340
- `\e@alloc@attribute@count`
04:66, 04:74, 04:75, 04:76, 04:80, 04:237
- `\e@alloc@bytecode@count` ... 04:70,
04:197, 04:198, 04:199, 04:203, 04:253
- `\e@alloc@ccodetable@count`
04:67, 04:84, 04:85, 04:86, 04:90, 04:241
- `\e@alloc@chardef` 04:48,
04:178, 04:194, 04:202, 04:210,
02:60, 57:12, 02:102, 02:211, 02:212
- `\e@alloc@intercharclass@top` .. 57:35
- `\e@alloc@luachunk@count` ... 04:71,
04:205, 04:206, 04:207, 04:211, 04:255
- `\e@alloc@luafunction@count`
... 04:68, 04:173, 04:174, 04:175,
04:179, 04:183, 04:187, 04:247, 04:249
- `\e@alloc@top` 04:47,
04:80, 04:179, 04:183, 04:187,
04:195, 04:203, 04:211, 02:55,
02:63, 57:12, 02:102, 02:188, 02:259
- `\e@alloc@whatsit@count` 04:69,
04:189, 04:190, 04:191, 04:195, 04:251
- `\e@ch@ck` .. 04:51, 04:55, 02:142, 02:152
- `\e@insert@top`
..... 02:257, 02:259, 02:276, 02:291
- `\e@mathgroup@top`
..... 28:57, 28:150, 28:151,
28:188, 28:218, 02:79, 02:124, 1330
- `\em@currfont` 29:557, 29:568, 715
- `\em@force`
.. 29:566, 29:571, 29:574, 29:584, 715
- `\emfontdeclare@clist`
..... 29:547, 29:549, 29:553,
29:558, 29:563, 29:569, 29:580, 714
- `\empty@sfcnt` 26:491,
26:492, 26:493, 26:507, 26:512, 26:564
- `\ENC@cmd` 1300
- `\enc@update` 24:320, 24:322,
24:338, 24:341, 26:148, 26:174, 1300
- `\end@dblfloat` 45:205
- `\end@float`
.. 45:189, 45:227, 45:243, 45:383, 986
- `\endlinechar` 329
- `\enlargethispage` 1016
- `\err@rel@i` . 27:12, 27:99, 27:132, 27:136
- `\error@fontshape` .. 24:579, 24:594,
24:619, 24:654, 26:108, 26:528, 28:389
- `\escapechar` 322
- `\et@xmaxfam` . 04:22, 04:26, 04:30, 04:38
- `\et@xmaxregs` 04:29, 04:31,
04:32, 04:33, 04:34, 04:35, 04:36, 04:37
- `\every@math@size`
..... 24:79, 26:236, 26:248, 1307
- `\every@size` 1281
- `\execute@size@function`
..... 26:363, 26:391, 26:405, 26:422
- `\expand@font@defaults` ... 29:38,
29:149, 29:257, 29:280, 29:310,
29:331, 29:355, 29:366, 29:397,
29:398, 29:437, 29:439, 29:471,
29:473, 29:502, 33:26, 33:47, 226
- `expand@font@defaults` 29:421
- `\expandafter` 1119
- `\external@font` 26:85, 26:88, 26:99,
26:103, 26:105, 26:392, 26:406,
26:468, 26:502, 26:570, 26:572, 26:574
- `\extra@def` 27:9, 27:84, 1280
- `\extract@alph@from@version`
..... 24:694, 24:700,
28:163, 28:194, 28:224, 28:256, 1283
- `\extract@default@composite`
..... 21:1084, 21:1091

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=ltthyphen.dtx, 57=ltfinal.dtx

- \extract@default@composite@a ... 29:695, 29:740, 33:7, 33:577, 713
- 21:1093, 21:1097
- \extract@default@composite@b ...
- 21:1095, 21:1099
- \extract@font 24:475, 26:82
- \extract@fontinfo 26:359, 26:366
- \extract@rangefontinfo
- 26:376, 26:383, 26:402, 26:435
- \extract@sizefn 26:351, 26:373
- \f@... 1300
- \f@baselineskip
- .. 21:892, 21:1242, 24:380, 24:387,
- 24:605, 26:122, 26:168, 26:183,
- 26:187, 26:202, 26:216, 26:227, 26:241
- \f@depth 45:291, 54:343
- \f@encoding 21:196, 04:260,
- 04:275, 04:283, 24:314, 24:333,
- 24:336, 24:337, 24:339, 24:389,
- 24:455, 24:460, 24:479, 24:481,
- 24:483, 24:488, 24:490, 24:521,
- 24:583, 24:615, 24:650, 25:2827,
- 25:2831, 25:3172, 25:3176, 26:92,
- 26:129, 26:308, 26:518, 28:374, 1300
- \f@family 24:209,
- 24:210, 24:213, 24:216, 24:233,
- 24:236, 24:342, 24:350, 24:362,
- 24:372, 24:383, 24:456, 24:460,
- 24:479, 24:481, 24:483, 24:488,
- 24:490, 24:522, 24:601, 24:626,
- 24:661, 25:2827, 25:2831, 25:3172,
- 25:3176, 26:92, 26:129, 28:374,
- 28:405, 29:152, 29:179, 29:180,
- 29:259, 29:260, 29:261, 29:282,
- 29:283, 29:284, 29:320, 29:321,
- 29:322, 29:340, 29:341, 29:342,
- 29:356, 29:357, 29:358, 29:367,
- 29:368, 29:369, 29:505, 29:518,
- 29:660, 29:678, 29:694, 33:23, 33:27,
- 33:29, 33:31, 33:44, 33:48, 33:50,
- 33:52, 33:57, 33:64, 33:858, 33:861,
- 33:875, 33:884, 33:890, 33:1105, 1280
- \f@linespread 24:383, 26:121,
- 26:167, 26:184, 26:185, 26:188,
- 26:196, 26:199, 26:210, 26:213, 1300
- \f@series 19:14,
- 24:342, 24:373, 24:384, 24:457,
- 24:460, 24:602, 24:623, 24:658,
- 25:2788, 25:2798, 25:2807, 25:2817,
- 25:2851, 25:2870, 25:2871, 25:3172,
- 25:3176, 26:126, 26:129, 26:132,
- 28:406, 29:145, 29:161, 29:163,
- 29:167, 29:168, 29:169, 29:190,
- 29:196, 29:199, 29:201, 29:223,
- 29:521, 29:525, 29:661, 29:679,
- \f@series@saved 26:126, 26:132
- \f@shape
- ... 24:342, 24:352, 24:364, 24:374,
- 24:385, 24:458, 24:460, 24:603,
- 24:621, 24:656, 25:2827, 25:2831,
- 25:3140, 25:3147, 25:3153, 25:3163,
- 25:3170, 25:3174, 25:3177, 25:3180,
- 25:3189, 25:3196, 25:3198, 25:3233,
- 26:125, 26:129, 26:131, 28:407,
- 29:662, 29:680, 29:696, 29:741, 720
- \f@shape@saved 26:125, 26:131
- \f@size 21:198, 21:891, 21:1241, 24:89,
- 24:380, 24:386, 24:459, 24:604,
- 24:643, 24:677, 24:719, 24:720,
- 24:723, 24:724, 26:122, 26:143,
- 26:168, 26:170, 26:181, 26:201,
- 26:216, 26:219, 26:222, 26:227,
- 26:234, 26:241, 26:253, 26:256,
- 26:262, 26:268, 26:285, 26:286,
- 26:289, 26:294, 26:360, 26:367,
- 26:386, 26:388, 26:403, 26:454,
- 26:456, 26:458, 26:474, 26:475,
- 26:480, 26:494, 26:506, 26:511,
- 26:523, 26:531, 26:536, 26:562,
- 26:576, 29:557, 29:566, 33:66, 33:892
- \f@user@size
- 26:474, 26:479, 26:523, 26:536
- \f@warn@break 1302
- \famdef@ult 29:445
- \filec@ntents
- 50:1209, 50:1212, 50:1215,
- 50:1226, 50:1251, 50:1344, 50:1347,
- 50:1350, 50:1361, 50:1382, 50:1475,
- 50:1497, 50:1585, 50:1808, 1312
- \filec@ntents@checkdir
- 50:1232, 50:1234, 50:1252,
- 50:1367, 50:1369, 50:1383, 50:1482
- \filec@ntents@force
- 50:1228, 50:1363, 50:1478
- \filec@ntents@noheader
- 50:1230, 50:1365, 50:1480
- \filec@ntents@nosearch
- 50:1231, 50:1366, 50:1481
- \filec@ntents@nowarn 50:1236
- \filec@ntents@opt
- 50:1212, 50:1215, 50:1217,
- 50:1347, 50:1350, 50:1352, 50:1477
- \filec@ntents@OPTION 1074
- \filec@ntents@overwrite
- 50:1229, 50:1364, 50:1479
- \filec@ntents@warning
- 50:1237, 50:1239, 50:1284

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

<code>\filec@ntents@where</code>	54:1502, 54:1507, 54:1512, 54:1517,
..... 50:1233, 50:1235, 50:1264,	54:1518, 54:1523, 54:1528, 54:1529,
50:1368, 50:1370, 50:1395, 50:1483	54:1530, 54:1538, 54:1542, 54:1547,
<code>\filename@area</code> ... 20:642, 20:657,	54:1551, 54:1556, 54:1567, 54:1568,
20:667, 20:699, 20:700, 20:704,	54:1570, 54:1588, 54:1596, 54:1600,
20:729, 20:732, 20:757, 20:773,	54:1608, 54:1611, 54:1617, 54:1627,
20:775, 01:230, 01:236, 01:243,	54:1631, 54:1640, 54:1646, 54:1652,
01:249, 01:256, 01:262, 01:269, 50:941	54:1658, 54:1665, 54:1667, 54:1673,
<code>\filename@base</code>	54:1678, 54:1680, 54:1682, 54:1690,
20:642, 20:657, 20:667, 20:699, 20:701,	54:1695, 54:1701, 54:1706, 54:1712,
20:704, 20:729, 20:732, 20:764,	54:1726, 54:1727, 54:1730, 54:1751,
20:773, 01:279, 01:286, 01:299, 50:942	54:1760, 54:1766, 54:1775, 54:1778,
<code>\filename@dot</code>	54:1785, 54:1795, 54:1799, 54:1811,
01:297, 01:303	54:1817, 54:1822, 54:1827, 54:1831,
<code>\filename@dots</code> . 01:281, 01:283, 01:288	54:1836, 54:1837, 54:1844, 54:1849,
<code>\filename@ext</code> 20:643, 20:658,	54:1853, 54:1860, 54:1869, 54:1873,
20:668, 20:695, 20:696, 20:699,	54:1877, 54:1878, 54:1882, 54:1883,
20:702, 20:704, 20:725, 20:726,	54:1893, 54:1899, 54:1904, 54:1909,
20:729, 20:732, 20:765, 01:278,	54:1913, 54:1917, 54:1918, 54:1925,
01:285, 01:295, 01:297, 50:943, <i>1110</i>	54:1930, 54:1934, 54:1941, 54:1950,
<code>\filename@parse</code>	54:1954, 54:1958, 54:1959, 54:1963,
... 01:94, 20:640, 20:655, 20:665,	54:1964, 54:1973, 54:1979, 54:1985,
20:694, 20:724, 20:762, <u>01:226</u> , 50:940	54:1991, 54:1995, 54:2001, 54:2003,
<code>\filename@path</code>	54:2011, 54:2016, 54:2021, 54:2029,
... 01:231, 01:232, 01:237, 01:244,	54:2038, 54:2043, 54:2048, 54:2050,
01:245, 01:250, 01:257, 01:258, 01:263	54:2055, 54:2057, 54:2068, 54:2074,
<code>\filename@simple</code>	54:2084, 54:2090, 54:2094, 54:2095,
..... 01:234, 01:247, 01:260,	54:2100, 54:2101, 54:2107, 54:2110,
01:270, 01:274, 01:276, 01:291, 01:293	54:2111, 54:2112, 54:2119, 54:2120,
<code>\finish@module@release</code> .. 03:89, 03:93	54:2121, 54:2129, 54:2134, 54:2146,
<code>\finph@nt</code>	54:2147, 54:2154, 54:2157, 54:2165,
38:104, 38:106,	54:2169, 54:2173, 54:2174, 54:2178,
38:110, 38:111, 38:119, 38:120, <i>1251</i>	54:2179, 54:2189, 54:2195, 54:2205,
<code>\finsm@sh</code> ... 38:134, 38:141, 38:147,	54:2211, 54:2215, 54:2216, 54:2222,
38:153, 38:154, 38:158, 38:159, <i>1251</i>	54:2223, 54:2230, 54:2233, 54:2234,
<code>\fix@penalty</code>	54:2235, 54:2243, 54:2244, 54:2245,
32:101	54:2254, 54:2259, 54:2272, 54:2274,
<code>\fixed@sfcnt</code> ... 26:566, 26:567, 26:568	54:2281, 54:2284, 54:2293, 54:2297,
<code>\fl@ShowFloat</code> 54:2404, 54:2410, 54:2425	54:2301, 54:2302, 54:2306, 54:2307,
<code>\fl@trace</code>	54:2359, 54:2364, 54:2370, 54:2380,
54:238, 54:265,	54:2387, 54:2401, 54:2402, 54:2406,
54:321, 54:349, 54:356, 54:377,	54:2417, <u>54:2429</u> , 54:2537, 54:2550,
54:424, 54:472, 54:514, 54:524,	54:2551, 54:2555, 54:2558, 54:2560,
54:525, 54:526, 54:527, 54:537,	54:2563, 54:2566, 54:2568, 54:2609,
54:538, 54:539, 54:540, 54:541,	54:2616, 54:2621, 54:2627, 54:2632,
54:551, 54:1055, 54:1074, 54:1093,	54:2636, 54:2642, 54:2655, 54:2657,
54:1111, 54:1113, 54:1252, 54:1256,	54:2664, 54:2669, 54:2674, 54:2676,
54:1268, 54:1269, 54:1270, 54:1271,	54:2682, 54:2684, 54:2691, 54:2705,
54:1277, 54:1280, 54:1288, 54:1292,	54:2707, 54:2714, 54:2719, 54:2724,
54:1303, 54:1308, 54:1313, 54:1314,	54:2726, 54:2732, 54:2734, 54:2741,
54:1315, 54:1316, 54:1323, 54:1326,	54:2770, 54:2772, 54:2787, 54:2789,
54:1334, 54:1345, 54:1351, 54:1356,	54:2803, 54:2813, 54:2816, 54:2821,
54:1361, 54:1367, 54:1368, 54:1373,	54:2874, 54:2891, 54:2896, 54:2904
54:1379, 54:1380, 54:1381, 54:1389,	
54:1393, 54:1398, 54:1402, 54:1407,	
54:1418, 54:1419, 54:1421, 54:1439,	
54:1448, 54:1454, 54:1463, 54:1466,	
54:1472, 54:1482, 54:1486, 54:1496,	
	<code>\fl@tracemessage</code>

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpara.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- ... 54:2401, 54:2418, 54:2427, [54:2429](#)
- `\fl@traceval`
 54:2411, 54:2412, 54:2413,
 54:2414, 54:2417, 54:2426, [54:2429](#)
- `\float@count` 02:51,
 02:52, 02:53, 02:62, 02:188, 02:205,
 02:211, 02:213, 02:214, 02:222, 02:230
- `\fmtversion@topatch` 57:661,
 57:663, 57:675, 57:676, 57:688, 57:696
- `\font@info` 26:100, 26:366, 26:435, 26:440
- `\font@name` .. 21:197, 21:200, 24:87,
 24:257, 24:259, 24:451, 24:466,
 24:642, 24:676, 26:85, 26:89, 26:91,
 26:106, 26:142, 26:145, 26:155,
 26:169, 26:172, 26:331, 26:332,
 26:333, 26:334, 26:335, 26:340, *1280*
- `\font@submax` 26:442, 26:471, 26:472,
 37:53, 37:55, 37:118, 37:120, 37:157,
 37:159, 57:299, 57:301, 57:310, *1295*
- `\fps@dbl` [45:34](#)
- `\freeze@math@version` . 28:155, [28:274](#)
- `\frozen@everydisplay`
 .. [24:418](#), [24:423](#), 24:440, 24:442, *672*
- `\frozen@everymath`
 [24:418](#), [24:432](#), 24:444, *672*
- `\g@addto@macro`
 ... 09:78, 25:3251, 28:419, 50:133,
 50:460, 50:1120, 50:1136, 50:1137,
 53:352, 53:358, 53:408, [06:918](#), *234*
- `\G@refundefined` *1318*
- `\G@refundefinedfalse` 35:5, *798*
- `\G@refundefinedtrue` [35:3](#),
 35:16, 35:33, 35:50, 36:203, 36:210,
 36:220, 36:226, 47:41, 47:68, 47:85, *816*
- `\gen@sfcnt` 26:503, 26:504, 26:505
- `\genb@sfcnt` 26:508, 26:509, 26:510, *1316*
- `\genb@x` 26:511, [26:513](#)
- `\genb@y` [26:513](#)
- `\get@cdp` 28:587, 28:595, [28:628](#)
- `\get@external@font` 26:84, [26:97](#), 26:537
- `\getanddefine@fonts` 24:689, 24:707,
 [26:321](#), 28:60, 28:88, 28:133, 28:160,
 28:191, 28:221, 28:252, 28:299,
 28:349, 28:454, 28:538, 28:592,
 28:594, 28:611, 28:734, 28:735,
 28:767, 28:768, 28:1210, 28:1211, *672*
- `\glb@currsize`
 ... 20:41, 20:111, 20:169, 24:409,
 [26:218](#), 26:253, 26:257, 26:263, 26:286
- `\glb@settings`
 24:410, [26:218](#), 26:265, 26:296
- `\gobble@finish@module@release` ...
 03:109, 03:111, 03:170
- `\gobble@font@spec` *1314*
- `\group@elt` 28:36, 28:452,
 28:499, 28:500, [28:531](#), 28:535, 28:1242
- `\group@list` 28:456, 28:506,
 [28:529](#), 28:534, 28:535, 28:584,
 28:810, 28:858, 28:919, 28:1004,
 28:1007, 28:1058, 28:1061, 28:1127,
 28:1130, 28:1197, 28:1248, *1323*
- `\h@false` 38:81
- `\h@true` 38:82, 38:83
- `\hb@xt@` 21:445, [06:29](#), 38:210,
 38:376, 38:455, 38:470, 38:482,
 38:509, 38:540, 40:48, 40:63, 40:86,
 40:104, 40:248, 40:267, 40:614,
 40:618, 40:619, 40:620, 41:37, 42:31,
 42:43, 42:62, 42:74, 42:105, 42:117,
 42:265, 42:299, 42:302, 42:305,
 42:307, 42:314, 42:373, 42:452,
 42:587, 42:597, 42:740, 42:796,
 44:243, 44:266, 44:273, 54:843,
 54:858, 54:908, 54:918, 54:967,
 54:977, 54:2343, 54:2807, 54:2808,
 54:2812, 54:2850, 54:2851, 54:2855,
 54:2879, 54:2880, 54:2886, 02:549, *1321*
- `\hexnumber@` . 28:831, 28:839, 28:874,
 28:882, 28:903, 28:913, 28:939,
 28:947, 28:955, 28:964, 28:967,
 28:976, 28:977, 28:1016, 28:1024,
 28:1070, 28:1078, 28:1097, 28:1098,
 28:1108, 28:1109, 28:1114, 28:1140,
 28:1148, 28:1153, 28:1155, [29:646](#)
- `\hgl@` 02:504, 02:505
- `\hmode@bgroup` 21:85, [21:93](#), 21:347,
 21:376, 21:410, 21:416, 21:447,
 21:458, 21:465, 21:497, 21:504,
 21:507, 21:509, 21:517, 21:533,
 21:752, 21:782, 21:788, 21:820,
 21:827, 21:855, 21:858, 21:915,
 21:1275, 32:7, 33:589, 33:596, *1325*
- `\hmode@start@before@group`
 . 21:86, 21:169, 21:171, 21:177, [21:202](#)
- `\hyper@nopatch@longtable` 55:302
- `\if@afterindent` [44:124](#), 44:131
- `\if@compatibility` [50:2](#), 50:676
- `\if@endpe` 37:338,
 37:353, 37:370, 37:380, [39:140](#), *872*
- `\if@eqnsw` [38:357](#), 38:423
- `\if@fcolmade`
 ... [54:93](#), 54:262, 54:392, 54:401,
 54:439, 54:449, 54:1053, 54:1073,
 54:1091, 54:1120, 54:1200, 54:2769,
 54:2786, 54:2820, 54:2862, 54:2901
- `\if@filesw` [20:5](#), 20:36,
 20:106, 20:164, 20:298, 20:310,
 20:331, 20:360, 20:372, 20:387,

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=ltthyphen.dtx, 57=ltfinal.dtx

- 20:407, 20:419, 20:426, 20:438,
37:21, 37:59, 37:87, 37:124, 37:149,
37:163, 44:153, 47:4, 47:8, 47:39,
47:48, 47:56, 47:67, 47:84, 50:1262,
50:1279, 50:1393, 50:1412, 53:361, *1356*
- `\if@firstamp` [41:251](#)
- `\if@firstcolumn` ... [54:93](#), [54:244](#),
[54:277](#), [54:394](#), [54:442](#), [54:2315](#),
[54:2800](#), [54:2829](#), [54:2870](#), *1012*
- `\if@font@series@context`
..... [29:510](#), [29:529](#), [29:537](#), *712*
- `\if@forced@series` [25:2789](#), [29:144](#), *1337*
- `\if@ignore` [28:317](#), [28:356](#),
[37:4](#), [37:339](#), [37:356](#), [37:372](#), [37:381](#)
- `\if@in@minipage@env` .. [40:423](#), [40:445](#)
- `\if@includeinrelease`
..... [03:73](#), [03:76](#), [03:132](#), [06:872](#)
- `\if@inlabel` [39:28](#), [39:65](#),
[39:102](#), [39:191](#), [39:214](#), [54:159](#), [54:186](#)
- `\if@inmath` *1320*
- `\if@insert` [54:93](#),
[54:1331](#), [54:1444](#), [54:1478](#), [54:1593](#),
[54:1623](#), [54:1756](#), [54:1791](#), [54:1866](#),
[54:1947](#), [54:2035](#), [54:2162](#), [54:2290](#)
- `\if@listfiles@hashes`
..... [20:743](#), [20:778](#), [20:785](#), [20:789](#)
- `\if@listfiles@sizes`
..... [20:743](#), [20:779](#), [20:783](#)
- `\if@minipage`
... [16:26](#), [18:240](#), [18:255](#), [18:272](#),
[18:291](#), [18:324](#), [18:358](#), [37:528](#),
[37:550](#), [39:180](#), [40:420](#), [41:79](#), [45:20](#)
- `\if@mparswitch` [54:93](#), [54:2317](#)
- `\if@multiplelabels` [35:93](#)
- `\if@mypkg@draft` *1091*
- `\if@negarg` [42:157](#), [42:198](#), [42:212](#), [42:273](#)
- `\if@newlist` .. [37:570](#), [39:29](#), [39:33](#),
[39:69](#), [39:78](#), [39:106](#), [39:197](#), [54:801](#),
[54:864](#), [54:877](#), [54:922](#), [54:935](#), [54:981](#)
- `\if@nmbrrlist` [39:33](#), [39:232](#)
- `\if@no@font@opt` . [27:16](#), [27:110](#), [27:129](#)
- `\if@nobreak` [18:121](#), [18:148](#),
[18:293](#), [18:326](#), [18:360](#), [20:203](#),
[20:215](#), [39:198](#), [39:223](#), [40:383](#),
[40:404](#), [44:47](#), [44:128](#), [45:180](#),
[45:373](#), [48:256](#), [49:48](#), [49:58](#), [49:71](#),
[49:79](#), [49:94](#), [49:102](#), [54:163](#), [54:190](#),
[54:333](#), [54:1423](#), [54:1572](#), [54:1733](#), *1321*
- `\if@noitemarg` [39:32](#), [39:230](#)
- `\if@noparitem` [39:30](#), [39:188](#)
- `\if@noparlist` [39:31](#), [39:114](#)
- `\if@noskipsec` [18:148](#),
[39:58](#), [40:384](#), [40:405](#), [44:38](#),
[44:40](#), [44:97](#), [45:374](#), [54:153](#), [54:180](#)
- `\if@ovb` [42:427](#),
[42:494](#), [42:525](#), [42:550](#), [42:561](#), [42:574](#)
- `\if@ovhline` [42:459](#), [42:589](#)
- `\if@ovl` [42:427](#),
[42:492](#), [42:521](#), [42:546](#), [42:591](#), [42:600](#)
- `\if@ovr` [42:427](#),
[42:491](#), [42:520](#), [42:545](#), [42:588](#), [42:598](#)
- `\if@ovt` [42:427](#),
[42:493](#), [42:524](#), [42:549](#), [42:566](#), [42:578](#)
- `\if@ovvline` [42:459](#), [42:564](#)
- `\if@partsw` . [20:5](#), [20:302](#), [20:364](#), [20:411](#)
- `\if@pboxsw` [40:351](#), [40:373](#), [40:529](#)
- `\if@reversemargin` [54:93](#), [54:2320](#)
- `\if@rjfield` [41:19](#), [41:33](#)
- `\if@skipping@module`
..... [03:137](#), [03:149](#), [03:152](#)
- `\if@specialpage`
..... [54:93](#), [54:809](#), [54:884](#), [54:942](#)
- `\if@tempswa` .. [12:9](#), [20:308](#), [20:370](#),
[20:417](#), [21:1564](#), [24:100](#), [24:785](#),
[28:487](#), [28:562](#), [28:636](#), [28:717](#),
[28:1241](#), [37:61](#), [37:126](#), [37:165](#),
[37:535](#), [37:556](#), [47:95](#), [50:1291](#),
[50:1424](#), [50:1513](#), [52:292](#), [52:293](#),
[52:315](#), [52:316](#), [52:336](#), [52:337](#),
[54:1264](#), [54:1300](#), [54:2126](#), [54:2251](#),
[02:270](#), [01:62](#), [01:63](#), [01:64](#), *1328*
- `\if@test` [54:8](#), [54:9](#),
[54:1161](#), [54:1180](#), [54:1220](#), [54:1242](#),
[54:1306](#), [54:1391](#), [54:1400](#), [54:1540](#),
[54:1549](#), [54:1693](#), [54:1704](#), [54:1847](#),
[54:1928](#), [54:2014](#), [54:2132](#), [54:2257](#)
- `\if@twocolumn` [20:26](#),
[20:96](#), [20:153](#), [45:32](#), [45:210](#), [45:235](#),
[54:93](#), [54:137](#), [54:265](#), [54:276](#),
[54:393](#), [54:441](#), [54:465](#), [54:1055](#),
[54:1111](#), [54:2314](#), [54:2771](#), [54:2788](#)
- `\if@twoside`
.. [54:93](#), [54:136](#), [54:813](#), [54:887](#), [54:945](#)
- `\ifdt@p` [38:203](#), [38:205](#)
- `\ifFileExists@` [20:471](#), [20:498](#)
- `\ifFileExists@C` [20:498](#)
- `\ifG@refundefined` [35:3](#), [35:4](#), [35:5](#)
- `\ifh@` [38:76](#), [38:114](#), [38:123](#)
- `\ifin@` [21:1584](#), [21:1587](#), [27:50](#), [27:52](#),
[28:3](#), [28:23](#), [28:441](#), [28:583](#), [28:585](#),
[28:646](#), [28:659](#), [28:729](#), [28:731](#),
[28:759](#), [28:811](#), [28:825](#), [28:859](#),
[28:871](#), [28:920](#), [28:936](#), [28:1005](#),
[28:1008](#), [28:1029](#), [28:1059](#), [28:1062](#),
[28:1125](#), [28:1128](#), [28:1131](#), [28:1198](#),
[28:1200](#), [28:1229](#), [29:211](#), [29:215](#),
[29:219](#), [50:236](#), [50:254](#), [50:548](#), [50:582](#)
- `\ifmath@fonts` [24:267](#), [26:223](#)

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=ltthyphen.dtx, 57=ltfinal.dtx

- `\ifmaybe@ic` 32:82, 32:91, 1328
`\ifnot@nil` 26:344, 26:361, 26:382
`\ifrestore@version` 1319
`\iftc@forced` .. 33:832, 33:843, 33:1111
`\ifv@` 38:75, 38:113, 38:122
`\ifx` 330
`\in@` 21:1582,
21:1585, 27:49, 27:51, 28:3, 28:22,
28:440, 28:582, 28:584, 28:642,
28:655, 28:728, 28:730, 28:757,
28:809, 28:820, 28:857, 28:868,
28:918, 28:932, 28:1003, 28:1006,
28:1026, 28:1057, 28:1060, 28:1122,
28:1126, 28:1129, 28:1196, 28:1199,
28:1227, 29:209, 29:213, 29:217,
50:235, 50:252, 50:545, 50:581, 1328
`\in@@` 25:2877, 25:2878,
25:2880, 25:2881, 28:6, 28:7, 28:8, 28:10
`\in@false` 28:11
`\in@true` 28:13
`\init@restore@glb@settings`
..... 26:266, 26:269, 26:271
`\init@restore@version` 28:63,
28:92, 28:109, 28:124, 28:125, 1319
`\init@series@setup` 25:3252,
25:3257, 29:121, 29:207, 29:247
`\input@path` 01:93,
01:115, 01:117, 01:123, 01:125,
01:131, 01:133, 01:138, 01:140,
20:484, 20:538, 20:560, 20:587,
20:604, 01:150, 01:217, 52:275, 1109
`\insec@unt` 02:37, 54:57,
02:51, 02:52, 02:53, 02:62, 02:90,
02:91, 02:92, 02:94, 02:247, 02:248,
02:249, 02:250, 02:251, 02:252,
02:263, 02:264, 02:265, 02:266,
02:267, 02:271, 02:273, 02:292,
02:293, 02:294, 02:295, 02:296, 02:297
`\install@mathalphabet`
..... 24:684, 24:701, 24:708,
28:460, 28:463, 28:589, 28:590,
28:687, 28:739, 28:742, 28:749,
28:764, 28:765, 28:772, 28:1212, 28:1214
`\is@range` 26:377, 26:378
`\kernel@ifnextchar` .. 03:90, 06:81,
06:100, 06:150, 50:440, 06:783, 06:798
`\kernel@make@fragile`
. 18:25, 18:26, 18:27, 18:28, 18:29,
21:190, 21:191, 37:487, 37:488,
37:489, 38:90, 38:91, 38:92, 38:93,
38:179, 38:180, 38:181, 41:160,
41:161, 41:162, 06:387, 42:822,
42:823, 42:824, 42:825, 42:826,
42:827, 42:828, 42:829, 42:830,
42:831, 42:832, 42:833, 44:23, 44:24,
44:25, 44:26, 44:27, 49:85, 49:86,
06:899, 06:900, 06:901, 06:902,
06:903, 06:904, 06:905, 06:906,
06:907, 06:908, 06:909, 06:910,
06:911, 06:912, 06:913, 06:914, 1036
`\l@ngrel@x`
... 06:74, 06:75, 06:76, 06:120, 06:167
`\l@nohyphenation`
..... 37:532, 37:724, 57:283, 1333
`\last@fontshape`
..... 24:617, 24:635, 24:652, 24:669
`\latexrelease@postltxcmd` 07:3281
`\latexrelease@postltxexpl` 05:132
`\leavevmode@ifvmode`
..... 18:537, 18:538, 18:546,
30:635, 30:637, 30:639, 30:641,
38:115, 38:154, 38:219, 38:239, 38:240
`\load@onefile@withoptions`
..... 50:945, 50:987, 50:1099, 1066
`\load@onefilewithoptions`
..... 50:892, 50:1049, 50:1600, 1065
`\lower@bound` ... 26:387, 26:388, 26:399
`\lst@vskip` 423
`\LT@cols` 1253
`\ltx@sh@ft` .. 21:410, 21:417, 21:497,
21:505, 21:782, 21:789, 02:541, 1328
`\ltx@star@counter` 35:111, 35:127, 35:140
`\m@ne` 02:39, 391
`\m@th` 19:13, 30:348,
30:470, 30:472, 30:473, 30:476,
30:517, 30:541, 30:544, 30:548,
30:551, 30:558, 30:561, 30:568,
30:571, 30:653, 38:68, 38:71, 38:106,
38:140, 38:147, 38:167, 38:169,
38:185, 38:204, 38:367, 38:470,
38:482, 38:509, 38:520, 40:351,
40:373, 40:555, 41:181, 44:239,
44:262, 45:400, 45:409, 45:416,
45:437, 45:444, 02:521, 02:533, 1303
`\makeph@nt` 38:101, 38:103
`\makesm@sh` 38:131, 38:133
`\mandatory@arg` 26:415,
26:502, 26:506, 26:511, 26:518,
26:520, 26:525, 26:527, 26:532,
26:534, 26:547, 26:563, 26:570, 26:572
`\math@bgroup` 24:715,
26:307, 26:313, 28:54, 28:82, 28:147,
28:176, 28:185, 28:207, 28:215,
28:246, 32:130, 32:131, 32:138, 669
`\math@egroup` 24:715, 26:311,
26:312, 32:131, 32:132, 32:139, 1281
`\math@famname` 1280

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltxexpl.dtx,
06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx,
16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
56=ltthyphen.dtx, 57=ltfinal.dtx

- `\math@fonts` [24:685](#),
[24:690](#), [26:233](#), [26:337](#), [28:61](#),
[28:90](#), [28:161](#), [28:192](#), [28:222](#), [28:254](#)
- `\math@fontsfalse`
..... [19:7](#), [21:322](#), [21:349](#),
[21:380](#), [21:1277](#), [24:78](#), [24:269](#),
[24:279](#), [24:302](#), [33:67](#), [33:637](#), [33:893](#)
- `\math@fontstrue` [24:267](#), [24:727](#)
- `\math@group` [1280](#)
- `\math@version`
..... [24:8](#), [24:399](#), [24:689](#), [24:693](#),
[24:695](#), [24:696](#), [24:698](#), [26:231](#),
[28:57](#), [28:60](#), [28:65](#), [28:66](#), [28:70](#),
[28:85](#), [28:89](#), [28:94](#), [28:95](#), [28:99](#),
[28:112](#), [28:113](#), [28:114](#), [28:127](#),
[28:128](#), [28:129](#), [28:150](#), [28:152](#),
[28:154](#), [28:155](#), [28:160](#), [28:164](#),
[28:166](#), [28:168](#), [28:172](#), [28:188](#),
[28:191](#), [28:195](#), [28:197](#), [28:199](#),
[28:203](#), [28:218](#), [28:221](#), [28:225](#),
[28:227](#), [28:229](#), [28:233](#), [28:249](#),
[28:253](#), [28:257](#), [28:259](#), [28:261](#),
[28:265](#), [28:296](#), [28:300](#), [29:618](#), [1280](#)
- `\math@version.` [1280](#)
- `\mathchar@type` [28:903](#),
[28:913](#), [28:964](#), [28:967](#), [28:976](#),
[28:992](#), [28:1097](#), [28:1108](#), [28:1171](#)
- `\mathph@nt` [38:99](#), [38:105](#)
- `\mathsm@sh`
[38:129](#), [38:138](#), [38:139](#), [38:145](#), [38:146](#)
- `\maybe@ic` [32:63](#), [32:64](#), [32:83](#)
- `\maybe@ic@` [32:83](#)
- `\maybe@icfalse` [32:97](#)
- `\maybe@ictrue` [32:87](#)
- `\maybe@load@fontshape` [21:89](#),
[25:2864](#), [25:2903](#), [26:128](#), [29:165](#), [482](#)
- `\maybe@update@bfseries@defaults` .
..... [29:39](#), [29:258](#), [29:267](#), [29:305](#)
- `\maybe@update@mdseries@defaults` .
..... [29:40](#), [29:281](#), [29:290](#), [29:306](#)
- `\mb@b` [40:76](#), [40:87](#), [40:95](#), [40:105](#)
- `\mb@l` [40:76](#), [40:80](#), [40:86](#),
[40:95](#), [40:99](#), [40:104](#), [42:137](#), [42:141](#)
- `\mb@r` [40:76](#), [40:80](#), [40:86](#),
[40:95](#), [40:99](#), [40:104](#), [42:137](#), [42:141](#)
- `\mb@t` [40:77](#), [40:84](#), [40:96](#), [40:103](#)
- `\md@def@ult` [29:398](#)
- `\mddef@ult` [29:199](#),
[29:294](#), [29:295](#), [29:296](#), [29:335](#),
[29:336](#), [29:337](#), [29:403](#), [29:444](#), [1338](#)
- `\mddefault@previous` [29:291](#), [29:293](#),
[29:332](#), [29:334](#), [30:118](#), [30:128](#), [1341](#)
- `\mdseries@..` [703](#)
- `\mdseries@previous` [1341](#)
- `\mdseries@rm` [29:112](#),
[29:128](#), [29:129](#), [29:230](#), [29:238](#),
[29:282](#), [29:294](#), [29:335](#), [29:340](#), [29:367](#)
- `\mdseries@sf` [29:113](#), [29:129](#), [29:239](#),
[29:283](#), [29:295](#), [29:336](#), [29:341](#), [29:368](#)
- `\mdseries@tt` [29:114](#), [29:129](#), [29:240](#),
[29:284](#), [29:296](#), [29:337](#), [29:342](#), [29:369](#)
- `\meaning` [314](#)
- `\merge@font@series` [25:2796](#), [25:2813](#),
[25:2814](#), [25:2895](#), [25:2897](#), [26:134](#), [620](#)
- `\merge@font@series@`
..... [25:2816](#), [25:2821](#), [25:2898](#)
- `\merge@font@series@without@substitution`
[25:2848](#), [25:2863](#), [25:2900](#), [26:137](#), [620](#)
- `\merge@font@series@without@substitution@`
..... [25:2848](#), [25:2901](#)
- `\merge@font@shape` [25:3146](#), [25:3159](#),
[25:3160](#), [25:3231](#), [25:3236](#), [26:133](#)
- `\merge@font@shape@`
..... [25:3162](#), [25:3167](#), [25:3237](#)
- `\merge@font@shape@without@substitution`
..... [25:3186](#), [25:3239](#), [26:136](#)
- `\merge@font@shape@without@substitution@`
..... [25:3186](#), [25:3240](#)
- `\mv@<version>` [671](#)
- `\mv@<version>@frozen` [668](#)
- `\mv@<version>@reset` [671](#)
- `\n@space`
... [30:636](#), [30:638](#), [30:640](#), [30:642](#),
[30:647](#), [30:648](#), [30:649](#), [30:650](#), [30:653](#)
- `\new@command` [06:78](#),
[06:77](#), [06:131](#), [06:165](#), [06:184](#), [06:251](#)
- `\new@environment` [06:147](#), [06:146](#), [06:159](#)
- `\new@fontshape` [27:2](#), [27:4](#), [27:22](#), [27:24](#)
- `\new@label@record`
..... [36:57](#), [36:128](#), [36:278](#), [37:25](#), [808](#)
- `\new@mathalphabet` [28:640](#), [28:661](#), [28:672](#)
- `\new@mathgroup` [04:27](#), [24:15](#), [28:490](#),
[02:78](#), [02:80](#), [02:98](#), [02:100](#), [1281](#)
- `\new@mathversion`
... [28:21](#), [28:423](#), [28:431](#), [28:436](#), [28:439](#)
- `\new@module@skip` [03:138](#), [03:150](#), [03:152](#)
- `\new@moduledate`
..... [03:82](#), [03:101](#), [03:104](#), [03:152](#)
- `\new@modulename` [03:96](#), [03:152](#)
- `\new@symbolfont` [28:491](#), [28:533](#)
- `\newcommand` [314](#)
- `\NewCommandCopy` [318](#)
- `\NewDocumentCommand` [311](#)
- `\newlinechar` [329](#)
- `\newmathalphabet@` [27:14](#)
- `\newmathalphabet@@` [27:109](#)
- `\newmathalphabet@@@` ... [27:15](#), [27:109](#)

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=ltthyphen.dtx, 57=ltfinal.dtx

- `\nfss@catcodes` 24:20, 24:121,
 24:484, 24:485, 24:492, 24:539,
 30:41, 30:47, 30:57, 30:146, *1309*
`\nfss@text` 21:335, 21:337, 29:649,
 32:5, 32:122, 35:17, 35:34, 35:51, *1303*
`\no@alphabet@error` 24:5,
 28:459, 28:461, 28:677, 28:678,
 28:692, 28:701, 28:787, 28:788, *1295*
`\no@alphabet@help` *1281*
`\no@version@warning` *1281*
`\noaccents@` 24:730, 30:140
`\noexpand` *331*
`\non@alpherr` 24:709,
 24:711, 28:73, 28:102, 28:118,
 28:175, 28:206, 28:236, 28:268, 28:1249
`\not@base` 29:721, 29:725, 29:726,
 29:727, 29:728, 29:729, 29:730,
 29:731, 29:732, 29:733, 29:734, 29:735
`\not@math@alphabet` 25:2915,
 25:2920, 25:2925, 25:3203, 25:3207,
 25:3210, 25:3213, 25:3216, 25:3219,
 25:3222, 25:3225, 29:6, 29:9, 29:12,
 29:15, 29:18, 29:21, 29:24, 29:27,
 29:30, 29:256, 29:279, 29:309,
 29:330, 29:354, 29:365, 29:382,
 29:385, 29:407, 29:412, 29:417,
 29:450, 29:455, 29:460, 29:476,
 29:479, 29:482, 29:485, 29:488, 29:598
`\now@and@everyjob`
 04:215, 04:221, 21:1033
`\o@lign` 21:410, 21:417,
 21:497, 21:505, 21:782, 21:789, 02:535
`\on@line` 08:2132,
 08:2146, 14:8, 14:15, 14:214, 29:145,
 29:148, 37:252, 37:273, 37:291,
 37:301, 40:189, 50:953, 50:1088,
 08:533, 08:608, 08:1479, 08:1514, *416*
`\operator@font` 30:654, 38:3, 38:4,
 38:5, 38:6, 38:7, 38:8, 38:9, 38:10,
 38:11, 38:12, 38:13, 38:14, 38:15,
 38:16, 38:17, 38:18, 38:19, 38:20,
 38:21, 38:22, 38:23, 38:24, 38:25,
 38:26, 38:27, 38:28, 38:29, 38:30,
 38:31, 38:32, 38:33, 38:34, 38:37, 38:40
`\optional@arg`
 26:416, 26:495, 26:497, 26:569, 26:572
`\outer@nobreak` 45:181,
 45:251, 45:255, 45:346, 45:364, *1308*
`\p@` 02:311
`\p@...` *1336*
`\p@enum` *875*
`\p@equation` 38:364, 38:518
`\p@renwd` *1291*
`\p@selectfont` 26:120, *1281*
`\par` *174*
`\par@deathcycles`
 39:56, 39:77, 39:79, 39:80
`\patch@level` 03:1,
 03:39, 03:44, 03:46, 03:48, 03:52,
 03:61, 57:664, 57:676, 57:678, *1331*
`\patchcmd` *311*
`\pdfannot@link@off@`
 54:794, 54:838, 54:853
`\pdfannot@link@on@`
 54:793, 54:846, 54:861
`\ph@nt` 38:81, 38:82, 38:83, 38:97
`\pickup@font` 21:199,
 24:258, 24:450, 24:644, 24:678,
 26:144, 26:171, 26:332, 26:334, 26:336
`\pictur@` 42:21
`\pkgcls@arg` 50:1611, 50:1742
`\pkgcls@candidate`
 50:1598, 50:1620, 50:1696,
 50:1700, 50:1704, 50:1773, 50:1776
`\pkgcls@debug` 50:1588, 50:1604,
 50:1605, 50:1606, 50:1607, 50:1608,
 50:1672, 50:1673, 50:1674, 50:1675,
 50:1684, 50:1689, 50:1707, 50:1716,
 50:1732, 50:1766, 50:1767, 50:1768
`\pkgcls@ext` 50:1612, 50:1660, *1355*
`\pkgcls@innerdate` 50:1593,
 50:1645, 50:1648, 50:1654, 50:1794
`\pkgcls@mindate` 50:1625,
 50:1634, 50:1650, 50:1655, *1084*
`\pkgcls@name` 50:1610, 50:1660
`\pkgcls@parse@date@arg`
 50:1619, 50:1630
`\pkgcls@parse@date@arg@`
 50:1636, 50:1639
`\pkgcls@parse@date@arg@version`
 50:1646, 50:1667
`\pkgcls@releasedate`
 50:1598, 50:1701, 50:1705, 50:1777
`\pkgcls@rollbackdate@error`
 50:1697, 50:1756, 50:1774
`\pkgcls@show@selection`
 50:1724, 50:1730, 50:1780, 50:1785
`\pkgcls@targetdate`
 50:1593, 50:1632, 50:1640,
 50:1643, 50:1644, 50:1648, 50:1656,
 50:1657, 50:1670, 50:1678, 50:1693,
 50:1695, 50:1725, 50:1737, 50:1739,
 50:1764, 50:1770, 50:1772, *1084*
`\pkgcls@targetlabel`
 50:1593, 50:1633,
 50:1653, 50:1668, 50:1680, 50:1712,
 50:1746, 50:1784, 50:1787, *1084*

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
 06=ltdefs.dtx, 07=ltcmd, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx,
 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
 56=lthyphen.dtx, 57=ltfinal.dtx

<code>\pkgcls@use@this@release</code>	<code>\relax</code>
..... 50:1681, 50:1698, 331
50:1700, 50:1713, 50:1723, 50:1776	<code>\rem@pt</code>
<code>\pr@@@s</code> 24:392
..... 38:259, 38:267	<code>\remove@angles</code>
<code>\pr@@@t</code> 26:348, 26:371
..... 38:262, 38:268	<code>\remove@nil</code>
<code>\pr@m@s</code> 28:37
..... 38:256, 38:257	<code>\remove@star</code>
<code>\preload@sizes</code> 26:348, 26:354
..... 27:11, 27:94, 1280	<code>\remove@tlig</code>
<code>\prepare@family@series@update</code> 21:1023,
..... 29:118,	21:1025, 21:1027, 21:1051, 21:1056,
29:142, 29:244, 29:408, 29:413,	21:1103, 21:1104, 21:1105, 1333
29:418, 29:451, 29:456, 29:461, 709	<code>\remove@to@nnil</code>
<code>\pretocmd</code> 24:391, 26:348, 26:374, 26:487
..... 310	<code>\renew@command</code>
<code>\prim@s</code> 06:125, 06:124, 06:185, 06:193
..... 38:253, 38:255, 38:267	<code>\renew@environment</code> ...
<code>\prime@s</code> 06:153, 06:152
..... 38:254	<code>\requested@test@context</code>
<code>\process@table</code> 29:504, 29:521, 29:525, 712
..... 20:40, 20:110, 20:168, 28:367	<code>\reserved@b</code>
<code>\propagate@doendpe</code> 686
..... 39:140, 870	<code>\reserved@a</code>
<code>\protectd@edf</code> 01:105, 03:13,
..... 1349	03:19, 03:34, 01:109, 01:110, 03:181,
<code>\protected</code>	03:182, 13:33, 13:37, 14:234, 18:456,
..... 289	18:459, 20:212, 20:213, 20:235,
<code>\protected@cmd</code>	20:244, 20:253, 20:306, 20:368,
..... 1313	20:415, 20:485, 20:487, 20:492,
<code>\protected@edef</code>	20:494, 20:506, 20:509, 20:512,
..... 22:245, 29:547, 35:130,	20:515, 20:516, 20:517, 20:539,
35:143, 35:150, 35:159, 35:161,	20:541, 20:546, 20:548, 20:549,
35:182, 35:186, 35:193, 35:197,	20:550, 20:561, 20:563, 20:568,
35:204, 36:40, 36:45, 36:83, 36:84,	20:570, 20:585, 20:591, 20:595,
06:222, 06:265, 40:486, 40:503,	20:602, 20:608, 20:612, 20:641,
40:520, 44:60, 45:472, 45:490,	20:644, 20:645, 20:647, 20:656,
45:508, 50:539, 50:914, 50:1181, 1306	20:659, 20:666, 20:669, 20:691,
<code>\protected@file@percent</code>	20:692, 20:693, 20:697, 20:705,
..... 37:197, 37:204, 37:219,	20:722, 20:723, 20:727, 20:733,
37:227, 37:228, 44:165, 44:172, 826	20:763, 20:767, 20:770, 20:775,
<code>\protected@wlog</code>	21:99, 21:100, 21:104, 21:107,
..... 50:386, 50:388, 50:402, 50:416	21:115, 21:125, 21:128, 21:137,
<code>\protected@write</code>	21:156, 21:161, 21:1017, 21:1021,
..... 20:202,	21:1065, 21:1067, 21:1068, 21:1070,
20:207, 35:105, 35:173, 36:55,	21:1072, 21:1080, 21:1089, 24:31,
44:189, 44:199, 46:14, 46:31, 1354	24:32, 24:33, 24:42, 24:45, 24:48,
<code>\protected@xdef</code>	24:64, 24:67, 24:70, 24:106, 24:109,
..... 06:265,	24:111, 24:148, 24:152, 24:486,
44:11, 45:453, 45:521, 45:537,	24:489, 24:616, 24:617, 24:632,
50:378, 50:397, 50:450, 50:811, 1350	24:635, 24:640, 24:651, 24:652,
<code>\provide@command</code>	24:665, 24:669, 24:674, 24:701,
..... 06:179, 06:178	24:704, 24:705, 24:713, 01:179,
<code>\ps@empty</code>	01:180, 01:183, 25:2823, 25:2824,
..... 49:10, 57:158	25:2827, 25:2828, 25:2843, 25:2844,
<code>\ps@plain</code>	25:2856, 25:2857, 01:201, 25:3168,
..... 49:13	25:3169, 25:3172, 25:3173, 25:3194,
<code>\q@curr@file</code>	25:3195, 01:205, 26:197, 26:199,
..... 50:1209, 50:1253,	26:201, 26:211, 26:213, 26:216,
50:1255, 50:1260, 50:1286, 50:1384,	26:345, 26:346, 26:359, 26:360,
50:1386, 50:1391, 50:1419, 1340	27:53, 27:57, 28:476, 28:479, 28:551,
<code>\quote@@name</code>	28:554, 28:587, 28:596, 28:598,
..... 20:462, 20:478	
<code>\quote@name</code>	
..... 20:462, 20:475,	
20:477, 20:588, 20:590, 20:599, 50:1384	
<code>\r@t</code>	
..... 38:66	
<code>\reenable@package@load</code>	
..... 52:477, 53:464, 1119	
<code>\reinstall@nfss@defs</code>	
..... 25:3205, 25:3246, 25:3250,	
25:3252, 25:3256, 25:3257, 25:3260	

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

28:642, 28:645, 28:655, 28:658,
 28:756, 28:758, 28:820, 28:824,
 28:868, 28:870, 28:931, 28:934,
 28:1026, 28:1028, 28:1122, 28:1124,
 28:1226, 28:1228, 28:1244, 28:1246,
 28:1247, 28:1252, 29:41, 29:42,
 29:74, 29:75, 29:175, 29:176, 01:227,
 05:22, 05:23, 05:24, 05:25, 05:26,
 29:517, 29:518, 05:47, 05:52, 01:234,
 05:89, 05:95, 05:105, 01:237, 01:239,
 01:240, 32:47, 32:48, 32:53, 32:54,
 32:65, 32:68, 32:88, 32:95, 01:247,
 01:250, 01:252, 01:253, 35:126,
 35:127, 35:128, 35:139, 35:140,
 35:141, 01:260, 35:159, 35:160,
 35:161, 35:162, 35:186, 35:187,
 35:197, 35:198, 36:40, 36:41, 36:45,
 36:46, 36:83, 36:85, 01:263, 01:265,
 37:195, 37:196, 06:117, 06:120,
 37:250, 37:251, 37:256, 37:271,
 37:272, 37:276, 37:289, 37:290,
 37:294, 37:299, 37:300, 37:304,
 06:133, 06:134, 06:135, 06:137,
 37:389, 37:390, 06:184, 06:185,
 06:186, 06:192, 06:193, 06:194,
 06:195, 06:198, 38:418, 38:419,
 38:420, 38:421, 38:423, 06:222,
 06:228, 06:238, 06:242, 40:78,
 40:79, 40:82, 40:97, 40:98, 40:101,
 40:184, 40:190, 06:300, 06:304,
 41:241, 41:245, 41:250, 41:269,
 06:338, 06:342, 41:360, 41:361,
 06:366, 06:370, 42:199, 42:201,
 42:205, 42:477, 42:478, 42:505,
 42:506, 42:534, 42:535, 45:29, 45:30,
 45:32, 45:33, 45:63, 45:67, 45:72,
 45:74, 45:76, 45:78, 45:83, 45:84,
 45:132, 45:136, 45:142, 45:145,
 45:148, 45:151, 06:481, 06:489,
 06:490, 01:315, 06:522, 06:530,
 06:531, 01:316, 06:532, 06:533,
 01:317, 06:573, 06:576, 06:591,
 06:593, 06:596, 06:605, 50:131,
 50:134, 50:136, 50:230, 50:238,
 50:242, 50:248, 50:256, 50:260,
 06:627, 06:634, 50:448, 50:450,
 50:451, 50:452, 50:456, 50:471,
 50:473, 50:474, 50:475, 50:479,
 50:656, 50:660, 50:666, 50:670,
 06:659, 50:748, 06:661, 50:749,
 50:752, 50:813, 50:817, 50:829,
 50:830, 50:832, 50:841, 50:845,
 50:857, 50:858, 50:860, 50:868,
 50:872, 50:884, 50:885, 50:886,
 50:888, 50:900, 50:903, 50:904,
 50:906, 50:990, 50:1039, 50:1056,
 50:1097, 50:1220, 50:1221, 50:1223,
 50:1355, 50:1356, 50:1358, 50:1400,
 50:1401, 50:1403, 50:1407, 50:1622,
 50:1627, 50:1679, 50:1680, 50:1711,
 50:1712, 50:1783, 50:1784, 50:1808,
 50:1810, 06:780, 06:789, 52:161,
 52:167, 52:168, 52:169, 54:33, 54:42,
 54:44, 54:46, 54:1151, 54:1171,
 54:2497, 54:2499, 54:2500, 54:2599,
 54:2601, 54:2607, 54:2610, 57:226,
 57:243, 57:244, 57:245, 57:252,
 57:253, 57:254, 57:489, 57:520,
 57:526, 57:527, 57:538, 57:539,
 57:540, 57:547, 57:548, 57:549,
 57:574, 57:575, 57:582, 57:586,
 57:588, 57:590, 57:595, 57:598,
 57:606, 57:607, 57:608, 57:677,
 57:680, 57:681, 57:698, 02:193, 1308
 \reserved@b 01:106, 01:107,
 13:33, 13:34, 13:37, 18:457, 18:458,
 18:465, 20:304, 20:306, 20:366,
 20:368, 20:413, 20:415, 20:586,
 20:588, 20:590, 20:603, 20:605,
 20:607, 20:691, 20:692, 20:693,
 20:766, 20:768, 20:769, 20:776,
 21:108, 21:115, 21:130, 21:137,
 21:1020, 21:1021, 21:1066, 21:1067,
 21:1089, 21:1098, 21:1100, 24:32,
 24:33, 24:39, 24:96, 24:98, 24:151,
 24:152, 24:702, 24:713, 25:2842,
 25:2843, 25:2845, 27:47, 27:54,
 27:71, 27:73, 28:477, 28:478, 28:479,
 28:483, 28:485, 28:552, 28:553,
 28:554, 28:558, 28:560, 28:595,
 28:596, 28:597, 28:632, 28:634,
 28:713, 28:715, 28:760, 28:761,
 28:762, 28:769, 28:929, 28:933,
 28:935, 29:182, 29:187, 29:191,
 29:192, 29:199, 29:200, 05:48, 05:54,
 32:52, 32:53, 32:66, 32:68, 32:95,
 32:96, 36:84, 36:85, 06:109, 06:111,
 06:118, 06:135, 06:136, 06:239,
 06:240, 06:242, 06:301, 06:302,
 06:304, 41:246, 41:248, 41:250,
 06:339, 06:340, 06:342, 06:367,
 06:368, 06:370, 45:43, 45:44, 45:112,
 45:113, 06:574, 06:576, 06:592,
 06:596, 50:231, 50:232, 50:233,
 50:235, 50:250, 50:253, 06:631,
 06:634, 50:448, 50:471, 50:821,
 50:827, 50:830, 50:849, 50:855,
 50:858, 50:876, 50:882, 50:886,

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
 06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpara.dtx,
 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
 56=ltthyphen.dtx, 57=ltfinal.dtx

- 50:900, 50:907, 50:1269, 50:1270,
50:1273, 50:1274, 50:1309, 50:1310,
50:1312, 50:1339, 50:1402, 50:1403,
50:1406, 50:1407, 50:1442, 50:1443,
50:1445, 50:1471, 50:1531, 50:1532,
50:1534, 50:1561, 06:781, 06:791,
54:1060, 54:1063, 54:1077, 54:1080,
54:1097, 54:1100, 57:229, 57:231,
57:235, 57:492, 57:494, 57:498,
57:578, 57:583, 57:624, [57:698](#), [534](#)
- `\reservedc`
... 01:107, 01:112, 20:756, 24:97,
24:98, 24:703, 24:706, 27:48, 27:55,
27:61, 27:68, 28:34, 28:38, 28:484,
28:485, 28:559, 28:560, 28:633,
28:634, 28:714, 28:715, 28:737,
28:746, 28:761, 28:775, 28:1016,
28:1033, 28:1042, 28:1070, 28:1081,
28:1139, 28:1152, 28:1154, 29:184,
29:187, 29:197, 29:198, 29:201,
29:202, 32:67, 32:69, 32:76, 50:901,
50:903, 50:904, 50:1254, 50:1259,
50:1260, 50:1278, 50:1286, 50:1292,
50:1314, 50:1322, 50:1385, 50:1390,
50:1391, 50:1411, 50:1419, 50:1425,
50:1447, 50:1454, 50:1502, 50:1503,
50:1504, 50:1514, 50:1536, 50:1543,
50:1571, 06:786, 06:789, 06:791,
06:794, 57:233, 57:238, 57:246,
57:496, 57:517, 57:518, 57:519,
57:521, 57:522, 57:523, 57:524,
57:525, 57:533, 57:541, 57:700, [1322](#)
- `\reservedd`
... 01:110, 01:113, 20:754, 20:756,
27:61, 27:68, 27:70, 27:74, 28:1024,
28:1033, 28:1042, 28:1078, 28:1081,
28:1147, 28:1152, 28:1156, 29:189,
29:190, 29:195, 29:196, 05:46,
05:51, 06:779, 06:788, 57:701, [1338](#)
- `\reservede`
18:58, 18:60, 18:70, 18:72, 18:101,
18:108, 18:116, 27:39, 27:45, 27:70,
27:73, 27:74, 28:35, 28:40, 57:702, [1315](#)
- `\reservedf`
18:59, 18:60, 18:71, 18:72, 18:116,
21:1565, 21:1566, 21:1567, 21:1568,
21:1570, 21:1581, 24:253, 24:255,
24:261, 24:262, 26:383, 26:394,
26:398, 26:402, 26:408, 26:411,
26:450, 26:487, 26:490, 27:27, 27:38,
27:45, 27:71, 27:73, 57:703, [1308](#)
- `\reset@font`
... 29:220, [29:654](#), 29:685, 29:700,
29:715, 35:17, 35:34, 35:51, 40:482,
40:500, 40:517, 45:175, 45:371,
45:466, 45:485, 45:503, 47:40,
49:14, 54:824, 54:894, 54:953, [1284](#)
- `\restglb@settings` 26:269, [26:279](#)
- `\restore@mathversion`
..... 28:108, 28:111, 28:126, 28:134
- `\restore@protect` [06:265](#)
- `\rlh@` 30:475, 30:476
- `\rm@def@ult` [29:398](#)
- `\rmdef@ult` 29:259, 29:282,
29:320, 29:340, 29:356, 29:367,
29:399, 29:440, 33:27, 33:48, [704](#)
- `\robust@command@act`
..... 09:107, 06:422, 06:423,
[06:425](#), 06:493, 06:550, 06:687, [96](#)
- `\robust@command@act@chk@args` ...
..... [06:447](#), [06:468](#), [97](#)
- `\robust@command@act@do`
..... [06:433](#), [06:465](#), [97](#)
- `\robust@command@act@end` .. 06:430,
[06:431](#), [06:443](#), [06:446](#), [06:466](#), [97](#)
- `\robust@command@act@loop`
..... [06:427](#), [06:433](#), [06:463](#), [97](#)
- `\robust@command@act@loop@aux` ...
..... [06:433](#), [06:464](#)
- `\robust@command@chk@safe`
..... 09:154, 06:315, 06:426,
[06:447](#), 06:467, 06:607, 06:619, [94](#)
- `\s@fct@` 26:427, [26:491](#)
- `\s@fct@alias` 26:553
- `\s@fct@fixed` 26:566
- `\s@fct@gen` 26:503
- `\s@fct@genb` 26:508
- `\s@fct@sgen` 26:503
- `\s@fct@sgenb` 26:508
- `\s@fct@sub` 26:515
- `\s@fct@subf` 26:558
- `\saved@reqcolroom`
.. 54:1377, 54:1834, [54:2523](#), 54:2651
- `\saved@space@catcode` .. 57:355, 57:424
- `\scan@fontshape` ... 27:7, 27:40, [27:43](#)
- `\scan@fontshape` ... 27:6, 27:26, [27:37](#)
- `\scantokens` [314](#)
- `\scriptfont@name` 26:334, 26:339
- `\section` [311](#)
- `\select@group` 24:686, 24:705,
[28:49](#), 28:403, 28:464, 28:642,
28:695, 28:704, 28:742, 28:774, [668](#)
- `\series@change@debug`
..... 29:138, 29:145, 29:148,
29:159, 29:162, 29:166, 29:178,
29:186, 29:191, 29:197, 29:200, 29:202
- `\series@check@toks`
..... 25:2878, 25:2880, 25:2887

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- `\series@drop@one@m`
..... 25:2884, 25:2888, 25:2906
- `\series@maybe@drop@one@m`
..... 24:32, 25:2871,
25:2873, 25:2905, 28:478, 28:553,
29:169, 29:188, 29:194, 29:402, 29:403
- `\series@maybe@drop@one@m@x`
..... 25:2874, 25:2876
- `\seriesdefault@kernel`
..... 29:221, 29:774, 721
- `\set@mathdelimiter` . 28:1079, 28:1113
- `\set@color` 40:109
- `\set@curr@file` ... 20:226, 20:235,
20:262, 20:270, 20:452, 20:470,
50:932, 50:1252, 50:1383, 52:266, 1115
- `\set@curr@file@aux`
..... 52:272, 52:278, 52:279
- `\set@curr@file@nosearch` . 52:266, 1348
- `\set@display@protect`
..... 14:7, 14:14, 14:34,
14:61, 06:8, 06:16, 06:263, 50:389, 1306
- `\set@fontsize` 24:380,
24:382, 26:122, 26:168, 26:178, 1298
- `\set@mathaccent`
..... 28:829, 28:837, 28:872, 28:880, 28:898
- `\set@mathchar` 28:953, 28:963
- `\set@mathdelimiter`
..... 28:1030, 28:1039, 28:1091
- `\set@mathradical` 28:411, 28:1149
- `\set@mathsymbol` 28:937, 28:945, 28:966
- `\set@simple@size@args`
..... 26:349, 26:362, 26:369, 26:390, 26:404
- `\set@size@funct@args`
..... 26:352, 26:354, 26:412
- `\set@size@funct@args@` 26:412
- `\set@target@series` 24:351, 24:363,
25:2825, 25:2829, 25:2832, 25:2835,
25:2858, 25:2860, 25:2869, 25:2904
- `\set@typeset@protect`
... 06:263, 06:282, 41:197, 41:235,
53:84, 53:128, 54:806, 54:808,
54:881, 54:883, 54:939, 54:941, 1306
- `\SetMathAlphabet@` 28:649, 28:718, 28:727
- `\SetSymbolFont@` 28:509, 28:563, 28:581
- `\SetSymbolFont@m@dropped`
..... 28:551, 28:556, 28:576, 28:577
- `\sf@def@ult` 29:398
- `\sf@size` 19:6, 21:322, 24:287,
24:306, 24:725, 26:329, 26:333,
33:636, 45:409, 45:416, 45:437, 45:444
- `\sfdef@ult` 29:260, 29:283,
29:321, 29:341, 29:357, 29:368,
29:400, 29:441, 33:29, 33:50, 704
- `\sh@ft` 02:539, 1309
- `\show@kernel@robust@command`
..... 06:613, 06:676, 06:704
- `\show@release@info`
... 03:38, 03:41, 03:46, 03:51, 37:45,
37:46, 37:48, 37:110, 37:111, 37:113, 38
- `\ShowCommand` 318
- `\sixt@n` 04:30, 24:15, 28:85,
28:249, 28:815, 28:817, 28:863,
28:865, 28:924, 28:926, 28:972,
28:974, 28:1012, 28:1014, 28:1020,
28:1022, 28:1066, 28:1068, 28:1074,
28:1076, 28:1135, 28:1137, 28:1143,
28:1145, 42:278, 42:293, 42:295,
45:62, 45:80, 45:131, 45:153, 02:16,
54:1279, 54:1325, 54:1465, 54:1610,
54:1777, 54:2092, 54:2156, 54:2213,
54:2283, 54:2553, 54:2562, 54:2618,
54:2634, 02:64, 54:2672, 54:2722,
02:66, 02:96, 02:97, 02:98, 01:55
- `\sixt@n_` 391
- `\size@update`
..... 26:147, 26:173, 26:186, 26:205, 26:207
- `\sizefn@info`
..... 26:353, 26:355, 26:363, 26:391, 26:405
- `\skip@` 32:105, 32:108,
02:41, 02:501, 02:503, 02:504, 02:506
- `\sp@ce@skip` 18:84, 18:517, 18:518
- `\sp@n` 41:379
- `\split@name` 24:454,
24:466, 24:580, 24:595, 26:520, 26:534
- `\ssf@size` .. 21:348, 21:380, 21:1276,
24:288, 24:307, 24:726, 26:329, 26:335
- `\string@makeletter` 50:906, 50:941,
50:942, 50:943, 06:806, 52:169, 1341
- `\strip@meaning` 1291
- `\strip@prefix`
... 01:95, 24:683, 01:212, 06:240,
06:302, 06:340, 06:368, 01:307, 06:803
- `\strip@pt` ... 24:279, 24:285, 24:286,
24:287, 24:288, 24:301, 24:305,
24:392, 24:725, 24:726, 26:181, 02:543
- `\sub@sf@cnt` 26:515, 26:516, 26:517, 26:544
- `\subf@sf@cnt` 26:558, 26:559, 26:560
- `\subst@correction` 24:86, 24:92
- `\subst@fontshape` 27:8, 27:80
- `\subst@size` 26:466
- `\sw@slant` 32:91, 32:101
- `\t@st@ic` 32:90, 32:94, 1286
- `\target@meta@family@value`
..... 29:151, 29:176, 29:183, 29:185
- `\target@series@value`
..... 29:150, 29:158, 29:161,
29:163, 29:167, 29:168, 29:169,
29:192, 29:198, 29:199, 29:201, 708

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

<code>\tc@check@accent</code>	<code>\text@command</code>
... 33:73 , 33:125 , 33:127 , 33:129 ,	32:8 , 32:46 , 1303
33:131 , 33:133 , 33:135 , 33:137 ,	<code>\textfont@name</code>
33:139 , 33:141 , 33:143 , 33:145 ,	26:332 , 26:338
33:147 , 33:149 , 33:151 , 33:153 ,	<code>\tf@size</code>
33:155 , 33:899 , 33:975 , 33:977 , 33:979	24:286 ,
<code>\tc@check@symbol</code> ... 33:73 , 33:175 ,	24:306 , 24:724 , 26:329 , 26:331 , 1284
33:177 , 33:179 , 33:181 , 33:183 ,	<code>\thr@@</code>
33:185 , 33:187 , 33:189 , 33:191 ,	02:563 ,
33:193 , 33:195 , 33:197 , 33:199 ,	02:573 , 02:607 , 26:59 , 26:255 ,
33:201 , 33:203 , 33:205 , 33:207 ,	26:261 , 26:274 , 26:281 , 26:288 ,
33:209 , 33:211 , 33:213 , 33:215 ,	26:293 , 38:376 , 38:539 , 39:263 ,
33:217 , 33:219 , 33:221 , 33:223 ,	39:274 , 42:287 , 42:288 , 42:290 ,
33:225 , 33:227 , 33:229 , 33:231 ,	42:291 , 42:329 , 42:355 , 42:388 ,
33:233 , 33:235 , 33:237 , 33:239 ,	42:411 , 02:16 , 57:84 , 57:92 , 1312
33:241 , 33:243 , 33:245 , 33:247 ,	<code>\toks@</code>
33:249 , 33:251 , 33:253 , 33:255 ,	03:94 , 03:106 ,
33:257 , 33:259 , 33:261 , 33:263 ,	03:108 , 03:115 , 03:119 , 03:122 ,
33:265 , 33:267 , 33:269 , 33:271 ,	03:127 , 18:455 , 18:456 , 18:461 ,
33:274 , 33:276 , 33:278 , 33:280 ,	24:149 , 24:153 , 24:155 , 24:158 ,
33:282 , 33:284 , 33:286 , 33:288 ,	24:284 , 24:289 , 28:7 , 28:8 , 28:450 ,
33:290 , 33:292 , 33:294 , 33:296 ,	28:454 , 28:460 , 28:463 , 28:468 ,
33:298 , 33:300 , 33:302 , 33:304 ,	28:534 , 28:535 , 28:537 , 28:538 ,
33:306 , 33:308 , 33:310 , 33:312 ,	28:588 , 28:590 , 28:594 , 28:611 ,
33:314 , 33:318 , 33:320 , 33:322 ,	28:614 , 28:673 , 28:685 , 28:686 ,
33:324 , 33:326 , 33:328 , 33:330 ,	28:687 , 28:733 , 28:735 , 28:741 ,
33:899 , 33:969 , 33:971 , 33:973 ,	28:749 , 28:753 , 28:765 , 28:768 ,
33:981 , 33:983 , 33:985 , 33:987 ,	28:771 , 28:779 , 28:781 , 28:1202 ,
33:989 , 33:991 , 33:993 , 33:995 ,	28:1204 , 28:1206 , 28:1209 , 28:1211 ,
33:997 , 33:999 , 33:1001 , 33:1003 ,	28:1214 , 28:1217 , 28:1249 , 28:1250 ,
33:1005 , 33:1007 , 33:1009 , 33:1011 ,	50:501 , 50:502 , 50:504 , 50:505 ,
33:1013 , 33:1015 , 33:1017 , 33:1019 ,	06:920 , 06:921 , 02:41 , 54:2836 ,
33:1021 , 33:1023 , 33:1025 , 33:1027 ,	54:2837 , 54:2838 , 54:2839 , 1311
33:1029 , 33:1031 , 33:1033 , 33:1035 ,	<code>\topmark</code>
33:1037 , 33:1039 , 33:1041 , 33:1043 ,	1010
33:1045 , 33:1047 , 33:1049 , 33:1051 ,	<code>\topmark(s)</code>
33:1053 , 33:1055 , 33:1057 , 33:1059 ,	1010
33:1061 , 33:1063 , 33:1065 , 33:1067 ,	<code>\try@load@font@shape</code>
33:1069 , 33:1071 , 33:1073 , 33:1075 ,	1323
33:1077 , 33:1079 , 33:1081 , 33:1083	<code>\try@load@fontshape</code>
<code>\tc@error</code> 24:469 , 24:477 , 24:523 , 24:628 ,
33:74 , 33:879 , 33:900	25:2867 , 26:521 , 28:375 , 28:392 , 621
<code>\tc@errorwarn</code>	<code>\try@simple@size</code>
33:21 ,	26:357 , 26:482
33:40 , 33:42 , 33:787 , 33:789 , 33:791 ,	<code>\try@simples</code> ...
33:792 , 33:838 , 33:839 , 33:840 , 33:873	26:440 , 26:446 , 26:450
<code>\tc@fake@euro</code>	<code>\try@size@range</code>
... 33:61 , 33:316 , 33:887 , 33:968	26:102 , 26:357 , 26:433
<code>\tc@forcedfalse</code>	<code>\try@size@substitution</code>
33:832	26:104 , 26:437
<code>\tc@forcedtrue</code>	<code>\tryif@simple</code>
33:837	26:448 , 26:449
<code>\tc@oldstylesubst</code>	<code>\tryis@simple</code>
33:16 , 33:20	26:449
<code>\tc@subst</code> ..	<code>\tt@def@ult</code>
33:41 , 33:73 , 33:872 , 33:899	29:261 , 29:284 ,
<code>\tc@swap@accent</code>	29:322 , 29:342 , 29:358 , 29:369 ,
33:76 , 33:77	29:401 , 29:442 , 33:31 , 33:52 , 704
<code>\test@font@series@context</code>	<code>\tw@</code>
... 29:506 , 29:516 , 29:536 , 712	02:16 , 1295
<code>\test@next</code>	<code>\two@digits</code>
1314	01:169 ,
	01:170 , 26:513 , 06:2 , 50:1205 ,
	50:1298 , 50:1431 , 50:1520 , 01:70
	<code>\type@restoreinfo</code>
	26:203 , 26:208
	<code>\unconditionally@reset@math@version</code>

	24:405 , 24:413
	<code>\undeclare@...</code>
	1149
	<code>\undeclare@file@substitution</code> ...

	52:247 , 1104

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=ltparams.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltparam.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

<code>\unexpandable@noexpand</code>	1317	<code>\verb@egroup</code>	37:654,
<code>\unexpanded</code>	1005		37:657, 37:671, 37:674, 37:685,
<code>\unqu@tefilef@und</code>	52:143		37:688, 37:700, 37:703, 37:709, 37:713
<code>\unquote@name</code>		<code>\verb@eol@error</code> 37:710, 37:722, 37:732	
	20:456, 20:462, 20:479, 52:349	<code>\verbatim@font</code>	37:542,
<code>\unrestored@protected@xdef</code> 06:265,			37:563, 37:571, 37:723, 37:733, 1315
45:458, 45:526, 45:542, 45:553,		<code>\verbatim@nolig@list</code> .	37:738, 37:744
48:239, 49:41, 49:67, 49:90, 49:123		<code>\verbatim@out</code>	1322
<code>\unskip</code>	1016	<code>\version@elt</code>	28:19,
<code>\update@series@target@value</code>			28:32, 28:33, 28:447, 28:448, 28:507,
	29:119, 29:153, 29:174, 29:245		28:537, 28:648, 28:686, 28:778, 28:1207
<code>\update@ucl@with@cyrillic</code>		<code>\version@list</code>	
	21:1527, 21:1555, 21:1589, 21:1597		28:17, 28:22, 28:33, 28:440,
<code>\upper@bound</code>			28:448, 28:512, 28:543, 28:582,
	26:384, 26:385, 26:386, 26:399		28:653, 28:698, 28:728, 28:783, 28:1220
<code>\use@mathgroup</code>	24:692, 24:710,	<code>\vgl@</code>	02:501, 02:502
	24:712, 26:300, 28:64, 28:93, 28:655,	<code>\voidb@x</code>	23:23, 23:29, 02:311, 02:532
	28:757, 28:760, 28:1227, 28:1251, 1281	<code>\vsplit</code>	1015
<code>\UTF@two@octets@noexpand</code>	1348	<code>\warn@rel@i</code> 27:5, 27:25, 27:29, 27:81,	
<code>\UTFviii@four@octets</code>			27:85, 27:90, 27:95, 27:119, 27:140
	57:410, 57:415, 57:421	<code>\wrong@fontshape</code> .	24:473, 24:610, 1280
<code>\UTFviii@four@octets@@</code> 57:410, 57:421		<code>\x@protect</code>	06:243, 06:254, 06:305,
<code>\UTFviii@four@octets@combine</code> 57:445			06:343, 06:371, 06:577, 06:597, 1306
<code>\UTFviii@four@octets@noexpand</code> 57:451		<code>\xe@alloc@</code>	57:56, 57:66
<code>\UTFviii@four@octets@string</code> .	57:448	<code>\xe@alloc@intercharclass</code> .	57:35, 1330
<code>\UTFviii@invalid</code>	57:349, 57:442	<code>\xe@ch@ck</code>	57:57, 57:61
<code>\UTFviii@invalid@err</code>		<code>\XXX@argdef</code>	1293
	57:407, 57:412, 57:418	<code>\z@</code>	02:311, 1317
<code>\UTFviii@invalid@err@@</code> 57:407, 57:418		<code>\z@skip</code>	02:311, 423
<code>\UTFviii@three@octets</code>		<code>\zap@space</code>	20:262, 20:282,
	57:409, 57:414, 57:420		29:547, 47:49, 50:231, 50:453,
<code>\UTFviii@three@octets@@</code> 57:409, 57:420			50:476, 50:488, 50:655, 50:771,
<code>\UTFviii@three@octets@combine</code> 57:444			50:811, 50:829, 50:840, 50:857,
<code>\UTFviii@three@octets@noexpand</code> 57:450			50:867, 50:884, 50:916, 50:1218, 50:1353
<code>\UTFviii@three@octets@string</code> 57:447		<code>\zref@labelbyprop</code>	810
<code>\UTFviii@two@octets</code>		tex commands:	
	57:408, 57:413, 57:419	<code>\tex_afterassignment:D</code>	
<code>\UTFviii@two@octets@@</code> 57:408, 57:419			53:49, 53:138, 07:2796
<code>\UTFviii@two@octets@combine</code> .	57:443	<code>\tex_aftergroup:D</code> .	53:57, 53:144, 1133
<code>\UTFviii@two@octets@noexpand</code> 57:449		<code>\tex_currentgrouplevel:D</code>	
<code>\UTFviii@two@octets@string</code> . .	57:446		10:33, 53:48, 53:56, 53:137, 53:143
<code>\UTFviii@undefined@err</code>		<code>\tex_deadcycles:D</code>	53:91
	57:406, 57:411, 57:417	<code>\tex_endlinechar:D</code>	
<code>\UTFviii@undefined@err@@</code>			09:464, 07:1603, 07:2084,
	57:406, 57:417		07:2186, 07:2187, 07:2197, 07:2198,
<code>\v@false</code>	38:82		07:2205, 07:2227, 07:2242, 1352
<code>\v@true</code>	38:81, 38:83	<code>\tex_escapechar:D</code>	
<code>\vbox</code>	1016		09:219, 09:294, 48:106, 48:107,
<code>\ver@...</code>	473		48:109, 52:212, 52:226, 07:1131,
<code>\ver@@...</code>	473		07:1142, 07:1350, 07:1360, 07:1602,
<code>\ver@{file}.{ext}</code>	1065		07:2083, 07:2670, 07:2696, 08:1165, 184
<code>\verb@balance@group</code>	37:654,	<code>\tex_everypar:D</code>	
	37:656, 37:671, 37:673, 37:685,		16:20, 16:24, 16:29, 16:53, 16:57,
	37:687, 37:700, 37:702, 37:708, 37:709		

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=ltthyphen.dtx, 57=ltfinal.dtx

- 16:61, 16:76, 16:76, 16:116, 16:118,
16:128, 16:129, 16:180, 16:181, 421
- `\tex_gdef:D` 09:209, 09:284, 324
- `\tex_hskip:D` 16:102
- `\tex_indent:D` 16:123
- `\tex_interactionmode:D`
. 48:104, 48:105, 48:110
- `\tex_lastnodetype:D` 16:101, 48:80, 424
- `\tex_lastxpos:D` 36:243
- `\tex_lastypos:D` 36:244
- `\tex_lowercase:D` 07:2493, 07:2676
- `\tex_marks:D` 48:247
- `\tex_newlinechar:D` 09:465
- `\tex_noindent:D` 16:27, 16:59, 16:133, 419
- `\tex_par:D`
. 16:18, 16:51, 16:104, 16:111,
16:135, 16:176, 16:177, 16:178, 422
- `\tex_parskip:D` . . . 16:25, 16:26, 16:58
- `\tex_savepos:D` 806
- `\tex_setbox:D` . . . 48:111, 53:50, 53:139
- `\tex_shipout:D` . 53:126, 53:380, 53:510
- `\tex_splitbotmarks:D`
. 48:130, 48:161, 1018
- `\tex_splitfirstmarks:D`
. 48:142, 48:180, 48:189, 1018
- `\tex_splitmaxdepth:D` 48:68
- `\tex_unskip:D` 16:97, 48:78, 423
- `\tex_vbadness:D` 48:69
- `\tex_vfuzz:D` 48:70
- `\tex_vsplit:D` 48:111, 1018
- `\tex_vss:D` 53:334
- `\tex_xdef:D` 09:215, 09:290, 324
- `\text` 1283
- text commands:
- `\l_text_case_exclude_arg_tl` . 57:627
- `\text_case_switch:nnnn` 57:630
- `\text_declare_case_equivalent:Nn`
. 57:632
- `\text_declare_lowercase_mapping:nn`
. 57:636
- `\text_declare_lowercase_mapping:nnn`
. 57:637
- `\text_declare_titlecase_mapping:nn`
. 57:643
- `\text_declare_titlecase_mapping:nnn`
. 57:644
- `\text_declare_uppercase_mapping:nn`
. 57:650
- `\text_declare_uppercase_mapping:nnn`
. 57:651
- `\text_lowercase:n` 1348
- `\text_titlecase_all:nn` 57:609
- `\text_titlecase_first:nn` 57:610
- `\textacutedbl` 21:937, 21:1175,
33:198, 33:199, 33:685, 33:932, 33:1313
- `\textascendercompwordmark`
21:876, 33:118, 33:649, 33:915, 33:1344
- `\textasciicircum` 21:987, 21:1136,
33:200, 33:201, 33:686, 33:956, 33:1332
- `\textasciibreve` 21:935, 21:1174,
33:202, 33:203, 33:687, 33:929, 33:1310
- `\textasciicaron` 21:936, 21:1173,
33:204, 33:205, 33:688, 33:930, 33:1311
- `\textasciicircum`
. 21:306, 21:579, 21:1110, 1333
- `\textasciidieresis` . . 21:975, 21:1123,
33:206, 33:207, 33:689, 33:946, 33:1329
- `\textasciigrave` 21:926, 21:1104,
33:208, 33:209, 33:690, 33:927, 33:1301
- `\textasciimacron` . . . 21:982, 21:1131,
33:210, 33:211, 33:691, 33:951, 33:1312
- `\textasciitilde`
. 21:307, 21:580, 21:1115, 1317
- `\textasteriskcentered`
. . . 21:287, 21:739, 21:886, 21:887,
21:1237, 22:217, 22:223, 33:84,
33:541, 33:606, 33:922, 33:1347, 1321
- `\textbackslash`
21:288, 21:581, 21:740, 21:1109, 1310
- `\textbaht` 21:961, 21:1178, 33:216, 33:217,
33:695, 33:1062, 33:1063, 33:1323
- `\textbar`
21:289, 21:582, 21:741, 21:1113, 1317
- `\textbardbl` 21:290,
21:742, 21:941, 21:1192, 22:222,
33:91, 33:329, 33:330, 33:542,
33:614, 33:761, 33:935, 33:1228, 1326
- `\textbf` 32:19, 36:238, 699
- `\textbigcircle` 21:751,
21:914, 21:1254, 33:218, 33:219,
33:696, 33:1014, 33:1015, 33:1298, 1327
- `\textblank` . . . 21:883, 21:1251, 33:309,
33:310, 33:749, 33:984, 33:985, 33:1235
- `\textborn` 21:927, 33:220, 33:221, 33:351,
33:697, 33:1020, 33:1021, 33:1302
- `\textbraceleft` 21:291,
21:328, 21:583, 21:743, 21:1112, 1304
- `\textbraceright` 21:292,
21:329, 21:584, 21:744, 21:1114, 1304
- `\textbrokenbar` 21:973,
21:1121, 33:92, 33:615, 33:944, 33:1356
- `\textbullet`
. . 21:293, 21:745, 21:943, 21:1201,
33:85, 33:543, 33:607, 33:937, 33:1351
- `\textcapitalcompwordmark`
21:875, 33:117, 33:648, 33:914, 33:1343

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx,
16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
56=lthyphen.dtx, 57=ltfinal.dtx

- `\textcelsius` 21:944, 21:1220, 33:93, 33:317,
 33:318, 33:616, 33:755, 33:938, 33:1229
`\textcent` 21:969,
 21:1117, 33:94, 33:617, 33:941, 33:1353
`\textcentoldstyle` 21:946,
 33:222, 33:223, 33:356, 33:359,
 33:698, 33:1036, 33:1037, 33:1316, 766
`\textcircled` ... 21:299, 21:303, 21:320,
 21:321, 21:752, 21:915, 33:119,
 33:120, 33:619, 33:635, 33:651,
 33:819, 33:1084, 33:1086, 33:1337, 1309
`\textcircledP`
 .. 21:980, 21:1222, 33:224, 33:225,
 33:699, 33:1078, 33:1079, 33:1331
`\textcolonmonetary`
 .. 21:948, 21:1213, 33:277, 33:278,
 33:729, 33:1038, 33:1039, 33:1259
`\textcommaabove`
 ... 21:373, 21:375, 21:389, 21:390,
 21:479, 21:480, 21:721, 21:722, 1330
`\textcommabelow`
 ... 21:344, 21:346, 21:352, 21:353,
 21:724, 21:725, 21:726, 21:727,
 21:728, 21:729, 21:730, 21:731,
 21:732, 21:733, 21:1274, 21:1477,
 21:1478, 21:1479, 21:1480, 1330
`\textcompsubstdefault` 33:34,
 33:39, 33:55, 33:582, 33:877, 1337
`\textcompwordmark`
 21:310, 21:311, 21:585, 21:1180, 1322
`\textcopyleft`
 ... 21:978, 33:226, 33:227, 33:332,
 33:700, 33:1076, 33:1077, 33:1330
`\textcopyright`
 .. 21:303, 21:337, 21:976, 21:1124,
 33:95, 33:618, 33:947, 33:1358, 1322
`\textcurrency` 21:971,
 21:1119, 33:299, 33:300, 33:742,
 33:814, 33:818, 33:972, 33:973, 33:1248
`\textdagger`
 ... 21:295, 21:332, 21:747, 21:939,
 21:1199, 22:218, 22:224, 33:87,
 33:545, 33:609, 33:933, 33:1348, 1305
`\textdaggerdbl`
 ... 21:294, 21:333, 21:746, 21:940,
 21:1200, 22:219, 22:225, 33:86,
 33:544, 33:608, 33:934, 33:1349, 1305
`\textdblhyphen` . 21:898, 33:230, 33:231,
 33:333, 33:702, 33:986, 33:987, 33:1286
`\textdblhyphenchar`
 ... 21:934, 33:228, 33:229, 33:334,
 33:701, 33:1032, 33:1033, 33:1309
`\textdegree` 21:983,
 21:1132, 33:96, 33:620, 33:952, 33:1362
`\textdied` 21:929, 33:232, 33:233, 33:352,
 33:703, 33:1024, 33:1025, 33:1304
`\textdiscount`
 .. 21:963, 21:1212, 33:234, 33:235,
 33:704, 33:1066, 33:1067, 33:1324
`\textdiv` 21:1000,
 21:1157, 33:97, 33:621, 33:966, 33:1371
`\textdivorced`
 .. 21:928, 21:1257, 33:236, 33:237,
 33:705, 33:1022, 33:1023, 33:1303
`\textdollar` ... 21:275, 21:327, 21:458,
 21:586, 21:820, 21:884, 21:1106,
 33:78, 33:79, 33:587, 33:589,
 33:920, 33:1092, 33:1094, 33:1345, 1304
`\textdollaroldstyle` 21:945,
 33:238, 33:239, 33:357, 33:358,
 33:706, 33:1034, 33:1035, 33:1315, 766
`\textdong` 21:957, 21:1217, 33:279, 33:280,
 33:730, 33:1056, 33:1057, 33:1262
`\textdownarrow`
 .. 21:925, 21:1232, 33:281, 33:282,
 33:731, 33:1018, 33:1019, 33:1258
`\texteightoldstyle`
 ... 21:908, 33:178, 33:179, 33:348,
 33:674, 33:1004, 33:1005, 33:1295
`\textellipsis` ... 21:316, 21:341, 21:1202
`\textemdash` ... 21:276, 21:427, 21:431,
 21:587, 21:591, 21:807, 21:1184, 1304
`\textendash` ... 21:277, 21:428, 21:430,
 21:588, 21:590, 21:808, 21:1183, 1304
`\textestimated`
 .. 21:964, 21:1228, 33:293, 33:294,
 33:738, 33:817, 33:970, 33:971, 33:1252
`\texteuro` 21:998, 21:1218, 33:315, 33:316,
 33:753, 33:815, 33:967, 33:968, 33:1238
`\textexclamdown` 21:278, 21:432,
 21:434, 21:592, 21:809, 21:1116, 1327
`\textfiguredash`
 21:430, 21:590, 21:1182, 21:1188, 1343
`\textfiveoldstyle`
 21:905, 33:180, 33:181,
 33:345, 33:675, 33:998, 33:999, 33:1292
`\textfloatsep` 54:1002, 54:1015,
 54:2653, 54:2703, 54:2753, 54:2927
`\textflorin` 21:947, 21:1171,
 33:297, 33:298, 33:741, 33:939, 33:1247
`\textfont` 26:338, 38:251
`\textfouroldstyle`
 21:904, 33:182, 33:183,
 33:344, 33:676, 33:996, 33:997, 33:1291
`\textfraction` 54:2450, 54:2453, 54:2478,
 54:2481, 54:2640, 54:2921, 1323

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
 06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx,
 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
 56=ltthyphen.dtx, 57=ltfinal.dtx

- `\textfractionsolidus`
 .. 21:899, 21:1209, 33:301, 33:302,
 33:744, 33:923, 33:1241, 33:1473, 768
`\textgravedbl` 21:938, 21:1176,
 33:212, 33:213, 33:692, 33:931, 33:1314
`\textgreater`
 21:301, 21:593, 21:762, 21:1108, 1304
`\textguarani`
 ... 21:951, 33:240, 33:241, 33:355,
 33:707, 33:1044, 33:1045, 33:1318
`\textheight` 20:22,
 20:23, 20:92, 20:93, 20:149, 20:150,
 45:257, 45:258, 45:261, 45:287,
 45:301, 53:380, 54:74, 54:223,
 54:224, 54:272, 54:397, 54:445,
 54:474, 54:865, 54:923, 54:982,
 54:1037, 54:1089, 57:156, 57:157, 1295
`\texthorizontalbar`
 21:431, 21:591, 21:1185, 21:1190, 1343
`\texthyphen`
 . 21:280, 21:437, 21:595, 21:811, 1310
`\texthyphenchar`
 . 21:279, 21:436, 21:594, 21:810, 1304
`\textindent` 1311
`\textinterrobang`
 .. 21:955, 21:1208, 33:313, 33:314,
 33:751, 33:1052, 33:1053, 33:1236
`\textinterrobangdown`
 .. 21:956, 21:1258, 33:311, 33:312,
 33:750, 33:1054, 33:1055, 33:1237
`\textit` 32:21
`\textlangle`
 .. 21:910, 21:1249, 33:273, 33:274,
 33:726, 33:1008, 33:1009, 33:1265
`\textlbrace` 1305
`\textlbrackdbl` 21:922, 33:174,
 33:175, 33:353, 33:671, 33:925, 33:1299
`\textleaf` 21:930, 33:242, 33:243, 33:337,
 33:708, 33:1026, 33:1027, 33:1305
`\textleftarrow` 21:881, 21:1229, 33:283,
 33:284, 33:732, 33:980, 33:981, 33:1255
`\textlegacyasteriskcentered`
 33:553, 33:764
`\textlegacybardbl` 33:553, 33:765
`\textlegacybullet` 33:553, 33:766
`\textlegacydagger` 33:553, 33:768
`\textlegacydaggerdbl` 33:553, 33:767
`\textlegacyparagraph` 33:553, 33:769
`\textlegacyperiodcentered` 33:553, 33:770
`\textlegacysection` 33:553, 33:771
`\textless`
 21:300, 21:596, 21:761, 21:1107, 1305
`\textlira` 21:953, 21:1214, 33:285, 33:286,
 33:733, 33:1048, 33:1049, 33:1261
`\textlnot` 21:979,
 21:1129, 33:98, 33:622, 33:949, 33:1360
`\textlquill`
 .. 21:967, 21:1210, 33:244, 33:245,
 33:709, 33:1072, 33:1073, 33:1327
`\textmacron` 1324
`\textmarried`
 .. 21:931, 21:1256, 33:246, 33:247,
 33:710, 33:1028, 33:1029, 33:1306
`\textmd` 32:19
`\textmho` 21:913, 21:1227, 33:248, 33:249,
 33:711, 33:1012, 33:1013, 33:1297
`\textminus` ... 21:911, 21:1233, 33:307,
 33:308, 33:747, 33:924, 33:1242, 770
`\textmu` 21:988, 21:1137,
 33:305, 33:306, 33:746, 33:957, 33:1244
`\textmusicalnote`
 .. 21:932, 21:1255, 33:250, 33:251,
 33:712, 33:1030, 33:1031, 33:1307
`\textnaira`
 .. 21:950, 21:1215, 33:252, 33:253,
 33:713, 33:1042, 33:1043, 33:1317
`\textnineoldstyle`
 ... 21:909, 33:184, 33:185, 33:349,
 33:677, 33:1006, 33:1007, 33:1296
`\textnonbreakinghyphen`
 21:429, 21:589, 21:1181, 21:1186, 1343
`\textnormal` 32:15
`\textnumero`
 .. 21:962, 21:1221, 33:295, 33:296,
 33:739, 33:1064, 33:1065, 33:1251
`\textogonekcentered`
 21:508, 21:719, 21:720, 1327
`\textohm` 21:921, 21:1226, 33:303,
 33:304, 33:745, 33:816, 33:969, 33:1243
`\textonehalf` 21:996,
 21:1146, 33:99, 33:623, 33:963, 33:1368
`\textoneoldstyle` 21:901, 33:186, 33:187,
 33:341, 33:678, 33:990, 33:991, 33:1288
`\textonequarter` 21:995,
 21:1145, 33:100, 33:624, 33:962, 33:1367
`\textonesuperior`
 .. 21:992, 21:1140, 33:101, 33:319,
 33:320, 33:625, 33:756, 33:960, 33:1232
`\textopenbullet`
 .. 21:965, 21:1253, 33:254, 33:255,
 33:714, 33:1068, 33:1069, 33:1325
`\textordfeminine` 21:325, 21:977,
 21:1125, 33:102, 33:626, 33:948, 33:1359
`\textordmasculine` ... 21:326, 21:993,
 21:1141, 33:103, 33:628, 33:961, 33:1366
`\TextOrMath`
 ... 22:214, 22:217, 22:218, 22:219,

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
 06=ltdefs.dtx, 07=ltcmd, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx,
 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
 56=ltthyphen.dtx, 57=ltfinal.dtx

- 22:220, 22:221, 22:222, 22:223,
 22:224, 22:225, 22:230, [22:237](#), [1329](#)
 \textparagraph . 21:296, 21:330, 21:748,
 21:989, 21:1138, 22:221, 33:88,
 33:546, 33:610, 33:958, 33:1364, [1305](#)
 \textperiodcentered
 .. 21:297, 21:749, 21:990, 21:1139,
 33:89, 33:547, 33:611, 33:959, 33:1365
 \textpermill [1323](#)
 \textpertenmill [1323](#)
 \textpertenthousand
 .. 21:513, 21:959, 21:1204, 33:270,
 33:271, 33:272, 33:723, 33:1058,
 33:1059, 33:1096, 33:1321, [1323](#)
 \textperthousand 21:511,
 21:942, 21:1203, 33:82, 33:83,
 33:603, 33:936, 33:1095, 33:1350, [1323](#)
 \textpeso 21:952, 21:1219, 33:256, 33:257,
 33:715, 33:1046, 33:1047, 33:1319
 \textpilcrow
 ... 21:960, 33:258, 33:259, 33:350,
 33:716, 33:1060, 33:1061, 33:1322
 \textpm 21:984,
 21:1133, 33:104, 33:630, 33:953, 33:1363
 \textquestiondown ... 21:281, 21:433,
 21:435, 21:597, 21:812, 21:1148, [1327](#)
 \textquotedbl 21:600, 21:1105, [1304](#)
 \textquotedblleft 21:282,
 21:438, 21:598, 21:813, 21:1196, [1304](#)
 \textquotedblright 21:283,
 21:439, 21:599, 21:814, 21:1197, [1304](#)
 \textquoteleft 21:284,
 21:440, 21:601, 21:815, 21:1193, [1304](#)
 \textquoteright 21:285,
 21:441, 21:602, 21:816, 21:1194, [1304](#)
 \textquotesingle 21:885,
 21:1103, 33:105, 33:631, 33:921, 33:1346
 \textquotestraightbase 21:877,
 33:106, 33:335, 33:632, 33:916, 33:1341
 \textquotestraightdblbase ... 21:878,
 33:107, 33:336, 33:633, 33:917, 33:1342
 \texttrangle
 .. 21:912, 21:1250, 33:275, 33:276,
 33:727, 33:1010, 33:1011, 33:1266
 \textrbrace [1304](#)
 \textbrackdbl 21:923, 33:176,
 33:177, 33:354, 33:672, 33:926, 33:1300
 \textrecipe
 .. 21:954, 21:1223, 33:260, 33:261,
 33:717, 33:1050, 33:1051, 33:1320
 \textreferencemark
 .. 21:991, 21:1207, 33:262, 33:263,
 33:718, 33:1080, 33:1081, 33:1333
 \textregistered
 .. 21:320, 21:321, 21:981, 21:1130,
 33:108, 33:634, 33:950, 33:1361, [1327](#)
 \textrightarrow 21:882, 21:1231, 33:287,
 33:288, 33:734, 33:982, 33:983, 33:1256
 \textrm [32:15](#)
 \texttrquill
 .. 21:968, 21:1211, 33:264, 33:265,
 33:719, 33:1074, 33:1075, 33:1328
 \textsc [32:21](#), [1322](#)
 \textsection 21:298,
 21:331, 21:603, 21:750, 21:974,
 21:1122, 22:220, 33:90, 33:548,
 33:549, 33:612, 33:945, 33:1357, [1305](#)
 \textservicemark
 .. 21:966, 21:1224, 33:266, 33:267,
 33:720, 33:1070, 33:1071, 33:1326
 \textsevenoldstyle
 ... 21:907, 33:188, 33:189, 33:347,
 33:679, 33:1002, 33:1003, 33:1294
 \textsf [32:15](#)
 \textsixoldstyle
 ... 21:906, 33:190, 33:191, 33:346,
 33:680, 33:1000, 33:1001, 33:1293
 \textsl [32:21](#)
 \textssc [25:2924](#), [32:25](#)
 \textsterling .. 21:286, 21:339, 21:465,
 21:604, 21:827, 21:970, 21:1118,
 33:80, 33:81, 33:588, 33:596,
 33:942, 33:1091, 33:1093, 33:1354, [1304](#)
 \textstyle 19:15, 30:480, 38:63
 \textsubscript [45:419](#)
 \textsuperscript 21:323, 21:325, 21:326,
 33:627, 33:629, 33:643, [45:402](#), [1312](#)
 \textsurd 21:994, 21:1248, 33:268, 33:269,
 33:721, 33:1082, 33:1083, 33:1334
 \textsw [25:2919](#), [32:25](#)
 \TextSymbolUnavailable [21:3](#), 21:780, [1317](#)
 \textthreeoldstyle
 21:903, 33:192, 33:193,
 33:343, 33:681, 33:994, 33:995, 33:1290
 \textthreequarters 21:997,
 21:1147, 33:110, 33:639, 33:964, 33:1369
 \textthreequartersemdash
 ... 21:880, 33:109, 33:321, 33:322,
 33:339, 33:638, 33:757, 33:919, 33:1227
 \textthreesuperior
 .. 21:986, 21:1135, 33:111, 33:323,
 33:324, 33:640, 33:758, 33:955, 33:1231
 \texttildelow 21:933, 21:1177,
 33:214, 33:215, 33:693, 33:928, 33:1308
 \texttimes 21:999,
 21:1151, 33:112, 33:641, 33:965, 33:1370

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx,
 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
 56=lthyphen.dtx, 57=ltfinal.dtx

- `\texttrademark` 21:323, 21:958, 21:1225,
 33:113, 33:642, 33:940, 33:1352, *1317*
`\texttt` *32:15*
`\texttwelveudash`
 ... 21:879, 33:114, 33:325, 33:326,
 33:338, 33:644, 33:759, 33:918, 33:1226
`\texttwooldstyle` 21:902, 33:194, 33:195,
 33:342, 33:682, 33:992, 33:993, 33:1289
`\texttwosuperior`
 .. 21:985, 21:1134, 33:115, 33:327,
 33:328, 33:645, 33:760, 33:954, 33:1230
`\textulc` *25:2914*, *32:25*
`\textunderline` *1304*
`\textunderscore`
 21:308, 21:335, 21:605, 21:1111, *1322*
`\textup` *32:21*, *625*
`\textuparrow`
 .. 21:924, 21:1230, 33:289, 33:290,
 33:735, 33:1016, 33:1017, 33:1257
`\textvisiblespace`
 21:312, 21:606, 21:1252, *1313*
`\textwidth` 20:24, 20:94, 20:151,
 40:444, 45:266, 54:75, 54:142,
 54:199, 54:216, 54:843, 54:858,
 54:908, 54:918, 54:967, 54:977,
 54:2807, 54:2850, 54:2879, 57:157, *1295*
`\textwon` 21:949, 21:1216, 33:291, 33:292,
 33:736, 33:1040, 33:1041, 33:1260
`\textyen` 21:972,
 21:1120, 33:116, 33:646, 33:943, 33:1355
`\textzeroldstyle`
 21:900, 33:196, 33:197,
 33:340, 33:683, 33:988, 33:989, 33:1287
`\TH` 21:556, 21:1153, 57:656, *1300*
`\th` 21:607, 21:1159, 57:656, *1300*
`\thanks` *958*
`\thanks` *44:10*, *44:26*, *1306*
`\the` *421*
`\the...` *1354*
`\the#1` *524*
`thebibliography` (env.) *998*
`\theenum` *875*
`\theequation` 38:352, 38:364, 38:457, 38:518
`\thefootnote` *45:396*, 45:521, 45:526, 45:546
`\thempfn` 40:454,
 45:453, 45:458, 45:537, 45:542, *45:545*
`\thempfootnote` 40:454, *45:398*
`\thepage` 20:209, 34:6, 35:18,
 35:35, 35:52, 35:106, 35:174, 36:211,
 36:222, 36:227, 36:234, 44:164,
 44:171, 44:177, 46:15, 46:32, 47:43,
 49:14, 54:242, 54:273, 54:2327, *1145*
`\Theta` 30:310
`\theta` 30:286
`\thetotalpages` 53:350, 53:442, *1145*
`\thicklines` *42:124*
`\thickmuskip` 30:657, 38:228, 38:230, 38:243
`\thickspace` *38:214*
`\thinlines` *42:124*, 42:815, 42:832
`\thinmuskip`
 30:655, 38:220, 38:222, 38:238, 38:244
`\thinspace` 18:536, 18:542,
 18:543, 38:189, *38:214*, 38:251, *1338*
`\thispagestyle` *49:6*
`\tilde` 18:472, 18:483, 30:530
`\time` 01:163, 01:167, 02:383
`\times` 30:407
`\title` *958*
`\title` 44:6, *44:7*, 44:21, 44:23, 44:31
`title` *36:237*, *810*
`tl` commands:
`\c_empty_tl` ... 08:2211, 11:1205, 08:65
`\c_novalue_tl` 07:528, 07:932, 07:999,
 07:1060, 07:1536, 07:1808, 07:1945,
 07:2050, 07:2211, 07:2218, 07:2289,
 07:2341, 07:2422, 07:2449, 07:2531, *141*
`\c_space_tl` . 09:382, 11:550, 11:554,
 11:597, 11:795, 11:798, 16:42,
 16:73, 16:82, 52:553, 52:556, 07:26,
 07:123, 07:139, 07:142, 07:156,
 07:167, 07:168, 07:184, 07:192,
 07:202, 07:205, 07:207, 07:209,
 07:211, 07:219, 07:225, 07:293,
 07:294, 57:604, 57:611, 57:616,
 57:621, 07:1186, 07:1201, 07:1233,
 07:1234, 07:1411, 07:1418, 07:1544
`\tl_clear:N` 09:213,
 09:288, 11:284, 11:323, 11:713,
 07:308, 07:353, 07:382, 07:449,
 07:450, 07:670, 07:788, 07:800,
 07:801, 07:802, 07:804, 07:984,
 07:1052, 07:1246, 07:1479, 07:1480,
 07:1552, 07:1573, 07:1601, 07:1856,
 07:1857, 07:1873, 07:2085, 07:2481
`\tl_const:Nn`
 09:11, 09:12, 11:9, 11:10,
 11:11, 11:12, 11:13, 11:14, 11:15,
 48:61, 53:311, 53:316, 07:1502,
 07:2801, 08:35, 08:36, 08:110,
 08:113, 08:920, 08:921, 08:923,
 08:924, 08:925, 08:926, 08:929,
 08:930, 08:932, 08:953, 08:954, 08:1298
`\tl_count:N` ... 52:546, 07:400, 07:2505
`\tl_count:n`
 .. 07:605, 07:610, 07:887, 07:1005,
 07:1059, 07:1286, 07:1436, 07:1444
`\tl_gclear:N` 48:149, 48:150, *287*
`\tl_gclear_new:N` . 36:25, 36:145, 08:270

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
 06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx,
 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
 56=lthyphen.dtx, 57=ltfinal.dtx

- \tl_gput_left:Nn 51:238
- \tl_gput_right:Nn 16:38,
16:69, 16:78, 28:336, 48:174, 48:185,
07:1161, 07:1472, 07:1474, 08:455, 271
- \tl_gremove_once:Nn 08:37, 08:37
- \tl_gset:Nn
..... 09:141, 16:15, 16:48, 17:34,
17:38, 28:285, 28:333, 36:26, 36:138,
36:146, 48:129, 48:140, 48:160,
48:224, 48:225, 48:226, 52:51,
55:166, 55:167, 55:187, 55:191,
55:237, 55:245, 55:291, 08:182,
08:194, 08:301, 08:319, 08:417,
08:429, 08:445, 08:484, 08:750,
08:804, 08:838, 08:842, 08:865, 08:869
- \tl_gset_eq:NN
48:57, 48:58, 48:59, 48:127, 48:128,
48:133, 48:135, 48:139, 48:202,
48:204, 48:206, 48:213, 48:215,
48:217, 48:455, 48:457, 48:459,
48:461, 48:467, 48:471, 48:474,
55:268, 55:269, 08:63, 08:452, 08:463
- \tl_head:N 07:2733
- \tl_head:n 07:11, 07:811
- \tl_head:w 11:79, 11:92
- \tl_if_blank:nTF .. 11:331, 11:335,
11:442, 11:445, 20:521, 55:50,
57:584, 07:601, 07:1613, 07:1731,
07:1839, 07:2303, 07:2344, 07:2555,
07:3256, 07:3257, 07:3258, 08:435
- \tl_if_blank_p:n 07:119
- \tl_if_empty:N 294
- \tl_if_empty:NNTF 08:1989,
08:1997, 08:2194, 08:2196, 11:245,
11:258, 11:311, 48:131, 48:162,
48:250, 07:319, 07:320, 07:468,
07:721, 07:1486, 08:235, 08:255,
08:366, 08:368, 08:638, 08:1039, 08:1429
- \tl_if_empty:nTF
..... 08:2520, 09:137, 09:322,
09:341, 11:1155, 11:1160, 11:1203,
11:1213, 49:26, 49:30, 49:45,
49:55, 52:40, 52:46, 52:58, 52:107,
52:114, 52:116, 52:118, 52:416,
07:1635, 07:2300, 07:2671, 07:2697,
07:2732, 07:2936, 07:3026, 08:338,
08:353, 08:356, 08:825, 08:857, 08:911
- \tl_if_empty_p:N
08:2470, 08:2471, 53:68, 53:114, 53:378
- \tl_if_empty_p:n 08:888
- \tl_if_eq:NNTF 48:279,
48:287, 48:294, 48:363, 48:464, 07:342
- \tl_if_eq:NnTF 28:296, 36:151
- \tl_if_eq:nnTF 07:733, 07:1018, 07:2676
- \tl_if_exist:N 289
- \tl_if_exist:NNTF
..... 08:2239, 08:2259, 08:2345,
08:2493, 08:2534, 36:106, 36:111,
36:130, 36:172, 36:187, 48:361,
07:1487, 08:135, 08:233, 08:253, 08:1276
- \tl_if_exist_p:N 08:276
- \tl_if_head_eq_charcode:nNTF ...
..... 20:523, 20:528
- \tl_if_head_is_group:nTF
... 09:349, 07:2553, 07:2597, 07:2629
- \tl_if_head_is_N_type:nTF 09:346,
07:2565, 07:2577, 07:2594, 07:2626
- \tl_if_in:NnTF 07:1870
- \tl_if_in:nnTF
..... 11:280, 11:304, 11:326,
11:745, 07:617, 07:1836, 07:1862, 175
- \tl_if_novalue:nTF
... 07:373, 07:392, 07:439, 07:988,
07:2373, 07:3248, 07:3249, 07:3250,
07:3251, 07:3252, 07:3253, 08:332
- \tl_if_single:nTF
..... 11:493, 07:2470, 07:3231
- \tl_if_single_token:nTF
... 09:332, 07:681, 07:2668, 07:3233, 196
- \tl_log:n 08:1822, 08:37, 08:39
- \tl_map_function:nN 07:100, 07:393,
07:602, 07:603, 07:1003, 07:1053, 1359
- \tl_map_inline:Nn 07:731
- \tl_map_inline:nn 07:1515, 07:1760,
07:1762, 07:1764, 07:1766, 07:1942
- \tl_map_tokens:nn 07:3272, 1359
- \tl_new:N ... 09:6, 09:8, 09:9, 09:10,
11:19, 11:20, 11:23, 11:24, 11:25,
11:26, 11:27, 11:37, 11:1218, 16:14,
36:137, 48:54, 48:55, 48:56, 48:64,
48:65, 48:117, 48:197, 48:198, 52:8,
52:9, 52:10, 52:11, 52:59, 53:53,
53:205, 07:12, 07:13, 07:14, 07:15,
07:19, 07:23, 07:30, 07:31, 07:32,
07:35, 07:40, 07:41, 07:43, 07:44,
07:53, 07:54, 55:163, 55:164, 55:168,
55:169, 55:170, 55:171, 07:1851,
07:1852, 07:2068, 07:2467, 07:3267,
08:25, 08:26, 08:27, 08:29, 08:32,
08:78, 08:96, 08:124, 08:138, 08:141,
08:165, 08:166, 08:299, 08:317,
08:1295, 08:1551, 08:1552, 08:1553
- \tl_put_left:Nn 07:849, 07:858, 07:1571
- \tl_put_right:Nn 09:185,
09:199, 09:260, 09:274, 11:299,
11:306, 11:777, 11:785, 11:812,
11:823, 11:840, 11:865, 11:872,
07:368, 57:627, 07:394, 07:538,

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

07:571, 07:594, 07:608, 07:625,
 07:630, 07:763, 07:878, 07:891,
 07:918, 07:926, 07:940, 07:942,
 07:949, 07:977, 07:993, 07:999,
 07:1068, 07:1093, 07:1261, 07:1269,
 07:1285, 07:1292, 07:1547, 07:1550,
 07:1634, 07:1637, 07:1662, 07:1676,
 07:1685, 07:1706, 07:1711, 07:1717,
 07:1860, 07:1861, 07:1868, 07:1878,
 07:1946, 07:1980, 07:2225, 07:2237,
 07:2253, 07:2264, 07:2483, 07:2528
 \tl_replace_all:Nnn .. 11:279, 07:2496
 \tl_rescan:nn .. 09:20, 09:20, 09:479, 09:555, 09:583
 \tl_reverse:n .. 07:811
 \tl_set:Nn .. 09:183, 09:196,
 09:197, 09:200, 09:202, 09:223,
 09:258, 09:271, 09:272, 09:275,
 09:277, 09:298, 09:390, 09:395,
 09:476, 09:521, 09:552, 09:580,
 11:255, 11:278, 11:322, 11:333,
 11:337, 11:423, 11:549, 11:553,
 11:596, 11:627, 11:734, 11:752,
 11:883, 11:895, 11:917, 48:102,
 52:81, 52:82, 52:83, 52:84, 53:47,
 53:136, 53:210, 53:236, 53:261,
 53:288, 07:24, 07:91, 07:122, 55:39,
 55:173, 55:174, 55:175, 55:176,
 55:220, 55:221, 07:309, 07:310,
 07:311, 07:312, 07:340, 07:391,
 07:897, 07:1105, 07:1123, 07:1325,
 07:1543, 07:1578, 07:1599, 07:1650,
 07:1667, 07:1738, 07:1833, 07:1939,
 07:1966, 07:1973, 07:1986, 07:1992,
 07:1998, 07:2004, 07:2010, 07:2016,
 07:2022, 07:2028, 07:2059, 07:2081,
 07:2104, 07:2463, 07:2464, 07:2490,
 07:2514, 07:2545, 07:2557, 07:2584,
 07:2650, 07:2652, 07:2771, 07:2772,
 08:37, 08:40, 08:558, 08:1149,
 08:1156, 08:1160, 08:1560, 08:1580,
 08:1581, 08:1586, 08:1587, 08:1602,
 08:1609, 08:1611, 08:1642, 08:1663,
 08:1664, 08:1670, 08:1672, 08:1691,
 08:1698, 08:1700, 08:1747, 08:1754
 \tl_set_eq:NN .. 09:214, 09:289, 11:1205,
 48:154, 48:155, 55:40, 55:217,
 55:222, 07:330, 07:336, 07:344,
 07:387, 07:1572, 07:1872, 07:1945,
 08:1590, 08:1614, 08:1675, 08:1703
 \tl_show:n .. 08:1827,
 08:1930, 08:2031, 07:1367, 07:1380,
 07:1389, 07:1425, 07:1448, 08:37, 08:38
 \tl_tail:N .. 07:2138
 \tl_to_str:N 07:2209, 07:2216, 07:2809
 \tl_to_str:n .. 08:1978, 08:1991, 08:2062,
 08:2134, 08:2148, 09:130, 09:135,
 09:474, 09:493, 09:522, 09:533,
 09:569, 11:302, 11:424, 11:628,
 11:883, 11:950, 11:951, 11:952,
 11:961, 11:962, 36:10, 36:19, 36:51,
 36:64, 36:92, 36:93, 36:100, 36:101,
 36:145, 36:146, 36:187, 48:245,
 48:260, 51:124, 51:153, 51:155,
 52:580, 07:73, 07:84, 07:251, 07:255,
 07:463, 07:471, 07:484, 07:495,
 07:499, 07:535, 07:566, 07:647,
 07:676, 07:684, 07:699, 07:712,
 07:724, 07:743, 07:783, 07:784,
 07:1259, 07:1333, 07:1339, 07:1547,
 07:1551, 07:2210, 07:2217, 07:2417,
 07:2444, 07:2524, 07:2525, 07:2685,
 07:2689, 08:384, 08:535, 08:608, 1096
 \tl_trim_spaces:n . 09:324, 11:301,
 11:322, 11:334, 11:424, 11:448,
 11:628, 11:883, 50:514, 51:114,
 51:145, 52:41, 52:42, 07:247, 07:420,
 07:538, 07:572, 07:1561, 07:1563,
 07:2416, 07:2443, 07:2545, 07:3111,
 07:3115, 07:3119, 07:3123, 08:413, 324
 \tl_trim_spaces_apply:nN . 07:681,
 07:2548, 07:2572, 07:2662, 08:334
 \tl_use:N .. 08:1813, 24:415, 28:291, 28:295,
 28:343, 28:347, 36:94, 36:107,
 07:1404, 07:1492, 08:1057, 08:1120
 \l_tmpa_tl .. 354
 tl internal commands:
 \c__mark_empty_tl .. 48:53, 48:214, 48:216, 48:218, 48:363
 \g__mark_first_marks_tl .. 48:149, 48:154, 48:185, 48:197, 1021
 \g__mark_last_marks_tl .. 48:150, 48:155, 48:174, 48:197, 1021
 \g__mark_new_top_tl .. 48:62, 48:127, 48:128, 48:134, 48:136
 \l__mark_saved_parameters_tl .. 48:102, 48:115, 48:117
 \g__mark_tmp_tl .. 48:62,
 48:129, 48:131, 48:139, 48:160,
 48:162, 48:172, 48:175, 48:239,
 48:245, 48:250, 48:252, 48:253, 1023
 \tmspace .. 38:214, 850
 \to .. 30:451, 30:453
 \today .. 01:168,
 01:172, 01:180, 01:183, 44:33, 1275

File Key: 01=ltldirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx,
 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx,
 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx,
 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx,
 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx,
 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx,
 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx,
 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx,
 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx,
 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx,
 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx,
 56=lthyphen.dtx, 57=ltfinal.dtx

- token commands:
- `\c_math_subscript_token` 07:1824
 - `\c_math_toggle_token` 07:1822
 - `\c_space_token` 07:1660, 07:2761
 - `\token_case_charcode:NnTF`
 07:1656, 07:1692
 - `\token_if_active:NTF` 07:2239, 07:2255
 - `\token_if_cs:NTF` 07:2472, 185
 - `\token_if_eq_catcode:NNTF`
 09:334, 07:1890
 - `\token_if_eq_charcode:NNTF`
 07:415, 07:2126, 07:2158, 07:2169
 - `\token_if_eq_meaning:NNTF` 09:204,
 09:279, 09:480, 52:454, 52:466,
 07:690, 07:693, 07:701, 07:1326,
 07:1817, 07:2775, 07:3235, 07:3237
 - `\token_if_long_macro_p:N` 07:1420
 - `\token_if_macro:NTF`
 09:101, 08:1205, 08:1231, 318
 - `\token_if_math_toggle_p:N`
 07:2618, 07:2640
 - `\token_if_protected_long_macro_p:N` 07:1421, 07:1430
 - `\token_if_protected_macro:NTF`
 07:1189, 07:1214, 07:1381
 - `\token_if_protected_macro_p:N` 07:1429
 - `\token_to_meaning:N` 09:97, 09:477,
 09:489, 09:553, 09:581, 16:41, 16:72
 - `\token_to_str:N`
 09:96, 09:97, 09:119, 09:539,
 09:546, 09:574, 09:577, 11:279,
 11:280, 11:297, 11:304, 11:307,
 11:314, 11:1086, 36:57, 52:241,
 07:73, 07:81, 07:84, 07:420, 07:827,
 07:1100, 07:1254, 07:1403, 07:1427,
 07:1450, 07:1903, 07:2049, 07:2244,
 07:2256, 07:2416, 07:2443, 07:2672,
 07:2685, 07:2689, 07:2698, 07:3078,
 07:3079, 07:3092, 07:3093, 07:3197,
 07:3198, 07:3211, 07:3212, 08:381, 165
 - `\toks` 16:40, 16:71,
 16:80, 04:36, 28:684, 28:685, 28:695,
 28:704, 02:31, 02:63, 57:704, 02:95, 421
 - `\toksdef` 04:230, 02:46, 02:63, 02:95
 - `\tokszero` 04:230
 - `\tolerance` 24:744,
 24:790, 24:805, 49:131, 49:139, 02:317
 - `\top` 30:331
 - `\topfigrule` 54:1001, 54:2949
 - `\topfraction` 45:273, 54:2915
 - `\topmargin` 54:67, 54:834, 54:902, 54:961
 - `\TopMark` 48:388, 48:510, 1005
 - `\topmark` 54:2836, 54:2845, 1003
 - `\topsep` 38:524, 39:1, 39:59, 862
 - `\topskip` 20:59,
 20:128, 20:183, 54:126, 02:418, 1314
 - `\totalheight` 40:33, 40:34, 40:35
 - `totalpages` 1129
- trace commands:
- `trace_stack_levels` 57:96
 - `\tracefloats` 54:2429
 - `\tracefloatsoff` 54:2429
 - `\tracefloatvals` 54:2429, 1345
 - `\traceoff` 407
 - `\traceon` 407
 - `\tracingall` 02:557, 407
 - `\tracingassigns`
 02:575, 02:609, 02:626, 02:667
 - `\tracingcommands` 02:573, 02:591, 02:607,
 02:616, 02:629, 02:653, 02:670, 02:347
 - `\tracingfonts` 26:17, 26:55,
 26:59, 26:87, 26:119, 26:154, 26:195,
 26:225, 26:239, 26:255, 26:261,
 26:274, 26:281, 26:288, 26:293,
 26:302, 26:315, 26:323, 26:326, 1281
 - `\tracinggroups`
 02:565, 02:600, 02:638, 02:678
 - `\tracingifs` 02:566, 02:601, 02:637, 02:677
 - `\tracinglostchars` 02:563, 02:586,
 02:598, 02:617, 02:641, 02:661,
 02:681, 24:754, 02:346, 02:558, 1345
 - `\tracingmacros` 02:572, 02:590, 02:606,
 02:618, 02:640, 02:660, 02:680, 02:341
 - `\tracingnesting`
 02:568, 02:603, 02:635, 02:675
 - `\tracingnone` 02:622, 1354
 - `\tracingoff` 26:119, 26:323
 - `\tracingon` 26:120, 26:324, 1282
 - `\tracingonline` 02:628, 02:652, 02:669,
 02:699, 24:753, 54:2415, 02:340, 02:552
 - `\tracingoutput` 02:632,
 02:656, 02:673, 02:696, 02:345, 02:553
 - `\tracingpages` 02:562, 02:585, 02:597,
 02:617, 02:642, 02:662, 02:682, 02:344
 - `\tracingparagraphs`
 02:564, 02:587, 02:599,
 02:618, 02:639, 02:659, 02:679, 02:343
 - `\tracingrestores` 02:574, 02:592, 02:608,
 02:618, 02:627, 02:658, 02:668, 02:348
 - `\tracingscantokens` 02:567,
 02:582, 02:602, 02:636, 02:650, 02:676
 - `\tracingstacklevels`
 02:570, 02:634, 02:349, 02:558, 24
 - `\tracingstats`
 02:584, 02:596, 02:616, 02:643,
 02:663, 02:683, 57:2, 02:342, 02:561
 - `\triangle` 30:333
 - `\triangleleft` 30:369, 30:493

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- `\triangleright` 30:370, 30:493
`\TrimSpaces` 07:3268
`trivlist (env.)` 39:89
`\trivlist` 37:444, 37:494, 37:496,
37:527, 37:549, 38:508, 39:89,
41:78, 43:52, 43:54, 43:59, 43:61, 869
TS1 commands:
`\TS1:?` 788
`\TS1:(family)` 788
`\tt` 1286
`\ttdefault` 29:13, 29:217, 29:401,
29:418, 29:442, 29:461, 29:489, 30:62
`\ttfamily` 29:11,
29:12, 29:411, 29:459, 29:460,
29:487, 29:488, 32:17, 37:571, 226
`ttfamily` 29:421
`\ttsubstdefault` 30:20, 30:32, 33:32, 33:53
`\twocolumn` 54:197, 1321
`\twocolumn[...]` 1315
`\twocolumn[]` 455
`\typein` 82, 82
`\typein` 06:31, 33:1584, 33:1589
`\typeout` 82
`\typeout` .. 01:100, 03:21, 03:38, 10:59,
10:62, 10:64, 10:66, 10:68, 10:71,
10:74, 14:74, 20:201, 20:673, 20:674,
20:680, 20:714, 20:760, 20:774,
20:795, 01:156, 24:463, 01:181,
01:183, 01:195, 25:2866, 01:210,
01:217, 29:139, 01:228, 29:564,
29:745, 29:755, 29:765, 30:9, 30:136,
01:241, 06:3, 01:254, 33:1180,
33:1217, 33:1224, 33:1377, 33:1380,
33:1383, 06:36, 33:1581, 33:1582,
33:1583, 33:1586, 33:1587, 33:1588,
33:1593, 33:1597, 06:43, 01:267,
01:305, 46:8, 46:25, 48:346, 48:365,
06:614, 06:615, 50:384, 50:400,
50:412, 06:649, 06:650, 06:655,
06:709, 06:710, 06:725, 06:727,
50:1589, 50:1796, 50:1799, 53:104,
53:115, 53:148, 54:2418, 54:2430,
55:179, 57:276, 57:658, 57:665,
57:677, 57:678, 57:686, 01:57, 1127
`\typeoutdetails` 33:1180, 33:1181,
33:1184, 33:1190, 33:1192, 33:1196,
33:1197, 33:1203, 33:1221, 33:1222,
33:1376, 33:1591, 33:1593, 33:1597
- U**
- `\u` 21:257, 21:406, 21:493, 21:610,
21:617, 21:637, 21:644, 21:776,
21:1264, 21:1339, 21:1340, 21:1355,
21:1356, 21:1365, 21:1366, 21:1379,
21:1380, 21:1381, 21:1405, 21:1406,
21:1431, 21:1432, 33:127, 33:158
`\uccode` 57:240, 57:248, 57:255,
57:257, 57:260, 57:262, 57:535,
57:543, 57:550, 57:552, 57:555, 57:557
`\Ucharcat` 29:625
`\uchyph` 02:370
`\ulcdefault` 25:2914, 25:2936
`\ulcshape`
25:2911, 25:2914, 25:2932, 25:2935,
25:3218, 25:3219, 29:550, 32:29, 623
`\Umathcode` 04:30, 18:552, 30:15,
30:55, 30:62, 30:77, 33:123, 33:331,
37:577, 57:160, 57:351, 57:529, 02:127
`\unboldmath` 29:616
`\UndeclareTextCommand`
21:209, 33:79, 33:81, 33:83, 33:272,
33:549, 33:1091, 33:1092, 33:1093,
33:1094, 33:1095, 33:1096, 1296
`\undefined` 22:124,
22:125, 22:181, 22:182, 22:183,
22:184, 22:185, 22:186, 01:12, 01:14,
30:127, 30:128, 35:40, 35:41, 35:57,
35:58, 35:59, 35:60, 01:20, 57:168,
57:180, 57:181, 57:201, 01:41, 1316
`\undefinedpagestyle` 49:4, 49:8
`\underbar` 06:893, 06:914, 02:522
`\underbrace` 30:551
`\underline` 879
`\underline` 40:551, 40:552, 02:522
`\underscore` 1322
`\unexpanded` 32:47, 37:279, 37:315, 06:632,
06:643, 50:824, 50:1624, 50:1626, 322
`\unhbox` 1283
`\unhcopy` ... 41:345, 42:741, 42:797, 02:524
`\unicodedataline`
04:143, 04:146, 04:160, 04:161, 04:162
`\UnicodeEncodingName`
..... 21:1003, 21:1009, 21:1060,
21:1066, 21:1070, 21:1081, 21:1085,
21:1101, 21:1102, 33:340, 33:341,
33:342, 33:343, 33:344, 33:345,
33:346, 33:347, 33:348, 33:349,
33:350, 33:351, 33:352, 33:353,
33:354, 33:355, 33:356, 33:357, 509
`\UnicodeFontFile` 21:1058
`\UnicodeFontName` 21:1059
`\UnicodeFontTeXLigatures` 21:1015, 21:1055
`\unicoderead` 04:143,
04:157, 04:158, 04:159, 04:160, 04:165
`uninstall` 04:999
`\unitlength` 40:74, 40:85,
40:94, 40:104, 42:5, 42:29, 42:30,
42:32, 42:34, 42:42, 42:43, 42:44,

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=ltthyphen.dtx, 57=ltfinal.dtx

42:45, 42:60, 42:63, 42:73, 42:74,	52:465, 57:595, 07:823, 07:1489,
42:84, 42:85, 42:93, 42:94, 42:107,	07:1819, 08:1630, 08:1763, 08:1764, 347
42:108, 42:119, 42:164, 42:176,	<code>\use:n</code> 08:2071, 08:2105,
42:241, 42:256, 42:316, 42:318,	08:2358, 08:2364, 08:2373, 09:122,
42:332, 42:340, 42:342, 42:357,	09:189, 09:192, 09:220, 09:229,
42:375, 42:377, 42:392, 42:397,	09:264, 09:267, 09:295, 09:301,
42:399, 42:414, 42:417, 42:422,	09:367, 09:400, 09:423, 09:442,
42:479, 42:480, 42:507, 42:508,	09:471, 09:501, 09:523, 09:530,
42:536, 42:537, 42:611, 42:627,	09:566, 11:294, 11:338, 11:372,
42:647, 42:653, 42:689, 42:690,	20:523, 48:112, 51:183, 07:189,
42:692, 42:693, 42:696, 42:697,	07:691, 07:694, 07:885, 07:1129,
42:699, 42:700, 42:711, 42:712,	07:1256, 07:1348, 07:1457, 07:1628,
42:714, 42:715, 42:717, 42:718,	07:1644, 07:1784, 07:2605, 07:2647,
42:720, 42:721, 42:749, 42:750,	07:3232, 07:3234, 07:3239, 08:238,
42:752, 42:753, 42:756, 42:757,	08:388, 08:511, 08:1153, 08:1628, 466
42:759, 42:760, 42:771, 42:773,	<code>\use:nn</code> 08:2121,
42:775, 42:777, 53:328, 57:149, 1143	08:2696, 09:234, 09:306, 08:1734
<code>\unkern</code> 24:773	<code>\use:nnn</code>
<code>\unless</code> 04:151, 04:159, 04:161	07:3078, 07:3092, 07:3197, 07:3211, 193
<code>\unlhd</code> 29:733	<code>\use_i:n</code> 08:2223, 08:2568,
<code>\unpenalty</code> 24:776,	07:980, 07:1861, 07:2700, 07:2753, 354
24:780, 32:116, 37:543, 37:565, 418	<code>\use_i:nnn</code> 08:2070, 07:2678, 268
<code>\unrhd</code> 29:735	<code>\use_i:nnnn</code> 09:365, 326
<code>\unsetattribute</code> 43	<code>\use_i_delimit_by_q_recursion_</code>
<code>\unsetattribute</code> 04:82, 04:240	stop:nw 09:430, 09:449
<code>\unskip</code> 1335	<code>\use_i_delimit_by_q_stop:nw</code> . 07:429
<code>\UnusedTemplateKeys</code>	<code>\use_ii:n</code> 08:2574,
. 11:1205, 11:1208, 11:1218, 354	09:74, 09:156, 55:19, 55:29, 07:1570,
<code>\unvcopy</code> 38:193, 1011	07:1782, 07:2043, 07:2280, 07:2293,
<code>\Uparrow</code> 30:595	07:2332, 07:2350, 07:2375, 07:2408,
<code>\uparrow</code> 30:589	07:2435, 07:2699, 07:3243, 08:638, 268
<code>\upbracefill</code> 30:554, 30:571	<code>\use_ii:nnn</code>
<code>\updefault</code> 25:3208, 29:22, 30:106, 07:1844, 07:2349, 07:2682, 07:3236
30:113, 30:114, 30:122, 30:124, 1337	<code>\use_ii:nnnn</code> 09:370
<code>\Updownarrow</code> 30:599	<code>\use_ii_i:n</code> 09:233, 09:305
<code>\updownarrow</code> 30:593	<code>\use_ii_iii:nnn</code>
<code>\uplus</code> 30:389 52:216, 52:230, 52:242, 52:242
<code>\uppercase</code> 1296	<code>\use_iii:n</code> 268
<code>\upshape</code> 21:462, 21:754,	<code>\use_iii:nnn</code> 52:47, 07:2309,
21:824, 21:917, 25:3206, 25:3207,	07:2677, 07:2680, 07:2752, 07:3238
29:20, 29:21, 29:550, 29:561,	<code>\use_iii:nnnn</code> 07:2308
29:594, 29:652, 32:24, 33:593, 623	<code>\use_iv:nnnn</code> 07:415, 07:421
<code>\Upsilon</code> 30:315	<code>\use_none:n</code>
<code>\upsilon</code> 30:297 08:2225, 08:2314, 09:372, 09:430,
use commands:	09:449, 10:7, 10:8, 48:326, 52:131,
<code>\use:N</code>	52:134, 53:7, 07:98, 55:3, 55:4,
08:1804, 08:2306, 08:2376, 08:2392,	55:18, 55:28, 55:182, 55:184, 07:695,
10:148, 10:155, 11:273, 11:496,	07:1781, 07:1785, 07:1839, 07:1840,
11:637, 11:763, 11:912, 28:300,	07:1843, 07:1846, 07:1860, 07:1887,
28:350, 05:181, 36:73, 48:113,	07:2024, 07:2344, 07:2345, 07:2348,
48:130, 48:143, 48:161, 48:181,	07:2351, 07:2555, 08:7, 08:639,
48:190, 48:247, 48:366, 48:367,	08:1349, 08:1420, 08:1467, 08:1732, 289
48:368, 52:442, 52:446, 52:448,	<code>\use_none:nn</code>
 08:2062, 08:2103, 28:299,

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- 28:349, 07:1146, 07:2018, 07:2303,
07:2304, 07:2307, 07:2310, 07:2584
- `\use_none:nnn` . . . 08:2068, 07:896,
07:2012, 07:2301, 07:2743, 07:3130,
07:3136, 07:3147, 07:3153, 08:745, 175
- `\use_none:nnnn` 08:2945, 07:2006, 07:2558
- `\use_none:nnnnn`
. 08:1841, 08:1959, 07:2000
- `\use_none:nnnnnn` 07:1994
- `\use_none:nnnnnnn` 07:1988
- `\use_none:nnnnnnnn`
. 08:1841, 08:1959, 08:1349
- `\use_none_delimit_by_q_recursion_`
`stop:w` 07:343
- `\use_none_delimit_by_q_stop:w`
. 07:2519, 07:2538, 07:2542
- `\usebox` 40:195
- `\usecounter` 39:256, 39:269
- `\UseExpandableTaggingSocket` 1243
- `\UseExpandableTaggingSocket` . 55:7, 1250
- `\usefont` 21:1596, 24:81, 24:345, 24:749,
29:710, 33:7, 33:577, 37:581, 1315
- `\UseHook` 08:2801, 08:2919,
09:307, 09:574, 09:577, 20:316,
20:322, 20:327, 20:338, 20:377,
20:381, 20:384, 26:146, 29:264,
29:275, 29:287, 29:297, 29:318,
29:325, 29:338, 29:345, 29:404,
29:409, 29:414, 29:419, 29:664,
29:681, 37:248, 37:260, 37:263,
37:351, 37:355, 37:368, 37:371,
50:1003, 50:1007, 50:1031, 50:1035,
52:170, 52:171, 52:174, 52:175,
53:118, 53:170, 53:382, 54:484,
54:498, 54:796, 54:804, 54:867, 322
- `\UseHookWithArguments` 08:2801,
09:236, 09:539, 09:546, 35:104, 218
- `\UseInstance` . 11:485, 11:1079, 11:1163, 358
- `\UseLegacyTextSymbols` . . . 33:540, 33:763
- `\UseName` 78
- `\UseName` 05:177, 05:195
- `\UseOneTimeHook` 08:2801, 08:2920, 20:15,
20:57, 20:70, 20:317, 20:321, 20:326,
20:339, 20:378, 20:380, 20:383,
37:14, 37:18, 37:29, 37:30, 37:32,
37:80, 37:84, 37:94, 37:95, 37:97,
50:1004, 50:1008, 50:1030, 50:1034, 202
- `\UseOneTimeHookWithArguments` 08:2801, 202
- `\usepackage` . . 50:676, 50:738, 50:1604, 303
- `\UseRawInputEncoding` 57:381, 57:437, 57:484
- `\UseSocket` 10:182, 10:209, 35:124,
35:132, 35:137, 54:491, 54:586, 342
- `\UseTaggingSocket` 1243
- `\UseTaggingSocket` 17:40, 35:133,
54:487, 54:589, 54:839, 54:854,
55:7, 55:234, 55:270, 55:285, 1244
- `\UseTemplate` 11:92, 11:1163, 363
- `\UseTextAccent`
. 21:167, 21:168, 21:206, 33:74,
33:75, 33:77, 33:120, 33:122, 33:900,
33:1085, 33:1086, 33:1088, 33:1089, 486
- `\UseTextSymbol` 21:168, 21:204,
33:73, 33:316, 33:899, 33:968, 1322
- `\usetikzlibrary` 303
- `\ushape` 1323
- ## V
- `\v` 21:258, 21:407,
21:492, 21:613, 21:614, 21:615,
21:619, 21:621, 21:624, 21:626,
21:628, 21:634, 21:640, 21:641,
21:642, 21:646, 21:648, 21:651,
21:653, 21:655, 21:661, 21:777,
21:1269, 21:1349, 21:1350, 21:1351,
21:1352, 21:1361, 21:1362, 21:1395,
21:1396, 21:1401, 21:1402, 21:1413,
21:1414, 21:1421, 21:1422, 21:1425,
21:1426, 21:1448, 21:1449, 21:1450,
21:1451, 21:1452, 21:1453, 21:1454,
21:1455, 21:1456, 21:1457, 21:1458,
21:1467, 21:1468, 21:1469, 21:1470,
21:1473, 21:1474, 33:129, 33:159
- `\vadjust` 18:38, 18:60, 18:72,
18:101, 18:108, 18:388, 18:404,
18:422, 18:438, 45:201, 45:223, 411
- `\valign` 33:65, 33:891
- `\value` 521
- `\value` 22:14,
22:16, 22:49, 22:86, 36:235, 47:9, 1283
- `\varbigtriangledown` 30:373, 30:376
- `\varbigtriangleup` 30:374, 30:375
- `\varepsilon` 19:15, 30:302
- `\varphi` 30:307
- `\varpi` 30:304
- `\varrho` 30:305
- `\varsigma` 30:306
- `\vartheta` 30:303
- `\vbadness` 53:212, 53:214,
53:263, 53:265, 54:2833, 02:319, 1141
- `\vbox` 1124
- vbox commands:
- `\vbox_set:Nn` 48:75
- `\vbox_set_to_ht:Nnn` . . 53:215, 53:266
- `\vbox_to_zero:n` 53:326
- `\vbox_unpack:N` 48:83, 48:96,
48:97, 48:407, 48:439, 53:226, 53:276
- `\vdash` 30:415

File Key: 01=ltdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=ltterror.dtx, 15=ltpara.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=lt pictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

- `\vdots` 30:521
`\vec` 30:535
`\vector` 14:269, 42:233, 42:816, 42:833
`\vee` 30:378, 30:380
`\verb` ... 37:575, 37:591, 37:604, 37:618,
37:630, 37:636, 37:642, 37:655,
37:667, 37:672, 37:680, 37:686,
37:696, 37:701, 37:714, 37:716, 1284
`\verb*` 1335
`verbatim (env.)` 37:569
`\verbatim` 37:569
`verbatim* (env.)` 37:584
`\verbvisiblespace`
... 37:575, 37:577, 37:591, 37:596,
37:606, 37:610, 37:620, 37:624, 1335
`\Vert` 30:582, 30:584
`\vert` 30:587
`\vfil` 33:66, 33:69, 33:892, 33:895,
42:565, 42:577, 53:393, 53:405,
54:173, 54:192, 54:410, 54:457,
54:837, 54:905, 54:964, 02:511, 1185
`\vfill` 54:634, 54:636,
54:645, 54:652, 54:658, 54:660, 1170
`\vfilneg` 02:511
`\vfuzz` 49:134, 49:141, 53:211,
53:213, 53:262, 53:264, 02:391, 1141
`\vglue` 02:501
`\vline` 41:366
`\voffset` 02:407
`\vphantom` 21:518, 21:534, 38:75
`\vrule` 18:528,
21:313, 21:315, 21:523, 21:539,
26:191, 30:569, 30:570, 30:572,
30:573, 40:204, 40:206, 40:282,
40:289, 40:550, 40:603, 40:610,
41:186, 41:219, 41:347, 41:366,
42:227, 42:299, 42:302, 42:324,
42:333, 42:350, 42:359, 42:383,
42:391, 42:406, 42:413, 42:564,
42:577, 42:725, 42:781, 54:2351,
54:2810, 54:2853, 54:2883, 02:505, 896
`\vsize` 1318
`\vskip` 1283
`\vspace` 18:377, 18:447, 18:448, 18:449
`\vsplit` 54:380, 54:427, 54:2835, 1016
- W**
- `\wedge` 30:377, 30:379
`\whatsit` 04:194, 43
`\widehat` 30:538
`\widetilde` 30:537
`\widowpenalties` 02:106
`\widowpenalty` 24:763, 02:326
`\width` 40:30, 1303
- X**
- `\wlog` 04:6, 04:7,
04:8, 04:54, 37:45, 37:110, 50:414,
02:40, 57:60, 57:711, 02:145, 02:239,
02:254, 02:284, 02:299, 01:84, 1052
`\wp` 30:323
`\wr` 30:393
`\write` 29
- X**
- `\x` 03:96, 03:99, 24:396, 24:397, 57:330, 57:332
`\xdef` 1282
`\XeTeXcharclass` 24:737, 57:39,
57:47, 57:54, 57:67, 57:73, 57:82, 57:89
`\XeTeXcharclassCL` 57:173
`\XeTeXcharclassCM` 57:177
`\XeTeXcharclassEX` 57:174
`\XeTeXcharclassID` 57:171
`\XeTeXcharclassIS` 57:175
`\XeTeXcharclassNS` 57:176
`\XeTeXcharclassOP` 57:172
`\XeTeXcharglyph` 21:1056
`\XeTeXdashbreakstate` 57:273
`\XeTeXglyph` 21:1056
`\XeTeXintercharclasses` .. 57:167, 57:200
`\XeTeXinterchartoks`
..... 57:168, 57:182, 57:183,
57:184, 57:185, 57:186, 57:187,
57:188, 57:189, 57:190, 57:191,
57:192, 57:193, 57:194, 57:195,
57:196, 57:201, 57:206, 57:207,
57:208, 57:209, 57:210, 57:211,
57:212, 57:213, 57:214, 57:215,
57:216, 57:217, 57:218, 57:219, 57:220
`\XeTeXmathcode` 57:161, 57:530
`\XeTeXrevision` 57:41
`\XeTeXuseglyphmetrics` ... 57:270, 57:272
`\XeTeXversion` 29:624, 57:41
`\Xi` 30:312
`\xi` 30:292
`xpos` 36:243, 810
`\xpt` 1280
`\xspaceskip` 02:422
`\xtxHanGlue` 57:180, 57:204,
57:212, 57:213, 57:214, 57:215,
57:216, 57:217, 57:218, 57:219, 57:220
`\xtxHanSpace` ... 57:181, 57:205, 57:206,
57:207, 57:208, 57:209, 57:210, 57:211
- Y**
- `\y` 03:97, 03:99
`\year` 03:11, 03:14, 01:169,
50:1298, 50:1431, 50:1520, 02:386
`ypos` 36:243, 810

File Key: 01=ltmdirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefs.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpara.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidxglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=lthyphen.dtx, 57=ltfinal.dtx

	Z	
\Z	57:252, 57:526, 57:547	\z
		57:243, 57:527, 57:538
		\zeta
		30:284

File Key: 01=ltDirchk.dtx, 02=ltplain.dtx, 03=ltvers.dtx, 04=ltluatex.dtx, 05=ltexpl.dtx, 06=ltdefns.dtx, 07=ltcmd.dtx, 08=lthooks.dtx, 09=ltcmdhooks.dtx, 10=ltsockets.dtx, 11=lttemplates.dtx, 12=ltalloc.dtx, 13=ltcntrl.dtx, 14=lterror.dtx, 15=ltpar.dtx, 16=ltpara.dtx, 17=ltmeta.dtx, 18=ltspace.dtx, 19=ltlogos.dtx, 20=ltfiles.dtx, 21=ltoutenc.dtx, 22=ltcounts.dtx, 23=ltlength.dtx, 24=ltfssbas.dtx, 25=ltfssaxes.dtx, 26=ltfssstrc.dtx, 27=ltfsscmp.dtx, 28=ltfssdcl.dtx, 29=ltfssini.dtx, 30=fontdef.dtx, 31=preload.dtx, 32=ltfntcmd.dtx, 33=lttextcomp.dtx, 34=ltpageno.dtx, 35=ltxref.dtx, 36=ltproperties.dtx, 37=ltmiscen.dtx, 38=ltmath.dtx, 39=ltlists.dtx, 40=ltboxes.dtx, 41=lttab.dtx, 42=ltpictur.dtx, 43=ltthm.dtx, 44=ltsect.dtx, 45=ltfloat.dtx, 46=ltidglo.dtx, 47=ltbibl.dtx, 48=ltmarks.dtx, 49=ltpage.dtx, 50=ltclass.dtx, 51=ltkeys.dtx, 52=ltfilehook.dtx, 53=ltshipout.dtx, 54=ltoutput.dtx, 55=lttagging.dtx, 56=ltthyphen.dtx, 57=ltfinal.dtx