Eine Einführung in LuaLaTeX

Manuel Pégourié-Gonnard <mpg@elzevir.fr>

12. Mai 2013

Dieses Dokument ist eine Karte oder vielmehr ein touristischer Reiseführer für die neue Welt von LuaLaTeX.[1] Die angestrebte Zielgruppe reicht von völligen Anfängern (mit etwas Wissen über gängiges LaTeX) bis hin zu Paket-Entwicklern. Dieser Führer versteht sich als ümfassendim folgenden Sinne: Es enthält Verweise zu allen relevanten Quellen, trägt Informationen zusammen, die sonst verstreut liegen und fügt einführende Materialien bei.

Rückmeldungen und Vorschläge für Verbesserungen sind besonders willkommen. Dieses Dokument ist unter permanenter Bearbeitung. Danke für ihr Verständnis und ihre Geduld.

[1] Obwohl der Fokus des Dokumentes auf LuaLaTeX liegt, beinhaltet es auch nützliche Informationen über LuaTeX im Nur-Text-Format.

1 Einführung

1.1 Was ist eigentlich LuaLaTeX?

Um diese Frage zu beantworten müssen wir ein paar Details über die Welt von TeX aufgreifen, die sie vielleicht für gewöhnlich nicht beachten würden, nämlich den Unterschied zwischen der *Maschine* und einem *Format*. Eine Maschine (engl. *engine*) ist ein Computerprogramm währenddessen ein Format eine Menge von Makros darstellt, die durch die Maschine ausgeführt wird. Üblicherweise werden die Makros vor der eigentlichen Ausführung mit einer konkreten Datei vorgeladen.

Eigentlich ist ein Format mehr oder weniger wie eine Dokumentenklasse oder ein Paket, mit dem Unterschied, dass es mit einem bestimmten Kommando verknüpft ist. Stellen sie sich vor, es gäbe ein Kommando `latex-article`, das das gleiche machen würde, wie `latex`, allerdings brauchen sie dabei nicht die Präambel `\documentclass{article}` zu Beginn ihrer Datei verwenden. Gleichermaßen ist das Kommando `pdflatex` dasselbe wie das Kommando `pdftex`, mit dem Unterschied, dass sie keine Anweisungen am Beginn ihrer Datei machen müssen, LaTeX zu laden. Das ist bequem und auch ein wenig effizienter.

Formate sind großartig, weil sie mächtige Kommandos mit den einfach Werkzeugen implementieren, die eine Maschine bereitstellt. Trotzdem ist die Mächtigkeit eines Formates manchmal durch die von der Maschine bereitgestellten Werkzeuge beschränkt, weshalb einige Leute begonnen haben, funktionsreichere Maschinen zu entwickeln, damit andere wiederum mächtigere Formate (oder Pakete) implementieren können. Die zur Zeit bekanntesten Maschinen (ausgenommen das ursprüngliche TeX) sind pdfTeX, XeTeX und LuaTeX.

Um das Bild noch etwas weiter zu verkomplizieren, erzeugte die ursprüngliche TeX Maschine ausschließlich DVI-Dateien, während seine Nachfolger zusätzlich PDF-Dateien erzeugen können. Jedes Kommando in ihrem System gehört zu einer bestimmten Maschine mit einem bestimmten Format und einem bestimmt Ausgabemodus. Die folgende Tabelle fasst dies zusammen: In der Zeile das Format, in der Spalte Maschine und in jeder Zelle steht oben das Kommando für den DVI-Modus und darunter das für den PDF-Modus.

	TeX	pdfTeX	XeTeX	LuaTeX
Plain	`tex`	`etex`	(none)	`dviluatex`
	(none)	`pdftex`	`xetex`	`luatex`
LaTeX	(none)	`latex`	(none)	`dvilualatex`
	(none)	`pdflatex`	`xelatex`	`lualatex`

Wir können jetzt die Frage in der Überschrift beantworten: LuaLaTeX ist die LuaTeX Maschine mit dem LaTeX Format. Nun gut, diese Antwort ist nicht besonders befriedigend, wenn sie den Unterschied zwischen LuaTeX und LaTeX nicht kennen.

Wie sie vielleicht wissen ist LaTeX das allgemeine Rahmenwerk, in welchem Dokumente mit `\documentclass` beginnen, Pakete mit `\usepackage` geladen, Schriftarten auf eine schlaue Art und Weise ausgewählt werden (damit sie in die Fettschrift wechseln können, aber kursiv beibehalten können), Seiten mit komplizierten Algorithmen erstellt werden, darunter Unterstützung für Kopf- und Fußzeilen, Fußnoten, Randnotizen, Textflussmaterial usw. Dies ändert

sich nicht mit LuaLaTeX, aber neue und mächtigere Pakete werden damit verfügbar, die Teile des Systems auf bessere Weise arbeiten lassen.

Was also ist LuaTeX? Kurz gesagt: Die heißeste TeX Maschine, die es zur Zeit gibt! Etwas genauer ausgedrückt: Es ist der vorgesehene Nachfolger von pdfTeX und beinhaltet all dessen Kernfunktionen: Direkte Generierung von PDF Dateien mit Unterstützung erweiterter PDF-Funktionen und mikrotypografischen Verbesserungen an den typografischen TeX-Algorithmen.

Die wichtigsten Neuerungen von LuaTeX sind:

1. Nativer Support von Unicode, dem modernen Standard von Zeichenklassifikation und Kodierung mit Unterstützung aller Buchstaben der Welt, vom Englischen über Arabisch bis hin zu traditionellem Chinesisch und mit Einbezug einer großen Menge von mathematischen und sonstigen Symbolen.
2. Die Inklusion von Lua als der eingebetteten Scriptsprache (siehe Kapitel 1.3 für Details).
3. Eine Menge von großartigen Lua-Bibliothenken, darunter:
   - `fontloader`, die moderne Schriftartenformate wie Truetype oder Opentype unterstützt;
   - `font`, die die Veränderung von Schriftarten innerhalb des Dokumentes ermöglicht;
   - `mplib`, eine eingebettete Version des Grafikprogramms Metapost;
   - `callback`, die Zugänge zu teilen der TeX-Maschine bereitstellt, die vorher nicht erreichbar waren;
   - Dienstprogramm-Bibliotheken, um Bilder, PDF und andere Dateien zu manipulieren.

Einige dieser Funktionen, wie zum Beispiel die Unicode-Unterstützung, wirken sich direkt auf alle Dokumente auf, während andere Funktionen hauptsächlich Werkzeuge bereitstellen, die Autoren von Paketen nutzen, um noch noch mehr Kommandos und Erweiterungen bereitzustellen.

1.2 Wechseln von LaTeX nach LuaLaTeX

Wie das vorangegangene Kapitel zeigt ist LuaLaTeX vergleichbar mit LaTeX, hat einige wenige Unterschiede und hält eine höhere Anzahl von funktionsreichen Paketen und Werkzeugen verfügbar. In diesem Abschnitt präsentieren wir das absolute Minimum, das sie wissen sollten, um ein Dokument mit LuaLaTeX zu erzeugen, während der Rest des dieses Dokumentes mehr Details über die verfügbaren Packages bereithält.

Es gibt nur drei Unterschiede: There are only three differences:

1. Verwenden sie nicht das Paket inputenc! Kodieren sie ihre Quelldatei einfach in UTF-8.
2. Laden sie nicht fontenc, sondern stattdessen fontspec.
3. Benutzen sie kein Paket, das die Schriftarten ändert, stattdessen verwenden sie fontspec's-Kommandos.

Sie müssen sich also lediglich mit fontspec vertraut machen, was sehr einfach ist: Wählen sie die Hauptserifenschriftart mit `\setmainfont` aus, die serifenlose Schriftart mit `\setsansfont`, die Festbreitenschriftart mit `\setmonofont`. Der Parameter für diese Kommandos ist ein menschenlesbarer Name der Schriftart, zum Beispiel `Latin Modern Roman` anstelle von `ec-lmr10`. Sie möchten gegebenenfalls `\defaultfontfeatures{Ligatures=TeX}` vor der Verwendung dieser Kommandos setzen, um die üblichen TEX-Ersetzungen (wie zum Beispiel `---` für einen hervorgehobenen Gedankenstrich) zu bewahren.

Die gute Nachricht ist, dass sie jetzt direkt auf jede Schriftart ihres Betriebssystems zugreifen können, zusätzlich zu denen der TEX-Distribution, darunter auch TrueType und OpenType-Schriftarten und sie haben Zugriff auf deren neueste Funktionen. Das bedeutet, dass es jetzt einfach ist ist, mit LuaLATEX jede moderne Schriftart zu verwenden, die sie heruntergeladen oder gekauft haben - und sie profitieren von deren vollem Potenzial.

Die schlechte Nachricht ist, dass es nicht immer einfach ist, eine Liste von allen verfügbaren Schriftarten zu erhalten. Unter Windows mit TEX Live listet ihnen das Kommandozeilenwerkzeug `fc-list` alle auf, aber es ist nicht besonders nutzerfreundlich. Unter Mac OS X, zeigt ihnen die *Fontbook*-Anwendung alle Schriftarten ihres Systems an, aber nicht die der TEX-Distribution. Das Gleiche gilt für `fc-list` unter Linux. Eine weitere schlechte Nachricht ist, dass sie auf diese Art nicht auf ihre alten Schriftarten zugreifen können. Glücklicherweise gibt es tagtäglich immer mehr neue Schriftarten in modernen Formaten.

Übrigens sollten wir hier erwähnen, dass der Inhalt dieses Kapitels so weit auch auf XeLATEX zutrifft, das heißt für LATEX nach XeTEX. Tatsächlich teilt XeTEX zwei der essenziellen Features von LuaTEX, nämlich die native Unicode-Unterstützung und die Verwendung moderner Schriftartenformate (allerdings ohne die anderen Funktionen von LuaTEX; auf der anderen Seite ist es zur Zeit stabiler). Obwohl sich die XeLATEX-Implementierung bezüglich Schriftarten erheblich unterscheidet, so stellt doch fontspec eine vereinheitlichte Schriftarten-Schnittstelle für beide bereit.

Um die Vorzüge der neuen Funktionen von LuaTEX zu erhalten, müssen sie einige Teile der alten Welt fallenlassen und zwar jene Schriftarten, die nicht in einem modernen Format verfügbar sind. Auch die Freiheit der Kodierung der Quelldatei nach Wahl muss aufgegeben werden, aber UTF-8 ist so überlegen, dass dieser Punkt kaum zählt. Das Paket luainputenc stellt verschiedene Krücken bereit, um diese Teile zu erhalten[2], allerdings mit dem Nachteil, die korrekte Unicode-Unterstützung zu verlieren.

Das ist alles, was sie wissen müssen, um Dokumente mit LuaLATEX zu erstellen. Ich empfehle ihnen einen Blick ins die fontspec-Anleitung zu werfen und konkret ein kleines Dokument mit lustigen Schriftarten zu übersetzen. Sie können dann den Rest des Dokuments nach Bedarf überfliegen. Kapitel 5 listet alle anderen Unterschiede zwischen konventionellem LATEX und LuaLATEX auf, die mir bekannt sind.

[2] Obwohl der Name andeutet, es ginge lediglich um Eingabe-Kodierungen, so sind die Details der LATEX-Schriftartenimplementierung für dieses Paket notwendig und unterstützen daher ebenfalls die alten Schriftarten.

1.3 Eine Lua-in-TEX Fibel

Lua ist eine kleine, feine Sprache ganz offensichtlich weniger überraschend und viel einfacher zu lernen als TEX. Die Haupt-Referenz ist das sehr gut lesbare Buch *Programming in Lua*, dessen erste Ausgabe online frei verfügbar ist. Für den Schnelleinstieg empfehle ich die Kapitel 1 bis 5 zu lesen und einen schnellen Blick auf Teil 3 zu werfen. Beachten sie bitte, dass alle Bibliotheken, die in Kapitel 3 erwähnt werden, auch Bestandteil von LuaTEX sind. Allerdings ist die `os`-Bibliothek aus Sicherheitsgründen beschränkt.

Je nach ihrer Programmierweise sind sie vielleicht gleich an den restlichen Abschnitten von Teil 1 und Teil 2, die ihnen die erweiterten Funktionalitäten der Sprache aufzeigen. Teil 4 ist bedeutungslos im Kontext von LuaTEX, sofern sie nicht an LuaTEX selbst entwickeln wollen. Schließlich ist das *Lua Referenz Handbuch* in einigen Sprachen online verfügbar und beinhaltet einen praktischen Index.

Kommen wir nun zu Lua in LuaTEX. Die gebräuchlichste Art, Lua-Code auszuführen ist das `\directlua`-Kommando, das Lua code als Argument erhält. Genauso können sie Informationen von Lua nach TEX übergeben mittels `tex.sprint`.[3] Zum Beispiel

```
die Standardnäherung von $\pi = \directlua{tex.sprint(math.pi)}$
```

gibt "die Standardnäherung von $\pi = 3.1415926535898$" in ihr Dokument aus. Sie sehen, wie einfach es ist TEX und Lua zu vermischen?

Tatsächlich gibt es einige wenige Treffer. Schauen wir zuerst auf den Lua nach TEX-Weg, denn es ist der einfachste. Wenn sie in das LuaTEX-Handbuch schauen, werden sie sehen, dass es dort eine weitere Funktion mit einem einfacheren Namen `tex.print`, gibt. Es fügt eine komplette Zeile in ihre TEX-Quelle ein, wobei der Inhalt das Argument des Kommandos ist. Falls sie das noch nicht wussten: TEX macht alle garstigen Dinge mit kompletten Quellcodezeilen, wie das Überspringen am Anfang und am Ende einer Zeile und dem Hinzufügen eines EOL-Zeichens am Ende der Zeile. Die meiste Zeit werden sie das Feature nicht brauchen, daher empfehle ich die Verwendung von `tex.sprint`.

Wenn sie genügend tief in TEX bewandt sind und sogenannte catcodes kennen, werden sie sich freuen, dass ihnen `tex.print` und seine Varianten beinahe die volle Kontrolle über catcodes gibt, die sie benutzen können, um die Argumente zu zerlegen, da sie eine catcode-Tabelle als erstes Argument übergeben können. Sie möchten vielleicht mehr über catcodes erfahren, die zur Zeit 2.7.6 im LuaTEX Handbuch sind. Wenn sie mit catcodes nichts am Hut haben, überspringen sie einfach diesen Abschnitt.

Lassen sie uns nun auf `\directlua` blicken. Um einen Eindruck davon zu bekommen wie es arbeitet, stellen sie sich vor, dass es ein `\write`-Kommando sei, aber es schreibt nur in eine virtuelle Datei und schickt diese Datei sofort an den Lua-Interpreter. Im Lua-Kontext ergibt sich daruas, dass jedes Argument eines `\directlua`-Kommandos seinen eigenen Geltungsbereich hat: Lokale Variablen des einen Argumentes sind nicht sichtbar für das andere (was eigentlich vernünftig ist, aber man weiß besser darüber bescheid).

[3] Der Name bedeutet vermutlich "Zeichenkettendruck" im Gegensatz zu "laufe sehr schnell für eine sehr kurze Zeit."

Nun der wichtigste Punkt ist, dass das Argument zuerst vom TEX-Interpreter zerlegt wird, dann vollständig erweitert und zurückverwandelt wird in eine reine Zeichenkette, bevor es in den Lua-Interpreter gesteckt wird. Das Einlesen in den TEX-Interpreter hat einige Konsequenzen. Eine davon ist, dass die EOL-Zeichen in Leerzeichen umgewandelt werden, sodass es nur eine (lange) Eingabezeile gibt. Da Lua eine formfreie Sprache ist, spielt das für gewöhnlich keine Rolle, allerdings dann schon, wenn es um Kommentare geht:

```
\directlua{a_function()
  -- a comment
  another_function()}
```

wird nicht das tun, was sie vermutlich erwarten würden. `another_function()` würde als Teil des Kommentars verstanden werden.

Eine weitere Konsequenz ist, dass aufeinanderfolgende Leerzeichen werden zusammengefasst zu einem Leerzeichen und TEX-Kommentare werden verworfen. Hier nun die korrekte Version des vorigen Beispiels:

```
\directlua{a_function()
  % a comment
  another_function()}
```

Es ist ebenfalls zu bemerken, dass das Argument general in einem `\write` ist, und daher im expansion-only-Kontext ist. Wenn sie nicht wissen, was das bedeutet, lassen sie es mich so sagen, dass die Expansion-Sache der Hauptgrund ist, der die TEX-Programmierung so schwierig gestaltet.

Ich entschuldige mich dafür, dass die letzten drei Abschnitte ein wenig TEXnisch gewesen sind, aber ich dachte mir, dass sie das wissen sollten. Um sie dafür zu belohnen, dabei geblieben zu sein, gebe ich ihnen hier einen debugging-Trick. Setzen sie den folgenden Code an den Anfang ihres Dokumentes:

```
\newwrite\luadebug
\immediate\openout\luadebug luadebug.lua
\AtEndDocument{\immediate\closeout\luadebug}
\newcommand\directluadebug{\immediate\write\luadebug}
```

Wenn sie einmal schwer mit einem speziellen Aufruf von `\directlua` zu kämpfen haben, weil er nicht das macht, was sie wollen, dann ersetzen sie den Aufruf dieser Instanz durch `\directluadebug`. Übersetzen sie dann wie immer und schauen in die Datei `luadebug.lua`. Sie sehen dannm, was der Lua-Interpreter tatsächlich gelesen hat.

Das luacode-Pakte stellt Kommandos und Umgebungen bereit, die ihnen helfen verschiedene Grade dieser Probleme zu lösen. Wie dem auch seim, für alles was nicht trivial ist, sollte man eine externe Datei mit dem Lua-Code verwenden, sie laden und verwenden. Zum Beispiel:

```
\directlua{dofile("my-lua-functions.lua")}
\newcommand*{\greatmacro}[2]{%
  \directlua{my_great_function("\luatexluaescapestring{#1}", #2)}}
```

Das Beispiel nimmt an, dass `my_great_function` in `my-lua-functions.lua` definiert ist und eine Zeichenkette und eine Zahl als Argument nimmt. Bemerken sie auch, dass wir vorsichtig die `\luatexluaescapestring` Primitive auf das Zeichenkettenargument anwenden, um jeden backslash oder Gänsefüßchen zu escapen. Das würde sonst den Lua Parser durcheinanderbringen.[4]

Das ist alles, was Lua in TEX betrifft. Wenn sie sich darüber wundern, warum `\luatexluaescapestring` einen so langen und dummen Namen hat, dann wollen sie vielleicht das nächste Kapitel anschauen.

1.4 Andere wissenswerte Dinge

Nur für den Fall, dass dies nicht offensichtlich ist: Das LuaTEX-Handbuch, `luatexref-t.pdf` ist eine großartige Quelle.

Es ist wichtig zu wissen, dass die Namen der Primitive von LuaTEX im Handbuch nicht mit den aktuellen Namen übereinstimmen. Um Konflikte mit bestehenden Makros zu vermeiden, sind alle neuen Primitive mit der Voranstellung `\luatex` versehen worden, es sei denn, sie begannen nicht bereits schon mit diesem Prefix.

To prevent clashes with existing macro names, all new primitives have been prefixed with `\luatex` unless they already start with it, so `\luaescapestring` becomes `\luatexluaescapetring` while `\luatexversion` remains `\luatexversion`. The rationale is detailed in section 4.

Oh, and by the way, did I mention that LuaTEX is in beta and version 1.0 is expected in late 2012? You can learn more on the roadmap page of the LuaTEX site. Stable betas are released regularly and are included in TEX Live since 2008 and MikTEX since 2.9.

Not surprisingly, support for LuaTEX in LATEX is shiny new, which means it may be full of (shiny) bugs, and things may change at any point. You might want to keep your TEX distribution very up-to-date[5] and also avoid using LuaLATEX for critical documents at least for some time.

As a general rule, this guide documents things as they are at the time it is written or updated, without keeping track of changes. Hopefully, you'll update your distribution as a whole so that you always get matching versions of this guide and the packages, formats and engine it describes.

2 Essenzielle Pakete und Praktiken

Dieser Abschnitt präsentiert die Pakete, die Sie als Anwender möglicherweise immer laden wollen oder über die Sie als Entwickler auf jeden Fall Bescheid wissen sollten.

2.1 Anwender-Level

fontspec **Engines:** XeTEX, LuaTEX. **Formats:** LATEX.
Authors: Will Robertson.

[4] Wenn sie jemals schonmal SQL verwendet haben, dann ist das Konzept, Zeichenketten zu escapen hoffentlich nicht neu für sie.

[5] For TEX Live, consider using the complementary tlcontrib repository.

CTAN location: macros/latex/contrib/fontspec/.
Development url: https://github.com/wspr/fontspec/.
Schönes Interface zur Schriftartenverwaltung, welches gut in die LaTeX Schriftartenauswahlschema integriert ist. Dies wurde schon in einem vorherigen Abschnitt vorgestellt.

2.2 Entwickler-Level

2.2.1 Naming conventions

Auf der TeX Seite sind Kontrollsequenzen, die mit `\luatex` beginnen, für Primitiven reserviert. Es wird stark empfohlen, dass Sie *keine* solcher Kontrollsequenzen definieren, um Namensueberschneidungen mit zukünftigen Versionen von LuaTeX zu verhindern. Sollten Sie hervorheben wollen, dass ein Makro LuaTeX-spezifisch ist, empfehlen wir, dass Sie stattdessen das `\lua` Präfix (ohne folgendem `tex`) benutzen. Es ist in Ordnung, das `\luatex@` Präfix für interne Makros zu benutzen, da primitive Namen nie `@` enthalten, dennoch könnte dies Verwirrung stiften. Ausserdem benutzen Sie ja schon ein einzigartigen Präfix für interne Makros in allen Ihren Paketen, nicht wahr?

Auf der Lua-Seite halten Sie bitte den globalen Namespace so sauber wie möglich. Das heisst, Sie verwenden eine Tabelle `mypackage` and setzen alle Ihre öffentlichen Funktionen und Objekte in diese. Sie möchten möglicherweise dafür Lua's `module()` benutzen. Andere Strategien für die Lua Modulverwaltung werden in Kapitel 15 aus *Programming in Lua* diskutiert. Es ist ausserdem eine gute Idee `local` für Ihre internen Variablen und Funktionen zu verwenden. Zu guter Letzt wird zur Vermeidung von Überschneidungen mit zukünftigen Versionen von LuaTeX empfohlen, die Namespaces von LuaTeX's Standardbibliotheken nicht zu verändern.

2.2.2 Engine und Modusfeststellung

Zahlreiche Pakete erlauben es, die Engine festzustellen, die gegenwärtig mit der Verarbeitung des Dokuments beschäftigt ist.

ifluatex **Engines:** all. **Formats:** LaTeX, Plain.
Authors: Heiko Oberdiek.
CTAN location: macros/latex/contrib/oberdiek/.
Stellt `\ifluatex` zur Verfügung und stellt sicher, dass `\luatexversion` benutzbar ist.

iftex **Engines:** all. **Formats:** LaTeX, Plain.
Authors: Vafa Khalighi.
CTAN location: macros/latex/contrib/iftex/.
Development url: http://bitbucket.org/vafa/iftex.
Stellt `\ifPDFTeX`, `\ifXeTeX`, `\ifLuaTeX` und korrespondierendes `\Require` Kommandos zur Verfügung.

expl3 **Engines:** all. **Formats:** LaTeX.
Authors: The LaTeX3 Project.
CTAN location: macros/latex/contrib/expl3/.
Development url: http://www.latex-project.org/code.html.

Stellt neben *vieler* anderer Dinge `\luatex_if_engine:TF`, `\xetex_if_engine:TF` und deren Varianten zur Verfügung.

ifpdf **Engines:** all. **Formats:** LaTeX, Plain.
Authors: Heiko Oberdiek.
CTAN location: `macros/latex/contrib/oberdiek/`.
Stellt den `\ifpdf` Schalter zur Verfügung. LuaTeX, wie auch pdfTeX kann sowohl PDF als auch DVI Ausgaben erzeugen; letzteres ist nicht wirklich sinnvoll mit LuaTeX, da es keine erweiterten Funktionen wie Unicode und moderne Schriftformate unterstützt. Der `\ifpdf` Schalter ist nur aktiv, wenn und nur wenn Sie pdfTeX oder LuaTeX im PDF Modus laufen lassen (beachten Sie dass dies nicht für XeTeX gilt, dessen PDF Unterstuetzung anders funktioniert).

2.2.3 Grundlegende Ressourcen

luatexbase **Engines:** LuaTeX. **Formats:** LaTeX, Plain.
Authors: Élie Roux & Manuel Pégourié-Gonnard.
CTAN location: `macros/luatex/generic/luatexbase/`.
Development url: `https://github.com/mpg/luatexbase`.
Die Text- und LaTeX-Formate enthalten Makros, um grundlegende TeX Ressourcen zu verwalten, wie zum Beispiel Zähl- und Kistenregister. LuaTeX führt neue Ressourcen ein, die anstandslos von Paketen gemeinsam benutzt werden können müssen. Dieses Paket stellt die grundlegenden Werkzeuge dafür zur Verfügung: die erweiterten konventionellen TeX Ressourcen, catcode Tabellen, Attribute, Callbacks sowie zum Laden und Identifizieren von Lua Modulen. Ausserdem verfügt es über Werkzeuge, um einige Kompatibilitätsprobleme mit älteren Versionen von LuaTeX zu händeln.
Warnung. Dieses Paket ist derzeitig im Konflikt mit dem luatex Paket, da sie nahezu das Gleiche machen. Die entsprechenden Paketautoren sind sich dieser Situation bewusst und planen die beiden Pakete in der nahen Zukunft zu verschmelzen. Noch ist nicht klar, wann dies geschehen wird.

luatex **Engines:** LuaTeX. **Formats:** LaTeX, Plain.
Authors: Heiko Oberdiek.
CTAN location: `macros/latex/contrib/oberdiek/`.
Sehen Sie sich die Beschreibung des oben an. Dieses Paket stellt die selben Kernfunktionalitaeten, ausser Callback-Management und Lua Moduldentifizierung, zur Verfügung.

lualibs **Engines:** LuaTeX. **Formats:** Lua.
Authors: Élie Roux.
CTAN location: `macros/luatex/generic/lualibs/`.
Development url: `https://github.com/mpg/lualibs`.
Eine Zusammenstellung von Lua-Bibliotheken und Erweiterungen der Standardbibliotheken; meist abgeleitet von den ConTeXt-Bibliotheken. Wenn Sie eine grundlegende Funktion brauchen, die Lua nicht zur Verfügung stellt, dann prüfen Sie dieses Paket, bevor Sie eine eigene Implementierung vornehmen.

2.2.4 Font internals

Diese Pakete werden von fontspec geladen, um einige Low-Level-Probleme von Schriften und Encoding zu haendeln. Ein normaler Anwender sollte nur fontspec nutzen, aber ein Entwickler sollte darüber Kenntnis haben.

luaotfload **Engines:** LuaTeX. **Formats:** LaTeX, Plain.
Authors: Élie Roux & Khaled Hosny.
CTAN location: macros/luatex/generic/luaotfload/.
Development url: https://github.com/khaledhosny/luaotfload.
Low-level Open Type Schriften Lader, adaptiert von einem Teil von ConTeXt. Grundsätzlich benutzt es die fontloader Lua-Bibliothek und die entsprechenden Callbacks, um eine Syntax für die \font Primitive ähnlich wie tetex bereitzustellen, und die entsprechenden Schriftenfunktionalitäten umzusetzen. Zudem verwaltet es auch eine Schriftendatenbank für transparenten Zugriff auf die Schriften im System und der TeX Distribution entweder durch den Schriftfamiliennamen oder Dateinamen. Ausserdem hat es auch einen Schriftencache für schnelleres Laden.

euenc **Engines:** XeTeX, LuaTeX. **Formats:** LaTeX.
Authors: Will Robertson, Élie Roux & Khaled Hosny.
CTAN location: macros/latex/contrib/euenc/.
Development url: https://github.com/wspr/euenc.
Setzt die EUx Unicode Schrift Enkodierungen für LaTeX's fontenc System um. Derzeit benutzt XeLaTeX EU1 und LuaTeX EU2. Enthält Definitionen (fd Dateien) für Latin Modern, die von fontspec geladene Standardschrift.

3 Andere Pakete

Bitte beachten Sie, dass diese Pakete nicht in einer bestimmten Reihenfolge aufgelistet sind.

3.1 Anwender-Level

luatextra **Engines:** LuaTeX. **Formats:** LaTeX.
Authors: Élie Roux & Manuel Pégourié-Gonnard.
CTAN location: macros/luatex/latex/luatextra/.
Development url: https://github.com/mpg/luatextra.
Lädt die üblichen Pakete, derzeitig fontspec, luacode, metalogo (Kommandos für Logos, unter anderem \LuaTeX und \LuaLaTeX), luatexbase, lualibs, fixltx2e (Fixes und Erweiterungen fuer den LaTeX-Kern).

luacode **Engines:** LuaTeX. **Formats:** LaTeX.
Authors: Manuel Pégourié-Gonnard.
CTAN location: macros/luatex/latex/luacode/.
Development url: https://github.com/mpg/luacode.
Stellt Kommandos und Makros zur Verfügung, die dabei helfen, Lua Code in eine TeX-Quelle einzufügen, insbesondere Sonderzeichen.

luainputenc **Engines:** LuaTeX, XeTeX, pdfTeX. **Formats:** LaTeX.
Authors: Élie Roux & Manuel Pégourié-Gonnard.
CTAN location: macros/luatex/latex/luainputenc/.
Development url: https://github.com/mpg/luainputenc.
Hilft, Dokumente zu kompilieren, die auf veralteten Enkodierenden basieren (sei es in der Quelle oder in den Schriften). Wurde schon in der Einfuehrung vorgestellt. Wenn XeTeX benutzt wird, dann lädt es einfach xetex-inputenc; unter pdfTeX laedt es das standardmäßige inputenc.

luamplib **Engines:** LuaTeX. **Formats:** LaTeX, Plain.
Authors: Hans Hagen, Taco Hoewater & Élie Roux.
CTAN location: macros/luatex/generic/luamplib/.
Development url: https://github.com/mpg/luamplib.
Stellt eine schöne Schnittstelle für die `mplib` Lua Bibliothek zur Verfügung die Metapost in LuaTeX einbettet.

luacolor **Engines:** LuaTeX. **Formats:** LaTeX.
Authors: Heiko Oberdiek.
CTAN location: macros/latex/contrib/oberdiek/.
Ändert die Low-Level-Farbenimplementierung, um LuaTeX Attribute anstatt whatsits. Dies macht die Implementierung robuster und korrigiert seltsame Fehler, wie zum Beispiel die fehlerhafte Ausrichtung, wenn `\color` zu Beginn einer `\vbox` gesetzt ist.

luadirections **Engines:** LuaTeX. **Formats:** LaTeX, Plain, ConTeXt.
Authors: Khaled Hosny.
Development url: https://github.com/khaledhosny/luadirections.
Hoehergelegene Schnittstelle zu multi-direcktionaler Unterstützung. Derzeit nicht im CTAN vorhanden.

3.2 Entwickler-Level

pdftexcmds **Engines:** LuaTeX, pdfTeX, XeTeX. **Formats:** LaTeX, Plain.
Authors: Heiko Oberdiek.
CTAN location: macros/latex/contrib/oberdiek/.
Auch wenn LuaTeX normalerweise eine Erweiterung von pdfTeX ist, wurden dennoch einige Primitiven entfernt (die, die durch Lua sozusagen abgelöst wurden) oder umbenannt. Dieses Paket stellt diese Primitiven mit konsistenten Namen durchgängig allen Engines zur Verfügung, unter anderem XeTeX, welches vor Kurzem einige dieser Primitiven implementiert hat, zum Beispiel `\strcmp`.

magicnum **Engines:** LuaTeX, pdfTeX, XeTeX. **Formats:** LaTeX, Plain.
Authors: Heiko Oberdiek.
CTAN location: macros/latex/contrib/oberdiek/.
Stellt einen hierarchischen Zugriff auf “magic numbers”, wie Catcodes, Gruppentypen usw., die intern von TeX und seinen Nachfolgern benutzt werden, zur Verfügung. In LuaTeX wird eine effizientere Implementierung benutzt und eine Lua-Schnittstelle ist bereitgestellt.

lua-alt-getopt **Engines:** texlua. **Formats:** command-line.
Authors: Aleksey Cheusov.
CTAN location: support/lua/lua-alt-getopt.
Development url: http://luaforge.net/project/lua_altgetopt.
Kommandozeilen-Options-Parser, nahezu vollständig kompatibel mit POSIX und GNU getopt, zu benutzen in Kommandozeilen-LUA-Skripten, zum Beispiel mkluatexfontdb aus luaotfload.

4 Die luatex - und lualatex-Formate

Dieser Abschnitt ist nur für Entwickler und neugierige Anwender gedacht; normale Anwender können dies gefahrlos überspringen. Die folgende Information bezieht sich auf TEX Live 2010 und höchstwahrscheinlich auch auf MikTEX 2.9, auch wenn ich dies nicht überprüft habe. Frühere Versionen von TEX Live hatten eicht unterschiedliche und unvollständigere Arrangements.

Primitivennamen Wie im Abschnitt 1.4 erwähnt wurde, sind die Namen der LuaTEX-spezifischen Primitiven nicht die selben im lualatex Format wie im LuaTEX Handbuch. Im lualatex Format (das heisst LuaTEX mit dem Plain Format) sind die Primitiven mit ihrem natürlichen Namen verfügbar, und zusätzlich auch mit einem Präfix vorangestellten Namen, um die Entwicklung von generischen Paketen zu erleichtern.

Die Rationale, entnommen der Datei lualatexiniconfig.tex , die dies für das lualatex Format umsetzt, ist:

1. Alle derzeitigen Makropakete laufen unproblematisch auf pdf(e) TeX, daher sind diese Primitiven unberührt.
2. Andere nicht-TeX82 Primitiven in LuaTEX können Namensueberschneidungen mit existierenden Makros in Makropaketen verursachen, besonders wenn sie sehr "natürliche" Namen benutzen, wie zum Beispiel \outputbox, \mathstyle usw. Solch eine Wahrscheinlichkeit für Überschneidungen ist nicht gewünscht, da die meisten existierenden LaTeX-Dokumente ohne Änderungen unter LuaTEX laufen.
3. Das LuaTEX Team möchte keine systematische Präfixrichtlinie vorschreiben, stellt aber netterweise ein Werkzeug zur Verfügung, mit dem Präfixe gesetzt werden können. Deshalb haben wir uns entschieden, dies zu nutzen. Vorher deaktivierten sogar die Extra-Primitiven, aber nun denken wir, dass es besser ist, sie mit der systematischen Präfixvoranstellung zu aktivieren, um zu vermeiden, dass jedes Makropaket (oder auch Anwender) diese mit vielfaltigen und inkonsistenten Präfixen aktiviert (inkl. dem leeren Präfix).
4. Der luatex-Praefix wurde ausgewählt, da er bereits als Präfix für einige Primitiven benutzt wird, wie zum Beispiel \luatexversion: dadurch erhalten diese nicht am Ende einen Doppelpräfix (für Details siehe tex.enableprimitives im LuaTEX-Handbuch).
5. Die \directlua Primitive wird mit ihrem natürlichen (erlaubt die leichte Festellung von LuaTEX) sowie mit einem Präfix versehen Namen (\luatexdirectlua (für Konsistenz mit \luatexlatelua)) bereitgestellt.
6. Einige Anmerkungen:
   - Der offensichtliche Nachteile einer solchen Präfixrichtlinie ist, dass die Namen, die von LATEX oder einem generischen Makroersteller benutzt werden, nicht mit den Namen im

Handbuch übereinstimmen. Wir hoffen, dass sich dies durch den Gewinn der Rückwärtskompatibilitaet ausgleicht.

- Alle Primitiven, die das Thema Unicode Mathematik behandeln beginnen bereits mit `\U` und werden möglicherweise in der Zukunft mit den Namen der XeTeX-Primitiven übereinstimmen, sodass möglicherweise ein Präfix für diese nicht notwendig oder gewünscht waren. Nichtsdestotrotz haben wir versucht, die Präfixregeln so einfach wie möglich zu halten, sodass der vorherige Punkt nicht noch schlimmer wird.
- Der endgültige Name einiger Primitiven möge sich seltsam anhören, besonders, die, die bereits den Namen einer Engine enthalten, wie zum Beispiel `\luatexOmegaVersion`. Da aber LuaTeX nicht einfach ein Ersatz für Omega/Aleph ist, empfanden wir es als falsch, `\OmegaVersion` zur Verfügung zu stellen.
- Vielleicht empfinden wir es eines Tages besser, alle Primitiven ohne Präfix bereitzustellen. Wenn dies passiert, wird es einfach sein du Primitiven ohne Präfix in dem Format hinzuzufügen, während die Namen mit Präfix aus Kompatibilitätsgründen beibehalten werden. Andersherum würde dies nicht funktionieren; zum Beispiel zu spät zu erkennen, dass wir die Primitiven ohne Präfix nicht hätten bereitstellen sollen würde dann alle LuaTeX-spezifische Makropakete kaputtmachen, die bereits geschrieben wurden.

`\jobname` Der LaTeX Kernel (and eine Menge Pakete) benutzen Konstrukte wie `\input\jobname.aux` aus verschiedensten Gründen. Wenn `\jobname` Leerzeichen enthält, funktioniert dies nicht richtig, da die Argumente von `\input` beim ersten auftretenden Leerzeichen enden. Um dies zu umgehen, setzt pdfTeX automatisch `\jobname` in Anführungszeichen wenn gebraucht, aber LuaTeX tut dies aus unerfindlichen Gründen nicht. Ein nahezu vollständiger Workaround ist in den LaTeX-basierten (im Gegensatz zu den plain-basierten) LuaTeX-Formaten vorhanden.

Dies funktioniert aber nicht, wenn LuaTeX als `lualatex†'\input†name'` aufgerufen wird, im Gegensatz zum gebräuchlicheren `lualatex†name`. Um diese Einschränkung zu umgehen, kann ein Jobname explizit angegeben werden, wie zum Beispiel `lualatex†jobname=name '\input†name'`. Oder besser noch, benutzen Sie keine Leerzeichen in den Namen Ihrer TeX-Dateien.

Für mehr Details siehe diesen alten Eintrag und ein neuerer Eintrag auf der LuaTeX Mailingliste, und `lualatexquotejobname.tex` für die Umsetzung eines Workarounds.

babel LuaTeX bietet dynamisches Laden von Silbentrennungsmustern. Derzeitig gibt es keine Unterstützung dafür in babel aber einige Dateien wurden angepasst, um ein halbdynamisches Laden bereitzustellen, welches eine bessere Ladezeit des Formates erreicht. Dies ist nur eine Änderung in der Implementierung; nichts sollte auf der Anwenderebene sichtbar sein. Ein verändertes Muster-Ladeschema wird auch für Plain-basierten benutzt.

Dokumentation und Implementierungsdetails sind enthalten in `luatex-hyphen.pdf`. Die Quellen sind Teil des texhyphen Projektes.

codes Die Engine setzt selbst keine `\catcode`s, `\lccode`s, usw. für nich-ASCII Zeichen. Korrekte `\lccode`s sind im Besonderen essenziell, so dass Silbentrennung funktioniert. Die Formate für LuaTeX enthalten nun `luatex-unicode-letters.tex`, eine angepasste Version von `unicode-letters.tex` aus der XeTeX Distribution, welches die entsprechenden Einstellungen konform zum Unicode Standard setzt.

Dies wurde hinzugefügt, nachdem TEX Live 2010 ausgeliefert. Daher wird stark empfohlen, dass Sie Ihre Installation aktualisieren, wenn Sie in den Genuss einer korrekten Silbentrennung für nicht-ASCII Text kommen wollen.

5 Dinge, die einfach funktionieren, teilweise funktionieren oder (noch) gar nicht funktionieren

5.1 Voll funktionierend

Unicode Konventionelles LATEX bietet etwas Unterstützung für UTF-8 in Eingabedateien an. Aber auf einer niedrigeren Ebene werden nicht-ASCII Zeichen in diesem Fall nicht atomar behandelt: sie bestehen auch mehreren elementaren Teilen (den TEXnikern als *Tokens* bekannt). Demzufolge haben Pakete, die Text Zeichen für Zeichen scannen oder andere atomare Operationen auf Zeichen ausüben(z.B. das Ändern ihrer catcodes), oft Probleme mit UTF-8 in konventionellem LATEX. Zum Beispiel können for nicht-ASCII Zeichen kein short verbatim mit *tokens* benutzt werden usw.

Die gute Nachricht ist, dass mit LuaLATEX einige der Features dieser Pakete damit beginnen, mit beliebigen Unicode Zeichen zu funktionieren ohne das Paket anpassen zu müssen. The schlechte Nachricht ist, dass dies nicht immer wahr ist. Siehe nächsten Abschnitt für die Details.

5.2 Teilweise funktionierend

microtype Paket microtype hat eingeschränkte Unterstützung für LuaTEX: präziser gesagt, seit Version 2.4 2010/01/10 sind Protrusion und Expansion verfügbar und standardmäßig aktiviert im PDF Modus, aber Kerning, Spacing und Tracking werden nicht unterstützt (siehe Tabelle 1 in Abschnitt 3.1 von `microtype.pdf`).

Auf der anderen Seite, besitzt luaotfload, dass durch fontspec geladen wird, eine Menge an Mikrotypografischer Funktionen. Das einzige Problem besteht daher im Fehlen einer einheitlichen Schnittstelle.

xunicode

Das Paket xunicode hat als Hauptaufgabe sicherzustellen, dass die üblichen Steuersequenzen für nicht-ASCII-Zeichen (wie etwa `\'e`) im Unicode-Kontext korrekt arbeiten. Es könnte *vermutlich* mit LuaTEX arbeiten, aber nur mit expliziten Tests für XETEX. Wie dem auch sei, fontspec benutzt einen Trick, um es dennoch zu laden. Daher können Sie es nicht explizit laden, müssen es auch nicht, da fontspec sich ohnehin darum kümmert.

encodings

Wie im vorherigen Abschnitt erwähnt, gibt es einige Dinge, die mit UTF-8 unter konventionellem Latex manchmal funktionieren, aber eben nicht immer. Zum Beispiel können sie mit dem listings-Paket unter LuaLATEX innerhalb ihrer Listings nur Zeichen unterhalb von 256 verwenden - das heißt, Zeichen aus dem Latin-1 Zeichensatz.

metrics

Dieser Abschnitt handelt nicht konkret von ”funktionierenöder ”nicht funktionieren”, sondern eher, dass es nicht wie pdfTEX oder XETEX funktioniert. Sie werden kleinere Unterschiede im Layout und der Silbentrennung ihres Textes bemerken. Diese entstehen wegen der Unterschiede zwischen zwei Versionen derselben Schriftart bei Anwendung von verschiedenen Maschinen, Anpassungen an der Silbentrennung, Ligaturen oder Kerning-Algorithmen, oder Unterschieden in den verwendeten Silbentrennungs-Mustern (Muster von pdfTEX frieren in der Regel ein, aber LuaTEX und XETEX nutzen neuere Versionen für Sprachen).

Wenn Sie jemals einen größeren Unterschied zwischen pdfLATEX und LuaLATEX mit der selben Schriftart sehen, dann ist es nicht ungewöhnlich, dass es sich um einen Fehler in LuaLATEX[6] handelt. Wie für gewöhnlich stellen Sie sicher, dass ihre Distribution aktuell ist.

babel

Kurz gesagt: Es arbeitet fast immer ohne Probleme mit latinisierten Sprachen. Für andere Sprachen kann es Probleme geben. Ein moderneres, weniger fertigeres Paket für die Mehrsprachigkeit ist polyglossia, das für XeLATEX verfügbar ist. Das Paket gibt es für LuaLATEX noch nicht.

5.3 (Noch) nicht funktionierend

Alte Encodings Pakete, die mit Eingabedateien oder Ausgabe (Schriftarten) spielen, werden sehr wahrscheinlich nicht mit LuaLATEX funktionieren. Die gute Nachricht ist, dass Unicode einen viel mächtigeren Ansatz zur Lösung von Encoding-Problemen bereitstellt, als die alten Pakete. Daher werden Sie die alten Pakete ohnehin nicht benötigen. Leider ist noch nicht alles in die glitzernde neue Welt von Unicode portiert worden und es wird noch ein reduzierte Auswahl für einige Zeit geben, was besonders auf Schriftarten zutrifft.

soul Das soul-Paket benutzt einen schlauen Trick mit einer Festbreitenschriftart, um Zeichen zu zählen. Wie dem auch sei, wegen Unterschiede im Schriftartenhandling funktioniert dies nicht mit mehr als 256 Zeichen.

Leerzeichen Leerzeichen in Dateinamen sind nicht gut unterstützt, was generell für die TEX Welt gilt.

[6] LuaLATEX 0.60 hatte einen Fehler, der die Silbentrennung nach einer ---Ligatur am Ende eines Abschnittes verhinderte.