# The nccfancyhdr package*

Alexander I. Rozhenko
rozhenko@oapmg.sscc.ru

2004/12/07

This package is originated on the `fancyhdr` package by Piet van Oostrum. It provides almost the same functionality but implements it in more safe and simple way. The most important reason for re-implementation the `fancyhdr` was that `fancy` page style breaks conventions on page styles definition: avoiding global definitions in page styling commands. If this contract is broken, a page style cannot be used locally as a parameter of the `\thispagestyle` command. Other reasons for such re-implementation were the following: some commands in `fancyhdr` do more than it is necessary (e.g. the `fancy` page style redefines section marks), incorrect vertical alignment in headers leads to raising headers a bit (this produces a page overfull if header height is exactly the same as a height of text in it), some features introduced in the `fancyhdr` are unsafe (a special cycle `\@forc` is introduced with the `\def` command), and the implementation of commands is frequently too complicated. All these disadvantages of `fancyhdr` set off me to prepare a new version of `fancyhdr` packaged named as the `nccfancyhdr`.

## Contents

---

[*] This file has version number v1.1, last revised 2004/12/07.

# 1 Using the Package

The package supports three-part headers and footers separated from the text area with optional decorative lines. Using fancy headers and footers you can easy customize page layout.

The first and the most useful benefit of fancy page styles is the possibility of decoration of headers and footers with a rule. If you want to add a rule to some of standard page styles (`empty`, `plain`, `myheadings`, and `headings`), put their names in the list of options of the `\usepackage` command:

```
\usepackage[⟨style-list⟩]{nccfancyhdr}
```

For example, the command

```
\usepackage[plain,headings]{nccfancyhdr}
```

loads the `nccfancyhdr` package and redefines the `plain` and `headings` styles on the base of `fancy` page style. It also sets the last style in the list (e.g. `headings` style) as a default page style.

# 2 Rule Control

`\headrulewidth`
`\footrulewidth`

The widths of decorative rules for header and footer are coded in the `\headrulewidth` and `\footrulewidth` commands respectively (these commands were ported from the `fancyhdr` package). The default values for these commands are `0.4pt` (standard head rule width) and `0pt` (no foot rule). To change defaults, you should redefine corresponding commands. For example, to set a head rule of `0.6pt` width in this document, we use the following command:

```
\renewcommand{\headrulewidth}{0.6pt}
```

`\headstrutheight`
`\footstrutheight`

A distance between rules and headers/footers is controlled with the `\headstrutheight` and `\footstrutheight` commands. Here is a distinction with the `fancyhdr` package. The `fancyhdr` allows control the distance between the decoration rule and the page foot only in the `\footruleskip` command. Moreover, we use another technique to provide separation between header/footer and its rule: we insert special struts in headers and footers whose height and depth are calculated using the values of the mentioned commands. The defaults for both `\headstrutheight` and `\footstrutheight` are `0.3\normalbaselineskip`. You can redefine them in just the same manner as rule width commands above.

`\headrule`
`\footrule`

The decorative rules in the header and footer are prepared with the `\headrule` and `\footrule` commands. These commands work in vertical mode. They put an `\hrule` and do a negative `\vskip` to compensate the rule height (see the implementation section for more details). You can redefine these rules to produce custom decoration lines. For example, the double line in the header of this document is produced with the following code:

```
\makeatletter
\renewcommand{\headrule}{%
  \setlength\@tempdima{\headrulewidth}%
  \hrule\@height\@tempdima\@width\headwidth
  \vskip 2\@tempdima
  \hrule\@height\@tempdima\@width\headwidth
  \vskip -4\@tempdima
}
\makeatother
```

`\headwidth`
`\normalheaders`
`\extendedheaders`

The width of header and footer (and, of course, the widths of their rules) is controlled with the `\headwidth` register. It is usually equal to the `\textwidth` but can exceed it. In the last case, the headers and footers are expanded on the marginal area. To simplify control of the `\headwidth`, two service commands are introduced in the package. The `\normalheaders` command sets the `\headwidth` to the `\textwidth`. The `\extendedheaders` enlarges the headers and footers on the whole marginal area: in two-column mode, header and footer are expanded to both margins and, in one-column mode, header and footer are expanded to the outer margin, but, if reverse margin mode is on, they are expanded to the inner margin. In this document, the `\headwidth` is expanded to marginal area as follows:

```
\addtolength{\headwidth}{\marginparsep}
\addtolength{\headwidth}{0.6\marginparwidth}
```

# 3 Page Style Customization

To customize a page style of your document, you can do the following: set the `\pagestyle{fancy}` in the preamble of the document and specify values of header and footer marks with the following commands:

| Command | Default optional parameter | Meaning |
|---|---|---|
| `\fancyhf[`⟨*pos-list*⟩`]{`⟨*mark*⟩`}` | `[lh,ch,rh,lf,cf,rf]` | Set a mark for header/footer |
| `\fancyhead[`⟨*pos-list*⟩`]{`⟨*mark*⟩`}` | `[l,c,r]` | Set a mark for header |
| `\fancyfoot[`⟨*pos-list*⟩`]{`⟨*mark*⟩`}` | `[l,c,r]` | Set a mark for footer |
| `\lhead[`⟨*even-mark*⟩`]{`⟨*odd-mark*⟩`}` | `[`⟨*odd-mark*⟩`]` | Set the left mark of header |
| `\chead[`⟨*even-mark*⟩`]{`⟨*odd-mark*⟩`}` | `[`⟨*odd-mark*⟩`]` | Set the center mark of header |
| `\rhead[`⟨*even-mark*⟩`]{`⟨*odd-mark*⟩`}` | `[`⟨*odd-mark*⟩`]` | Set the right mark of header |
| `\lfoot[`⟨*even-mark*⟩`]{`⟨*odd-mark*⟩`}` | `[`⟨*odd-mark*⟩`]` | Set the left mark of footer |
| `\cfoot[`⟨*even-mark*⟩`]{`⟨*odd-mark*⟩`}` | `[`⟨*odd-mark*⟩`]` | Set the center mark of footer |
| `\rfoot[`⟨*even-mark*⟩`]{`⟨*odd-mark*⟩`}` | `[`⟨*odd-mark*⟩`]` | Set the right mark of footer |

All these commands are ported from the `fancyhdr` package.

`\fancyhf`

The `\fancyhf` command allows specify any mark of header or footer. The ⟨*pos-list*⟩ argument specifies marks to set. A mark position selector in the ⟨*pos-list*⟩ argument consists of up to three letters: header/footer selector (`h` or `f`), horizontal position selector (`l` or `c` or `r`), odd/even page selector (`o` or `e`). The odd/even page selector is optional. If it is omitted, the command is applied to the corresponding mark on both odd and even pages. For example, `\fancyhf[hco]{mark}` sets the center mark of a header for odd pages.

*Note*: the even page selector has a sense for two-side mode only. In one-side documents (e.g. reports), even page marks are ignored.

`\fancyhead`
`\fancyfoot`

The `\fancyhead` and `\fancyfoot` commands allows specify any mark of header and footer respectively. A mark position selector in the ⟨*pos-list*⟩ argument consists of up to two letters: horizontal position selector (`l` or `c` or `r`) and odd/even page selector (`o` or `e`). The odd/even page selector is also optional. For example, `\fancyhead[l]{mark}` sets the left mark of a header for both odd and even pages.

*Note*: The command `\fancyhf{}` clears all marks of headers and footers. The `\fancyhead{}` and `\fancyfoot{}` commands clear all marks in headers and footers respectively.

`\lhead`
`\chead`
`\rhead`
`\lfoot`
`\cfoot`
`\rfoot`

We also implement the old-style macros `\lhead`, `\chead`, `\rhead`, `\lfoot`, `\cfoot`, and `\rfoot`. Their meaning is clear enough. For example, the command `\rhead[even-mark]{odd-mark}` is equivalent to the following commands:

```
\fancyhead[le]{even-mark}
\fancyhead[lo]{odd-mark}
```

If an optional parameter of these commands is omitted, the same mark is set for both odd and even pages. For example, the command `\cfoot{mark}` is equivalent to the `\fancyfoot[c]{mark}`.

`\nouppercase`

You can use the `\nouppercase{`⟨*text*⟩`}` command within a mark commands to ignore the `\uppercase` and `\MakeUppercase` commands in its parameter. For example, the `\rhead{\nouppercase{\rightmark}}` command ignores conversion the contents of `\rightmark` to uppercase.

## 4 Fancy Centering

The marks in a fancy header and footer are prepared using `\parbox` command. So, you can use multiline marks. In the header, they are aligned to the bottom line, but, in the footer, they are aligned to the top line. The maximum width of every mark is equal to the `\headwidth`. This can lead to overlapping of neighbour marks.

`\fancycenter`

If you want to prepare marks in more traditional way in a line not exceeding the `\headwidth`, you can use the following command in any mark command:

```
\fancycenter[⟨distance⟩][⟨stretch⟩]
            {⟨left-mark⟩}{⟨center-mark⟩}{⟨right-mark⟩}
```

This command works like

```
\hbox to\linewidth{{⟨left-mark⟩}\hfil{⟨center-mark⟩}\hfil{⟨right-mark⟩}}
```

but does this work more carefully trying to exactly center the central part of the text if possible. The solution for exact centering is applied if the width of ⟨*center-mark*⟩ is less than

```
\linewidth - 2*(⟨stretch⟩*⟨distance⟩ +
                  max(width(⟨left-mark⟩), width(⟨right-mark⟩))).
```

Otherwise the ⟨*center-mark*⟩ will slightly migrate to a shorter item (⟨*left-mark*⟩ or ⟨*right-mark*⟩), but at least ⟨*distance*⟩ space between all parts of line is provided. The default values of ⟨*distance*⟩ and ⟨*stretch*⟩ are `1em` and `3`.

If the ⟨*center-mark*⟩ is empty, the `\fancycenter` is equivalent to the following command:

```
\hbox to\linewidth {{⟨left-mark⟩}\hfil {⟨right-mark⟩}}
```

*Note*: The usage of `\fancycenter` command is not limited with the argument of header/footer marks only. You can use it anywhere in your document.

# 5 Prepare Custom Page Styles

`\newpagestyle` In the `nccfancyhdr` package, we recommend to set fancy marks within definition of a custom page style. In this case, you can easy select a custom style with the `\pagestyle` or `\thispagestyle` command. To support this, the `\newpagestyle` command is introduced:

```
\newpagestyle{⟨style-name⟩}[⟨base-style⟩]{⟨definitions⟩}
```

It is allowed in the preamble only. The ⟨*base-style*⟩ is a style the new style will be based on. If the optional parameter is omitted, the `fancy` style is used as the base style. The `fancy` style works as the `empty` style, but support decorative rules and extended headers/footers and allows fancy marks. A desired page style works as follows: at the first, the base style is applied; after that, the ⟨*definitions*⟩ customize the base style.

*Note*: You can use any existing ⟨*base-style*⟩ in the definition of a new style, but, if you apply fancy mark commands in the ⟨*definitions*⟩ parameter, the base style should be based on the `fancy` style.

For example, all pages of this document except the first one were prepared with the custom page style as follows:

```
\usepackage[headings]{nccfancyhdr}
\newpagestyle{lheadings}[headings]{%
  \fancyhead[ce]{\nouppercase{%
    \fancycenter{\thepage}{}{\slshape\leftmark}}}%
  \fancyhead[co]{\nouppercase{%
    \fancycenter{\slshape\rightmark}{}{\thepage}}}%
}
\pagestyle{lheadings}
```

As you can see from here, the fancy versions of `headings` and `myheadings` styles use the center mark only and prepare it with the help of the `\fancycenter` command.

## 6 How to Change a Page Style in Floatpages?

A floatpage is a page consisting of floats only. You cannot directly change a page style for such a page, because it is prepared in whole in the LaTeX Output Routine. We recommend to change a page style for floatpages with the help of the `afterpage` package. Just add a command `\usepackage{afterpage}` in the preamble and put the command:

`\afterpage{\thispagestyle{`⟨*special-style*⟩`}}`

anywhere in the page going before the floatpage. The ⟨*special-style*⟩ is a style you want to apply for floatpages.

`\iffloatpage`
`\iftopfloat`
`\ifbotfloat`

Another way for change a page style on pages with floats consists in using the following conditional commands within marks of a page style:

`\iffloatpage{`⟨*true-clause*⟩`}{`⟨*false-clause*⟩`}`
`\iftopfloat{`⟨*true-clause*⟩`}{`⟨*false-clause*⟩`}`
`\ifbotfloat{`⟨*true-clause*⟩`}{`⟨*false-clause*⟩`}`

These commands were ported from the `fancyhdr` package. The `\iffloatpage` command executes the ⟨*true-clause*⟩ if this is a floatpage. Otherwise, it executes the ⟨*false-clause*⟩. Analogously, the `\iftopfloat` and `\ifbotfloat` test the lists of top floats and bottom floats of the page to be nonempty.

Whereas these commands are rare used, they are defined if the package is loaded with the `testfloats` option.

## 7 Package Options

In conclusion, we enumerate all package options available:

| Option | Meaning |
|---|---|
| `empty` | redefine the `empty` page style to be fancy-based style |
| `plain` | redefine the `plain` page style to be fancy-based style |
| `headings` | redefine the `headings` page style to be fancy-based style |
| `myheadings` | redefine the `myheadings` page style to be fancy-based style |
| `testfloats` | define `\iffloatpage`, `\iftopfloat`, and `\ifbotfloat` commands |

The options are executed in the order they are specified in the list of options. Every page style redefinition option sets a redefined style to be the current page style. Therefore, after loading of this package, the style redefined in the last order will be the current page style.

## 8 The Implementation

`\newpagestyle`

We start with the `\newpagestyle` command. It was introduced in the version 1.1 of the package. It is available in the preamble only.

```
⟨*package⟩
\newcommand*{\newpagestyle}[1]{%
  \@ifnextchar[{\NCC@newpagestyle{#1}}{\NCC@newpagestyle{#1}[fancy]}%
}
\long\def\NCC@newpagestyle#1[#2]#3{%
  \@ifundefined{ps@#2}{%
    \PackageError{nccfancyhdr}
      {\string\newpagestyle: Unknown base page style ‘#2’}{}%
  }{}%
  \edef\@tempa{\noexpand\newcommand \expandafter\noexpand
    \csname ps@#1\endcsname}%
  \expandafter\@tempa\expandafter{\csname ps@#2\endcsname #3}%
}
\@onlypreamble\newpagestyle
\@onlypreamble\NCC@newpagestyle
```

`\fancyhf`
`\fancyhead`
`\fancyfoot`

Now we define the new-style fancy marking commands. They are based on the `\NCC@fancyhf` command.

```
\newcommand*{\fancyhf}[1][lh,ch,rh,lf,cf,rf]{\NCC@fancyhf{}{#1}}
\newcommand*{\fancyhead}[1][l,c,r]{\NCC@fancyhf h{#1}}
\newcommand*{\fancyfoot}[1][l,c,r]{\NCC@fancyhf f{#1}}
```

`\lhead`
`\chead`
`\rhead`
`\lfoot`
`\cfoot`
`\rfoot`

The old-style fancy-marking commands are based on the `\NCC@fancy` command.

```
\newcommand{\lhead}{\@dblarg{\NCC@fancy{lh}}}
\newcommand{\chead}{\@dblarg{\NCC@fancy{ch}}}
\newcommand{\rhead}{\@dblarg{\NCC@fancy{rh}}}
\newcommand{\lfoot}{\@dblarg{\NCC@fancy{lf}}}
\newcommand{\cfoot}{\@dblarg{\NCC@fancy{cf}}}
\newcommand{\rfoot}{\@dblarg{\NCC@fancy{rf}}}
```

`\NCC@fancy`

The `\NCC@fancy{`⟨*selector*⟩`}[`⟨*even-mark*⟩`]{`⟨*odd-mark*⟩`}` command sets a pair of marks. A ⟨*selector*⟩ consists of two letters: (lcr) and (hf). We need not test the ⟨*selector*⟩ on correctness because this command is applied internally.

```
\def\NCC@fancy#1[#2]#3{
  \expandafter\def\csname NCC@f@e#1\endcsname{#2}%
  \expandafter\def\csname NCC@f@o#1\endcsname{#3}%
}
```

`\NCC@fancyhf`

The `\NCC@fancyhf{`⟨*hf*⟩`}{`⟨*pos-list*⟩`}{`⟨*mark*⟩`}` command parses the ⟨*pos-list*⟩ by selectors and executes the `\NCC@fancydef` for every selector. The ⟨*hf*⟩ argument contains the common part of all selectors added at their beginning.

```
\def\NCC@fancyhf#1#2#3{%
  \@for\@tempa:=#2\do
    {\edef\@tempa{\noexpand\NCC@fancydef{#1\@tempa}}\@tempa{#3}}%
}
```

`\NCC@fancydef`

The `\NCC@fancydef{`⟨*selector*⟩`}{`⟨*mark*⟩`}` command analyzes the ⟨*selector*⟩ and defines corresponding fancy mark. It uses the `\NCC@fancyclass` command that prevents using many letters of the same class in the ⟨*selector*⟩. For example, it will be an error if more that one ‘l’ or ‘c’ or ‘r’ letter appears in the selector.

```
\def\NCC@fancydef#1#2{%
```

The `\NCC@hf`, `\NCC@lcr`, and `\NCC@oe` will contain a letter of the corresponding class found in selector. Before the cycle, they are set to `\relax`.

```
  \let\NCC@hf\relax \let\NCC@lcr\relax \let\NCC@oe\relax
  \@tfor \@nextchar:=#1\do
```

Prepare in `\@tempa` a next letter in uppercase.

```
    {\edef\@tempa{\noexpand\uppercase{\noexpand\def%
       \noexpand\@tempa{\@nextchar}}}\@tempa
```

Test the letter and specify corresponding class.

```
     \if\@tempa H\NCC@fancyclass\NCC@hf{h}{#1}\else
      \if\@tempa F\NCC@fancyclass\NCC@hf{f}{#1}\else
       \if\@tempa L\NCC@fancyclass\NCC@lcr{l}{#1}\else
        \if\@tempa C\NCC@fancyclass\NCC@lcr{c}{#1}\else
         \if\@tempa R\NCC@fancyclass\NCC@lcr{r}{#1}\else
          \if\@tempa O\NCC@fancyclass\NCC@oe{o}{#1}\else
           \if\@tempa E\NCC@fancyclass\NCC@oe{e}{#1}\else
             \NCC@fancyerror{Illegal char ‘\@nextchar’ in argument ‘#1’}%
           \fi
          \fi
         \fi
        \fi
       \fi
      \fi
     \fi
    }%
```

After cycle, we test that the `\NCC@hf` and `\NCC@lcr` classes are specified. The `\NCC@oe` class is optional. So, we do not test it. Finally, we define appropriate mark commands.

```
  \ifx\NCC@hf\relax \NCC@fancyerror{No ‘h’ or ‘f’ specified}\else
   \ifx\NCC@lcr\relax \NCC@fancyerror{No ‘l’ or ‘c’ or ‘r’ specified}\else
    \ifx\NCC@oe\relax
      \expandafter\def\csname NCC@f@o\NCC@lcr\NCC@hf\endcsname{#2}%
      \expandafter\def\csname NCC@f@e\NCC@lcr\NCC@hf\endcsname{#2}%
    \else
      \expandafter\def\csname NCC@f@\NCC@oe\NCC@lcr\NCC@hf\endcsname{#2}%
    \fi
   \fi
  \fi
}
```

`\NCC@fancyclass`

The `\NCC@fancyclass{`⟨*command*⟩`}{`⟨*letter*⟩`}{`⟨*selector*⟩`}` command tests the ⟨*command*⟩ to be `\relax` and defines it with the ⟨*letter*⟩ argument. If the command is already defined, the error message is generated.

```
\def\NCC@fancyclass#1#2#3{%
```

```
  \ifx#1\relax
    \def#1{#2}%
  \else
    \NCC@fancyerror{Misusing the char `\@nextchar' in argument `#3'}%
  \fi
}
```

`\NCC@fancyerror`

A handler of errors in fancy mark definitions.

```
\def\NCC@fancyerror#1{%
  \PackageError{nccfancyhdr}%
    {Fancy mark definitions:\MessageBreak#1}%
}
```

`\headwidth`
`\extendedheaders`
`\normalheaders`

Now we allocate the `\headwidth` register and define its control commands.

```
\newdimen\headwidth
\newcommand{\extendedheaders}{
  \@tempdima\marginparwidth \advance\@tempdima\marginparsep
  \@tempdimb\textwidth \advance\@tempdimb\@tempdima
  \if@twocolumn \advance\@tempdimb\@tempdima \fi
  \global\headwidth\@tempdimb
}
\newcommand{\normalheaders}{\global\headwidth\textwidth}
```

`\headrulewidth`
`\footrulewidth`
`\headstrutheight`
`\footstrutheight`

Now we specify parameters of decoration rules: widths and struts.

```
\newcommand{\headrulewidth}{.4\p@}
\newcommand{\footrulewidth}{\z@}
\newcommand{\headstrutheight}{.3\normalbaselineskip}
\newcommand{\footstrutheight}{.3\normalbaselineskip}
```

*Note*: Really, the head strut height is zero but its depth is equal to the value of `\headstrutheight`. Moreover, the foot strut height is a sum of `0.55\normalbaseskip` and the value of `\footstrutheight`. But we prefer the universal notations for command names instead of strict one, because users do not interested in implementation details.

`\headrule`

The default implementation of the `\headrule`. It works in vertical mode. At first it draws a rule and then it inserts a negative skip for compensation.

```
\newcommand{\headrule}{%
  \setlength\@tempdima{\headrulewidth}%
  \hrule\@height\@tempdima\@width\headwidth
  \vskip-\@tempdima
}
```

`\footrule`

The `\footrule` works in reverse order: at first it inserts a negative skip and after that it draws a rule.

```
\newcommand{\footrule}{%
  \setlength\@tempdima{\footrulewidth}% Can use calc here
  \vskip -\@tempdima
  \hrule \@height\@tempdima \@width\headwidth
}
```

`\NCC@fancyreset` The `\NCC@fancyreset` command is used at the beginning of fancy headers and footers. It resets font, removes baseline stretch and locally defines the `\nouppercase` command. In comparison with the `fancyhdr` package, we do not call the `\restorecr` command because it is obsolete now. We also redefine the `\uppercase` and `\MakeUppercase` commands in more appropriate way than in `fancyhdr`.

```
\def\NCC@fancyreset{\let\baselinestretch\@empty
  \long\def\nouppercase##1{%
    \begingroup
      \long\def\uppercase####1{####1}%
      \long\def\MakeUppercase####1{####1}%
      ##1%
    \endgroup
  }%
  \reset@font
}
```

`\NCC@fancyhead` The `\NCC@fancyhead{`⟨*left-mark*⟩`}{`⟨*center-mark*⟩`}{`⟨*right-mark*⟩`}` command prepares the fancy header. It differs from the implementation in the `fancyhdr` at the following issue: the vertical box in this command (`\@tempboxa`) is prepared as `\vtop` box, but in the `fancyhdr` package it is prepared as `\vbox` box. As a consequence, in the `fancyhdr` version, the base line of the vertical box goes at the rule and the header slightly moves up.

```
\def\NCC@fancyhead#1#2#3{%
  \hb@xt@\headwidth{\NCC@fancyreset
    \setbox\@tempboxa\vtop{%
      \hbox{%
```

Prepare the left mark:

```
        \rlap{\parbox[b]\headwidth{\raggedright#1}}%
```

Insert the strut:

```
        \setlength\@tempdima{\headstrutheight}%
        \vrule\@width\z@\@height\z@\@depth\@tempdima
```

Prepare the center mark:

```
        \parbox[b]\headwidth{\centering#2}%
```

Prepare the right mark:

```
        \llap{\parbox[b]\headwidth{\raggedleft#3}}%
      }%
```

Draw decoration rule:

```
      \headrule
    }%
```

Compare the height of `\@tempboxa` with the `\headheaght` and correct the last one if vertical overfull appears:

```
    \NCC@fancytest\headheight
```

Put the fancy header:

```
    \box\@tempboxa
  }%
}
```

`\NCC@fancyfoot`

The `\NCC@fancyfoot{`⟨*left-mark*⟩`}{`⟨*center-mark*⟩`}{`⟨*right-mark*⟩`}` command prepares the fancy footer. Its implementation is similar to the `\NCC@fancyhead`.

```
\def\NCC@fancyfoot#1#2#3{%
  \hb@xt@\headwidth{\NCC@fancyreset
    \setbox\@tempboxa\vbox{%
      \footrule
      \hbox{%
        \rlap{\parbox[t]\headwidth{\raggedright#1}}%
        \@tempdima .55\normalbaselineskip
        \addtolength\@tempdima{\footstrutheight}%
        \vrule\@width\z@\@height\@tempdima\@depth\z@
        \parbox[t]\headwidth{\centering#2}%
        \llap{\parbox[t]\headwidth{\raggedleft#3}}%
      }%
    }%
    \NCC@fancytest\footskip
    \box\@tempboxa
  }%
}
```

`\NCC@fancytest`

The `\NCC@fancytest{`⟨*register*⟩`}` command compares a value of the ⟨*register*⟩ with the height of `\@tempboxa` and modifies the ⟨*register*⟩ value if necessary.

```
\def\NCC@fancytest#1{%
  \ifdim\ht\@tempboxa>#1%
    \PackageWarning{nccfancyhdr}%
      {\string#1 is too small (\the#1):\MessageBreak
       Make it at least \the\ht\@tempboxa.\MessageBreak
       We now enlarge it for the rest of the document.\MessageBreak
       This may cause the page layout to be inconsistent, however}%
    \@tempdima#1\global\setlength{#1}{\ht\@tempboxa}%
    \ht\@tempboxa\@tempdima
  \fi
}
```

`\NCC@ihss`
`\NCC@ohss`

The `\NCC@ihss` and `\NCC@ohss` hooks insert the `\hss` command at the outer and/or inner sides of header/footer to provide the proper enlargement it on margins.

```
\def\NCC@ihss{\if@twocolumn\hss\else\if@reversemargin\hss\fi\fi}
\def\NCC@ohss{\if@twocolumn\hss\else\if@reversemargin\else\hss\fi\fi}
```

`\fancycenter`

`\fancycenter[`⟨*dist*⟩`][`⟨*stretch*⟩`]{`⟨*left-mark*⟩`}{`⟨*center-mark*⟩`}{`⟨*right-mark*⟩`}`

```
\newcommand*{\fancycenter}[1][1em]{%
  \@ifnextchar[{\NCC@fancycenter{#1}}{\NCC@fancycenter{#1}[3]}%
}
```

```
\def\NCC@fancycenter#1[#2]#3#4#5{%
```

At first, we execute the case when the ⟨*center-mark*⟩ is empty:

```
  \def\@tempa{#4}\ifx\@tempa\@empty
    \hb@xt@\linewidth{\color@begingroup{#3}\hfil {#5}\color@endgroup}%
  \else
```

All that we need to do is to calculate skips inserted before and after ⟨*center-mark*⟩. We will calculate them in the `\@tempskipa` and `\@tempskipb`. At first:

```
\@tempdima:=⟨dist⟩;
\@tempdimb:=⟨dist⟩*⟨stretch⟩;
\@tempdimc:=⟨dist⟩*⟨stretch⟩-⟨dist⟩;
\@tempskipa:=\@tempskipb:=\@tempdimb + 1fil - \@tempdimc;
```

```
    \setlength\@tempdima{#1}%
    \setlength{\@tempdimb}{#2\@tempdima}%
    \@tempdimc \@tempdimb \advance\@tempdimc -\@tempdima
    \setlength\@tempskipa{\@tempdimb \@plus 1fil \@minus \@tempdimc}%
    \@tempskipb\@tempskipa
```

At this point, the `\@tempskipa` and `\@tempskipb` registers have the natural size ⟨*dist*⟩`*`⟨*stretch*⟩, unlimited stretchability, and the minimum size ⟨*dist*⟩. Now we decrease the minimum size of `\@tempskipa` to zero if the ⟨*left-mark*⟩ is empty:

```
    \def\@tempa{#3}\ifx\@tempa\@empty
      \addtolength\@tempskipa{\z@ \@minus \@tempdima}%
    \fi
```

Do the same things with the `\@tempskipb` register if the ⟨*right-mark*⟩ is empty:

```
    \def\@tempa{#5}\ifx\@tempa\@empty % empty right
      \addtolength\@tempskipb{\z@ \@minus \@tempdima}%
    \fi
```

Finally, we correct the left and right glues taking into account the difference between lengthes of ⟨*left-mark*⟩ and ⟨*right-mark*⟩. We calculate what mark is shorter and increase the natural size of corresponding register on the difference between their lengthes.

```
    \settowidth{\@tempdimb}{#3}%
    \settowidth{\@tempdimc}{#5}%
    \ifdim\@tempdimb>\@tempdimc
      \advance\@tempdimb -\@tempdimc
      \addtolength\@tempskipb{\@tempdimb \@minus \@tempdimb}%
    \else
      \advance\@tempdimc -\@tempdimb
      \addtolength\@tempskipa{\@tempdimc \@minus \@tempdimc}%
    \fi
```

The `\@tempskipa` and `\@tempskipb` are calculated. Put the box.

```
    \hb@xt@\linewidth{\color@begingroup{#3}\hskip \@tempskipa
                      {#4}\hskip \@tempskipb {#5}\color@endgroup}%
  \fi
}
```

The rest of the package consists of games with styles and options.

`\ps@fancy` We start from declaring the `fancy` page style. It extends the `empty` page style. So, we simply call the empty style and then redefine `\@oddhead`, `\@evenhead`, `\@oddfoot`, and `\@evenfoot` to be fancy one. The `\NCC@ihss` and `\NCC@ohss` hooks provide proper enlargement of headers/footers on margins. The `\fancyhf{}` command at the end of macro clears all marks.

```
\def\ps@fancy{\ps@empty
  \def\@oddhead{%
    \NCC@ihss \NCC@fancyhead\NCC@f@olh\NCC@f@och\NCC@f@orh \NCC@ohss}%
  \def\@evenhead{%
    \NCC@ohss \NCC@fancyhead\NCC@f@elh\NCC@f@ech\NCC@f@erh \NCC@ihss}%
  \def\@oddfoot{%
    \NCC@ihss \NCC@fancyfoot\NCC@f@olf\NCC@f@ocf\NCC@f@orf \NCC@ohss}%
  \def\@evenfoot{%
    \NCC@ohss \NCC@fancyfoot\NCC@f@elf\NCC@f@ecf\NCC@f@erf \NCC@ihss}%
  \fancyhf{}%
}
```

Standard styles are redefined optionally.

`\ps@empty` When we redefine the `empty` style, we must take into account that it can be also redefined (as in `amsart` and `amsbook` classes). So, we save its previous meaning in the `\NCC@psempty` macro and base the `empty` style on the saved style.

```
\DeclareOption{empty}{%
  \let\NCC@psempty\ps@empty
  \def\ps@empty{\NCC@psempty
    \def\@oddhead{%
      \NCC@ihss \NCC@fancyhead\NCC@f@olh\NCC@f@och\NCC@f@orh \NCC@ohss}%
    \def\@evenhead{%
      \NCC@ohss \NCC@fancyhead\NCC@f@elh\NCC@f@ech\NCC@f@erh \NCC@ihss}%
    \def\@oddfoot{%
      \NCC@ihss \NCC@fancyfoot\NCC@f@olf\NCC@f@ocf\NCC@f@orf \NCC@ohss}%
    \def\@evenfoot{%
      \NCC@ohss \NCC@fancyfoot\NCC@f@elf\NCC@f@ecf\NCC@f@erf \NCC@ihss}%
    \fancyhf{}%
  }%
  \pagestyle{empty}%
}
```

`\ps@plain` The redefinition of the `plain` style is quite simple:

```
\DeclareOption{plain}{%
  \def\ps@plain{\ps@fancy \let\@mkboth\@gobbletwo
    \fancyfoot[c]{\thepage}%
  }%
  \pagestyle{plain}%
}
```

`\ps@myheadings` The redefinition of the `myheadings` style is conditional. We test the `\chapter` command on existence and redefine the style in a bit different ways.

```
\DeclareOption{myheadings}{%
  \@ifundefined{chapter}{%
    \def\ps@myheadings{\ps@fancy \let\@mkboth\@gobbletwo
      \fancyhead[ce]{\fancycenter{\thepage}{}{\slshape\leftmark}}%
      \fancyhead[co]{\fancycenter{\slshape\rightmark}{}{\thepage}}%
      \let\sectionmark\@gobble
      \let\subsectionmark\@gobble
    }%
  }{\def\ps@myheadings{\ps@fancy \let\@mkboth\@gobbletwo
      \fancyhead[ce]{\fancycenter{\thepage}{}{\slshape\leftmark}}%
      \fancyhead[co]{\fancycenter{\slshape\rightmark}{}{\thepage}}%
      \let\chaptermark\@gobble
      \let\sectionmark\@gobble
    }%
  }%
  \pagestyle{myheadings}%
}
```

`\ps@headings` The redefinition of the `headings` style also differs for book-like and article-like classes. It also differs for one-side and two-side modes.

```
\DeclareOption{headings}{%
  \@ifundefined{chapter}{%
    \if@twoside
```

An article in two-side mode:

```
      \def\ps@headings{\ps@fancy \let\@mkboth\markboth
        \fancyhead[ce]{\fancycenter{\thepage}{}{\slshape\leftmark}}%
        \fancyhead[co]{\fancycenter{\slshape\rightmark}{}{\thepage}}%
        \def\sectionmark##1{%
          \markboth{\MakeUppercase{%
            \ifnum \c@secnumdepth >\z@ \thesection\quad \fi##1}}{}}%
        \def\subsectionmark##1{%
          \markright{%
            \ifnum \c@secnumdepth >\@ne \thesubsection\quad \fi##1}}%
      }%
    \else
```

An article in one-side mode:

```
      \def\ps@headings{\ps@fancy \let\@mkboth\markboth
        \fancyhead[ce]{\fancycenter{\thepage}{}{\slshape\leftmark}}%
        \fancyhead[co]{\fancycenter{\slshape\rightmark}{}{\thepage}}%
        \def\sectionmark##1{%
          \markright {\MakeUppercase{%
            \ifnum \c@secnumdepth >\z@ \thesection\quad \fi##1}}}%
        \let\subsectionmark\@gobble % Not needed but inserted for safety
      }%
    \fi
  }{\if@twoside
```

A book in two-side mode:

```
   \def\ps@headings{\ps@fancy \let\@mkboth\markboth
    \fancyhead[ce]{\fancycenter{\thepage}{}{\slshape\leftmark}}%
    \fancyhead[co]{\fancycenter{\slshape\rightmark}{}{\thepage}}%
    \def\chaptermark##1{%
     \markboth{\MakeUppercase{%
      \ifnum \c@secnumdepth >\m@ne \if@mainmatter
       \@chapapp\ \thechapter. \ \fi\fi##1}}{}}%
    \def\sectionmark##1{%
     \markright {\MakeUppercase{%
      \ifnum \c@secnumdepth >\z@ \thesection. \ \fi##1}}}%
   }%
  \else
```

A book in one-side mode:

```
   \def\ps@headings{\ps@fancy \let\@mkboth\markboth
    \fancyhead[ce]{\fancycenter{\thepage}{}{\slshape\leftmark}}%
    \fancyhead[co]{\fancycenter{\slshape\rightmark}{}{\thepage}}%
    \def\chaptermark##1{%
     \markright{\MakeUppercase{%
      \ifnum \c@secnumdepth >\m@ne \if@mainmatter
       \@chapapp\ \thechapter. \ \fi\fi##1}}}%
    \let\sectionmark\@gobble % Not needed but inserted for safety
   }%
  \fi
 }%
 \pagestyle{headings}%
}
```

`\iffloatpage`
`\iftopfloat`
`\ifbotfloat`

These macros are defined in the `testfloats` option. They were ported from the `fancyhdr` and modified a bit in more LaTeX way. It is no warrantee that these macros will proper work in all cases. They must be used inside fancy mark commands.

```
\DeclareOption{testfloats}{%
 \let\NCC@fancymakecol\@makecol
 \let\NCC@fancytoplist\@empty
 \let\NCC@fancybotlist\@empty
 \def\@makecol{%
  \let\NCC@fancytoplist\@toplist
  \let\NCC@fancybotlist\@botlist
  \NCC@fancymakecol
 }%
 \newcommand\iftopfloat{%
  \ifx\NCC@fancytoplist\@empty
   \expandafter\@secondoftwo
  \else
   \expandafter\@firstoftwo
  \fi
 }%
 \newcommand\ifbotfloat{%
```

```
    \ifx\NCC@fancybotlist\@empty
      \expandafter\@secondoftwo
    \else
      \expandafter\@firstoftwo
    \fi
  }%
  \newcommand\iffloatpage{%
    \if@fcolmade
      \expandafter\@firstoftwo
    \else
      \expandafter\@secondoftwo
    \fi
  }%
}
```

Finally, we process the options in the order they are specified in the options list and set the defaults.

```
\ProcessOptions*
\normalheaders
\fancyhf{}
⟨/package⟩
```