The fancyhdr and extramarks packages

version v4.0.1.

Pieter van Oostrum[*]
Dept. of Computer Science[†]
Utrecht University

January 28, 2021

Abstract

This document describes how to customize the page layout of your LaTeX documents, i.e how to change page margings and sizes, headers and footers, and the proper placement of figures and tables (collectively called floats) on the page.

This documentation describes version 4.0 or later of the fancyhdr and extramarks packages. The user documentation is valid for the versions 3.8 or later of the fancyhdr package (except for the changes mentioned in section 32.1), and version 2.1 or later of the extramarks package.

Contents

[*]A considerable part of this documentation was written by George Grätzer (University of Manitoba) in *Notices Amer. Math. Soc.* Thanks, George!

[†]This was my employer at the time I developed this package. I am now retired.

Part I

Introduction

This document contains four parts:

Part I is a short documentation on the user commands of the fancyhdr and extramarks packages.

Part II contains elaborate documentation on page layout in LaTeX. This used to be the complete documentation of fancyhdr and extramarks for several years.

Part III contains Questions and Answers.

Part IV contains the annotated implementation.

This document describes version 4 of fancyhdr. This version is also described in the forthcoming edition 3 of *The LaTeX Companion*, whereas the previous editions describe version 3 of fancyhdr. There are some significant differences between these versions. These are summarized in section 32.1 on page 47. Throughout this documentation it is mentioned when a specific feature is only available in version 4, or when there are differences between version 3 and 4.

1 Installation

The preferred way to install this package is with a package installer. If you want to install it by hand, then first run the command `tex fancyhdr.ins` and then move the files `fancyhdr.sty`, `extramarks.sty` and `fancyheadings.sty` to a place where LaTeX can find it, preferably in a directory similar to `.../texmf/tex/latex/fancyhdr/` in your TeX directory tree.

2 Using fancyhdr

The package fancyhdr gives you several commands to define headers and footers of the pages in a LaTeX document. You load the package with the following command in the preamble:

`\usepackage[⟨options⟩]{fancyhdr}`

(Options are available since version 4.0) The following options are supported:

Option	Meaning
`nocheck`	do not check the heights of the header and footer (see section 17 on page 22)
`compatV3`	keep some behaviour (now considered undesirable) as in version 3 (see section 13 and section 17 on page 22)
`headings`	redefine the `headings` page style to be fancy-based
`myheadings`	redefine the `myheadings` page style to be fancy-based

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

```
\fancyhead[places]{field}
\fancyfoot[places]{field}
\fancyhf[places]{field}
```

Here `places` is a comma-separated list of places where `field` will be placed. There are 12 places defined: Left, Center and Right Headers and Footers, and both can be on Even or Odd pages. Each place therefore has 3 coordinates which are the inital letters of the above description: (1) `E` or `O`, (2) `L`, `C` or `R`, (3) `H` or `F`. So a place is given with 3 letters, like `EOH`. A missing coordinate means: all possibilities, except for `\fancyhead` where `H` is implied and `\fancyfoot` where `F` is implied.

`\fancyheadoffset`
`\fancyfootoffset`
`\fancyhfoffset`

```
\fancyheadoffset[places]{field}
\fancyfootoffset[places]{field}
\fancyhfoffset[places]{field}
```

These define offsets to let the headers stick into the margin (or to the inside if negative). Places cannot contain the `C` specifier. See section 18 for more details.

`\headrulewidth` `\footrulewidth` `\headruleskip` `\footruleskip` `\headrule` `\footrule` `\headwidth`

`\headrulewidth` and `\footrulewidth` are macros to define the thickness of a line under the header and above the footer. `\headruleskip` and `\footruleskip` are macros that define the distance between the lines and the header and footer text, respectively. (But `\headruleskip` is only available since version 4.0.) `\headrule` and `\footrule` are macros to completely redefine these lines. And `\headwidth` is a length parameter that defines the total width of the headers and footers. See section 18 for more details.

`\fancyheadinit` `\fancyfootinit` `\fancyhfinit`

`\fancyheadinit` and `\fancyfootinit` can be used to define initialisation code for the header and footer, respectively, and `\fancyhfinit` defines both of these. These commands are only available in fancyhdr version 4.0 and later. See section 24.

`\fancycenter`

(Only in version 4.0 and later.) The command `\fancycenter` packs 3 header fields into a full-width header. See section 9.

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

The macros `\iftopfloat`, `\ifbotfloat`, `\iffloatpage` and `\iffootnote` are used to detect if there is a float on the top or the bottom of the page, or the page is a float page, or if there is a footnote at the bottom of the page. These can be used to choose different headers and/or footers if these conditions are met. See section 19 for more details.

`\fancypagestyle`

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

This command lets you (re)define page styles for use in special situations. See section 12 for more details.

3 Using extramarks

The extramarks gives you some extra marks in LaTeX, besides the normal `\leftmark` and `\rightmark`, that are defined by the `\markboth` and `\markright` commands.

`\firstleftmark` `\lastrightmark` `\firstrightmark` `\lastleftmark`

```
\firstleftmark
\lastrightmark
\firstrightmark
\lastleftmark
```

Standard LaTeX has two marks: a left and a right one. The standard command `\leftmark` gives you the last left mark on a page, and `\rightmark` gives you the first right one. These macros give you also the other combinations, where `\firstrightmark` = `\rightmark` and `\lastleftmark` = `\leftmark`. As with the standard marks, these are meant to be used in headers and footers. In other places they will not work properly.

```
\extramarks{aa}{bb}
\firstleftxmark
\firstrightxmark
\topleftxmark
\toprightxmark
\lastleftxmark
\lastrightxmark
\firstxmark
\lastxmark
\topxmark
```

The command `\extramarks{aa}{bb}` defines two extra marks, similar to the standard ones by LaTeX, where `aa` is the left one and `bb` is the right one. The other commands are to extract these in the headers and footers, similar to the ones without the `x`. See sections 15 and 25 for more details.

Part II
Page Layout in LaTeX

4 Introduction

A page in a LaTeX document is built from various elements as shown in figure 1. The body contains the main text of the document together with the so called floats (tables and figures).

The pages are constructed by LaTeX's output routine, which is quite complicated and should therefore not be modified. Some of the packages described in this paper contains small modifications to the output routine to accomplish things that cannot be done in another way. You should use these packages to get the desired result rather than fiddling with the output routine yourself.

There are a number of things that you must be aware of:

1. The margins on the left are not called `\leftmargin`, but `\evensidemargin` (on even-numbered pages) and `\oddsidemargin` (on odd-numbered pages). In one-sided documents `\oddsidemargin` is used for either. `\leftmargin` is also a valid LaTeX parameter but it has a different use (namely the indentation of lists).

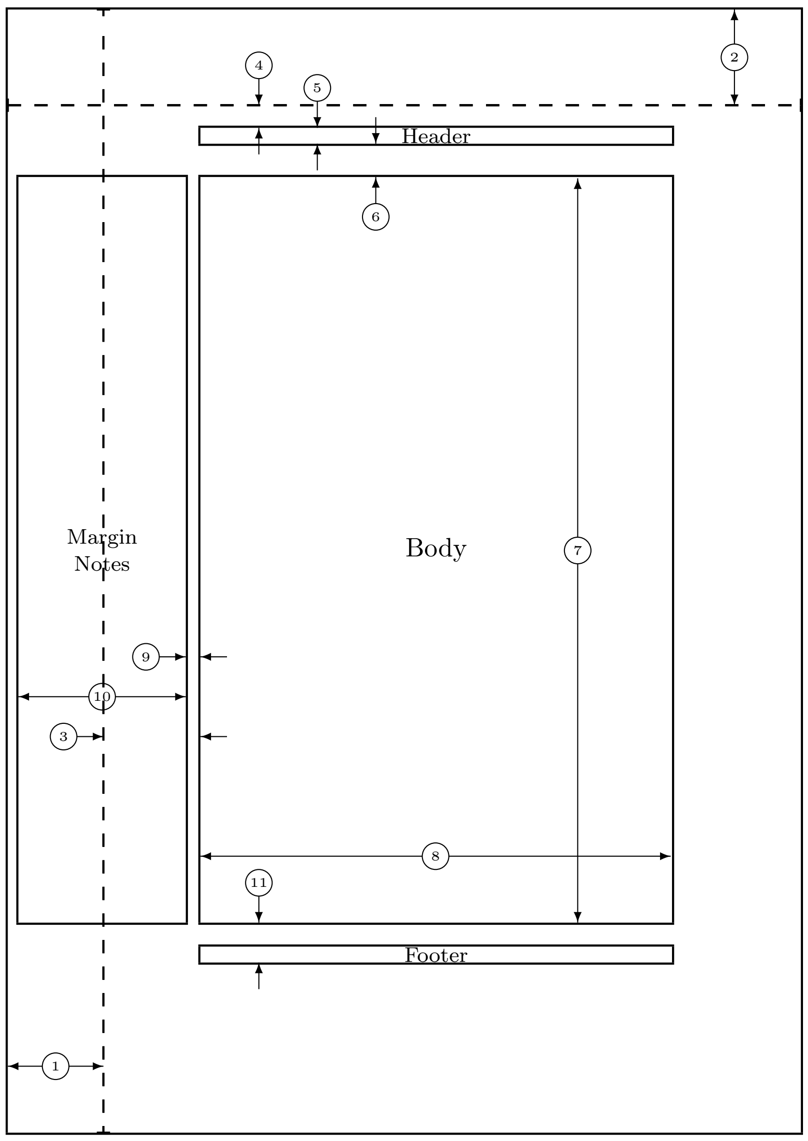
2. Most of the parameters should not be changed in the middle of a document. Some changes might work at a pagebreak. If you want to change the height of a single page, you can use the `\enlargethispage` command.

The margin notes area contains small pieces of information created by the `\marginpar` command. On twosided documents the margin notes appear on the left and right alternatively. The margin notes are not on fixed places with respect to the paper but at approximately the same height as the paragraph in which they appear. Due to the algorithm used to decide the placement of margin notes, in a twosided document unfortunately they may appear on the wrong side if they are close to a page break. If you want to put information on fixed places in the margins you may use the technique described in sections 26 and 27.

The first part of this paper describes how to change the header and footer areas. The last part describes how to get your floats at the desired place.

5 Page headers and footers

The page headers and footers in LaTeX are defined by the `\pagestyle` and `\pagenumbering` commands. `\pagestyle` defines the general contents of the headers and footers (e.g. where the page number will be printed), while `\pagenumbering` defines the format of the page number. LaTeX has four standard page styles:



1	`one inch + \hoffset`	2	`one inch + \voffset`
3	`\oddsidemargin = 73pt`	4	`\topmargin = 17pt`
5	`\headheight = 12pt`	6	`\headsep = 25pt`
7	`\textheight = 561pt`	8	`\textwidth = 355pt`
9	`\marginparsep = 11pt`	10	`\marginparwidth = 126pt`
11	`\footskip = 30pt`		`\marginparpush = 0pt (not shown)`
	`\hoffset = 0pt`		`\voffset = 0pt`
	`\paperwidth = 597pt`		`\paperheight = 845pt`

Figure 1: Page elements. The values shown are those in effect in the current document, not the defaults.

`empty`	no headers or footers
`plain`	no header, footer contains page number centered
`headings`	no footer, header contains name of chapter/section and/or subsection and page number
`myheadings`	no footer, header contains page number and user supplied information

Although these are useful styles, they are quite limited. Additional page styles can be defined by defining commands of the form `\ps@xxx`. This command is executed when a `\pagestyle{xxx}` is given in the document. The `\ps@xxx` command should define the following commands for the contents of the headers and footers:

`\@oddhead`	header on odd numbered pages in two-sided documents (on all pages in one-sided)
`\@evenhead`	header on even numbered pages in two-sided documents
`\@oddfoot`	footer on odd numbered pages in two-sided documents (on all pages in one-sided)
`\@evenfoot`	footer on even numbered pages in two-sided documents

These are not user commands, but rather "variables" that are used by LaTeX's output routine. As the command names contain the character '`@`', they should be defined in a package file, or otherwise be sandwiched between the commands `\makeatletter` and `\makeatother`.

The `\pagenumbering` command defines the layout of the page number. It has a parameter from the following list:

`arabic`	arabic numerals
`roman`	lower case roman numerals
`Roman`	upper case roman numerals
`alph`	lower case letter
`Alph`	upper case letter

The `\pagenumbering{xxx}` defines the command `\thepage` to be the expansion of the page number in the given notation `xxx`. The pagestyle command then would include `\thepage` in the appropriate place. Additionally the `\pagenumbering` command resets the page number to 1. The `\pagestyle` and `\pagenumbering` apply to the page that is being constructed, so they should be used at a location where it is clear to what page they apply (see section 24).

6 What is fancyhdr

The fancyhdr macro package allows you to customize in LaTeX your page headers and footers in an easy way. You can define:

- three-part headers and footers
- decorative lines in headers and footers

- headers and footers wider than the width of the text
- multi-line headers and footers
- separate headers and footers for even and odd pages
- different headers and footers for chapter pages
- different headers and footer on pages with floats

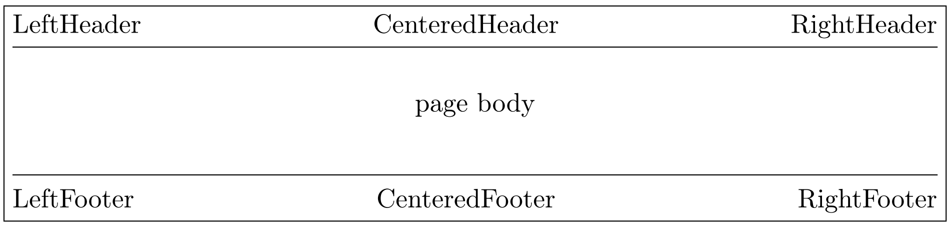
Of course, you also have complete control over fonts, uppercase and lowercase displays, etc.

7 Simple use of fancyhdr

To use this package install it is a place where LaTeX can find it (see section 1)[1], and include in the preamble of your document the commands:

```
\usepackage{fancyhdr}
\pagestyle{fancy}
```

We can visualize the page layout we can create with fancyhdr as follows:

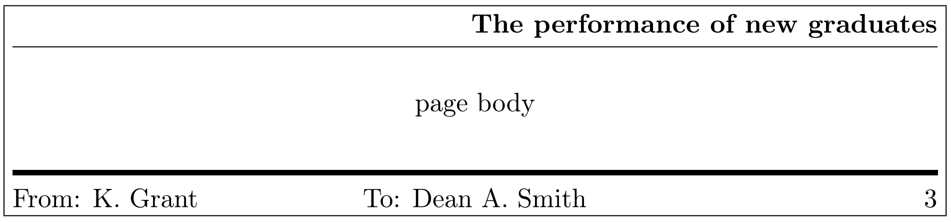


The LeftHeader and LeftFooter are left justified; the CenteredHeader and CenteredFooter are centered; the RightHeader and RightFooter are right justified.

We define each of the six “fields” and the two decorative lines separately.

8 A simple example

K. Grant is writing a report to Dean A. Smith, on “The performance of new graduates” with the following page layout:



[1] In most modern TeX installation the package is already included.

where “3” is the page number. The title: “The performance of new graduates” is bold. The rule above the footer is a bit thicker (2pt).

This is accomplished by these commands following `\pagestyle{fancy}`[2]:

```
\fancyhead[L,C]{}
\fancyhead[R]{\textbf{The performance of new graduates}}
\fancyfoot[L]{From: K. Grant}
\fancyfoot[C]{To: Dean A. Smith}
\fancyfoot[R]{\thepage}
\renewcommand{\headrulewidth}{0.4pt}
\renewcommand{\footrulewidth}{2pt}
```

(The `\thepage` macro displays the current page number. `\textbf` puts it in bold face.)

This is now fine, except that the first page does not need all these headers and footers. To eliminate all but the centered page number, issue the command

```
\thispagestyle{plain}
```

after the `\begin{document}` and the `\maketitle` commands.

Alternatively, issue

```
\thispagestyle{empty}
```

if you do not want any headers or footers.

In fact the standard LaTeX classes have the command `\maketitle` defined in such a way that a `\thispagestyle{plain}` is automatically issued. So if you *do* want the fancy layout on a page containing `\maketitle` you must issue a `\thispagestyle{fancy}` after the `\maketitle`.

9 Fancy Centering

Note: This section only applies to fancyhdr version 4.0 and later[3].

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 neighbouring 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⟩}
```

[2]Note that version 1 of fancyheadings used the `\setlength` command to change the `\...rulewidth` parameters.

[3]This comes from the nccfancyhdr package by Alexander I. Rozhenko.

This command works like

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

but does this 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⟩}}
```

You would use this in a header for example with

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

and leave the `[L,R]` parts empty.

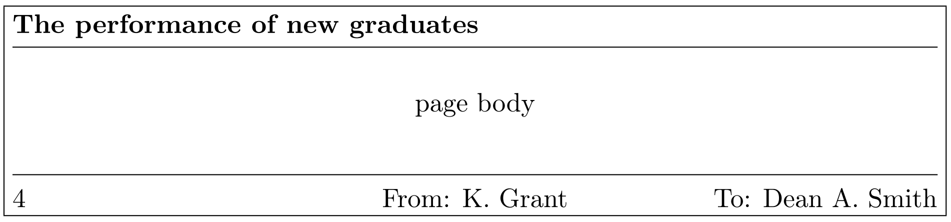
Note 1: If the whole of the `\fancycenter` is wider than `\headwidth` it will stick out on the right. See section 33 for possible solutions.

Note 2: The usage of `\fancycenter` command is not limited to the argument of header/footer marks. You can use it anywhere in your document.

10 An example of two-sided printing

Some document classes, such as `book.cls`, print two-sided by default: the even pages and the odd pages have different layouts; other document classes use the `twoside` option to print two-sided.

Now let us print the report two-sided. Let the above page layout be used for the odd (right-side) pages, and the following for the even (left-side) pages:



where "4" is the page number.

Here are the commands:

```
 \fancyhead{} % clear all header fields
```

```
\fancyhead[RO,LE]{\textbf{The performance of new graduates}}
\fancyfoot{} % clear all footer fields
\fancyfoot[LE,RO]{\thepage}
\fancyfoot[LO,CE]{From: K. Grant}
\fancyfoot[CO,RE]{To: Dean A. Smith}
\renewcommand{\headrulewidth}{0.4pt}
\renewcommand{\footrulewidth}{0.4pt}
```

The commands `\fancyhead` and `\fancyfoot` have an additional parameter between square brackets that specifies for which pages and/or parts of the header/footer they apply. The first `\fancyhead` command above omits this parameter, and thus applies to all header fields. In general this is only useful to get rid of the defaults or a previous definition, as is done here. Similar the `\fancyfoot` command without square brackets clears all footer fields. In this particular example it could be omitted as all footer fields have a value specified. The selectors that can be used between the square brackets are given in figure 2. Selectors can be combined so `\fancyhead[LE,RO]{text}` will define the field for both the left header on even pages and the right header on odd pages. If you don't give an `E` or `O` the definition applies to both. Similar for `LRC`. The selectors may be given as uppercase or lowercase letters.

E	Even page
O	Odd page
L	Left field
C	Center field
R	Right field
H	Header
F	Footer

Figure 2: Selectors

There is also a more general command `\fancyhf` that you can use to combine the specifications for headers and footers. This allows additional selectors `H` (header) and `F` (footer). In fact `\fancyhead` and `\fancyfoot` are just `\fancyhf` with `H` and `F` pre-specified, respectively.

Again, you may use `\thispagestyle{plain}` for a simple page layout for page 1.

11 Redefining page style `plain`

Some LaTeX commands, like `\chapter`, use the `\thispagestyle` command to automatically switch to the `plain` page style, thus ignoring the page style currently in effect.

They do this by issuing a `\thispagestyle{plain}` command. The most well-known places where this could happen are:

- The first pages of chapters in the `book` and `report` class
- The first page of a document in the `article` class when `\maketitle` is used
- The first page of an index

but it could happen at other places depending on the class and the packages used.

To customize even such pages you must redefine the `plain` page style. As we indicated before you could do this by defining the `\ps@plain` command, but fancyhdr gives you an easier way with the `\fancypagestyle` command. This command can be used to redefine existing page styles (like `plain`) or to define new ones, e.g. if part of your document needs a different page style. This command has two mandatory parameters: the first one is the name of the page style to be defined, the second consists of commands that change the headers and/or footers, i.e. `fancyhead` etc. Also allowed are changes to `\headrulewidth` and `\footrulewidth` or even `\headrule` and `\footrule`. The (re)defined page style uses the standard `fancy` definitions, amended by the definitions in the second parameter. In other words, those parts that are not redefined in the second parameter get their value from the `fancy` definition that is current. In particular, if the second parameter is empty, i.e. given as `{}`, then the new page style is equal to page style `fancy`.

As an example, let us redefine the `plain` style so that it will be the same as page style `fancy`:

```
\fancypagestyle{plain}{}
```

Now when these special pages use the `plain` page style, they use your redefined version.

As another example, let us redefine the `plain` style for the report in Section 10 by making the page number bold and enclosing it in en-dashes without any rules.

```
\fancypagestyle{plain}{%
  \fancyhf{}% clear all header and footer fields
  \fancyfoot[C]{\textbf{--~\thepage~--}} % except the center
  \renewcommand{\headrulewidth}{0pt}%
  \renewcommand{\footrulewidth}{0pt}%
}
```

12 Defining other page styles

Just like redefining the `plain` page style in the previous section, you can define or redefine other page styles based on page style `fancy`. This is also done with the `\fancypagestyle` command. The general form of this command is:

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

As you see, there is an optional argument between the two mandatory arguments.
For example:

```
\fancypagestyle{toc}{%
  \fancyhf{}%
  \fancyhead[RO]{\thepage}%
  \fancyhead[RO]{\textsl{TABLE OF CONTENTS}}%
  \fancyfoot[C]{\thepage}
}
```

This defines a special page style `toc` for use in the table of contents with `\pagestyle{toc}`. Inside the definition you can define the headers and/or footers, change the header and footer rules, and redefine commands like `\chaptermark` (see section 13 for an example). The headers and footers and marks that are not redefined inside the `\fancypagestyle` definition, are taken from the global page style `fancy` values.

You can also give an optional base page style to the `\fancypagestyle` command. Then the new page style will be based on the base style. This base style must be a `fancyhdr`-defined style. Also you should take care not to create circular dependencies. In this case the order of picking up the definitions (headers, footers, marks) is:

1. The definitions from the base style are taken.
2. The definitions given in the `\fancypagestyle` command override and/or augment these.
3. Any definitions that are not given by the two above, are taken from the environment at the time the new page style is used.

Only the first two parts are embedded in the page style. When no base style is given, part 1 is null.

The optional base style argument is only available since version 4.0. In this version it is also possible to redefine page style `fancy` in this way. In version 3.x and earlier this was not possible.

If you want to restore the original default definitions from page style `fancy` as described in section 14, you can use

```
\fancypagestyle{myfancy}[fancydefault]{
  . . . override some here
}
```

Page style `fancydefault` is the version of page style `fancy` that has all the initialisation embedded. Contrary to this, page style `fancy` as defined in the package uses the same defaults, but doesn't have them embedded. It picks them up from the environment. So if the environment changes, because you redefine headers, footers, mark commands, etc, the functioning of page style `fancy` changes with it. The page style `fancydefault` does not change, however. However, `fancydefault` is only available since fancyhdr version 4.0.

13 Package options

NOTE: This section applies to fancyhdr version 4.0 and later.

You can supply options to the `\usepackage` command:

```
\usepackage[⟨options⟩]{fancyhdr}
```

The following options are supported:

Option	Meaning
`nocheck`	do not check the heights of the header and footer
`compatV3`	keep some behaviour (now considered undesirable) as in version 3
`headings`	redefine the `headings` page style to be fancy-based
`myheadings`	redefine the `myheadings` page style to be fancy-based

- Option `nocheck` is described in section 17 on page 22.
- Option `compatV3` keeps two fancyhdr version 3.x (or earlier) features that are now considered undesirable.

  1. The automatic adjustment of `\headheight` or `\footskip` when these are too small. This causes the page layout to become inconsistent. See section 17 on page 22.
  2. In these previous versions the changes to the fancyhdr headers and footers (including those by `\fancyhead`, `\fancyheadoffset` and similar commands) are made globally, except within a page style defined by `\fancypagestyle`. That is, when these commands are given inside a LaTeX group, they affect the whole document, not only the group. If your document depends on this behaviour, you can give the `compatV3` package option. However, this is only considered a short-time solution. You should change your document as soon as possible to work around this problem. In version 4.0 and later, without this option, the changes are always local.

  The option is scheduled to disappear in version 5 of fancyhdr.

- The options `headings` and `myheadings` redefine the corresponding page style with fancyhdr commands (including a decorative line under the header), so that you can later select this page style as the page style for (part of) the document[4].

The page style `headings` is in some aspects similar to the default page style `fancy` settings. In the `fancy` page style, the page number is in the footer, but in the `headings` page style it is in the header. The header fields look similar, however.

[4]These options were copied from the `nccfancyhdr` package, but contrary to that package, they are not automatically selected.

Please note that these page styles redefine the `\chaptermark` and/or `\[sub]sectionmark` commands (see section 15), as do the standard LaTeX page styles. The consequence is, that if you select e.g. `\pagestyle{headings}`, the definitions of `\pagestyle{fancy}` are overridden. Also when you change the headers and/or footers while such a page style is in effect, and you then switch back to this page style, for example with `\pagestyle{headings}`, they revert to the built-in settings. Therefore it is not advisable to change the headers or footers in this way, but instead define your own page style, as explained in section 12.

14 The default layout

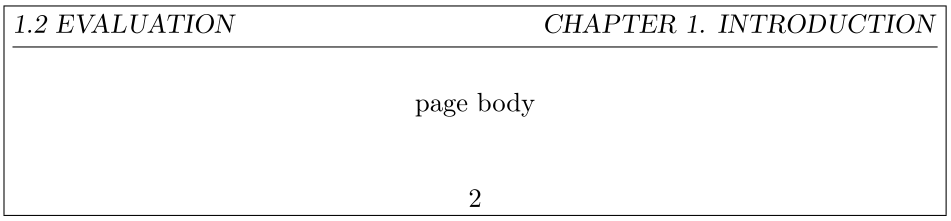
Let us use the `book.cls` documentclass and the default settings for fancyhdr; so we don't use any of the page style options in the `\usepackage{fancyhdr}` command, and we don't redefine any headers or footers. So just:

```
\usepackage{fancyhdr}
\pagestyle{fancy}
```

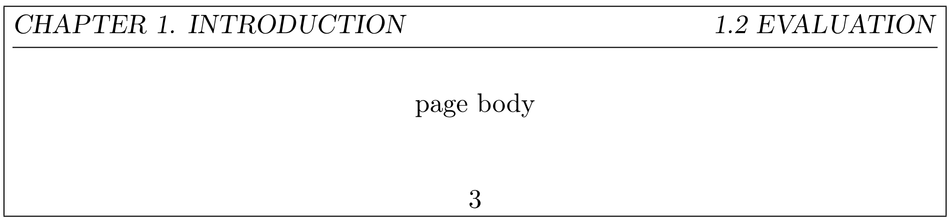
and let fancyhdr take care of everything. As mentioned before, we get a layout that is very similar to the page style `headings`.

On the pages where new chapters start, we get a centered page number in the footer; there is no header, and there are no decorative lines.

On an even page, we get the layout:



On an odd page, we get the layout:



where the header text is slanted uppercase.

This default layout is produced by the following commands:

```
\fancyhead[LE,RO]{\textsl{\rightmark}}
\fancyhead[LO,RE]{\textsl{\leftmark}}
\fancyfoot[C]{\thepage}
```

The following settings are used for the decorative lines:

`\headrulewidth`	0.4pt
`\footrulewidth`	0 pt

The header text is turned into all uppercase by the standard LaTeX code in `book.cls`.

15 The scoop on LaTeX's marks

Usually, for documents of class `book` and `report`, you may want to use chapter and section information in the headings (chapter only for one-sided printing), and for documents of class `article`, section and subsection information (section only for one-sided printing). LaTeX uses a marker mechanism to remember the chapter and section (section and subsection) information for a page; this is discussed in detail in *The LaTeX Companion*, Section 4.3.1, or *The LaTeX Companion, Second Edition*, Section 4.3.4.

There are two ways you can use and change the higher- and lower-level sectioning information available to you. The macros: `\leftmark` (higher-level) and `\rightmark` (lower-level) contain the information processed by LaTeX, and you can use them directly as shown in section 14.

These marks are set by the commands `\markboth{leftmark}{rightmark}` and `\markright{rightmark}`. These commands are usually used inside commands like `\chaptermark` and `\sectionmark` but they can be also be given directly in your document, although this not very usual.

The `\leftmark` contains the **L**eft argument of the ***L**ast* `\markboth` on the page, the `\rightmark` contains the **R**ight argument of the *fi**R**st* `\markboth` or the only argument of the *fi**R**st* `\markright` on the page. If no marks are present on a page they are "inherited" from the previous page.

You can influence how chapter, section, and subsection information (only two of them!) is displayed by redefining the `\chaptermark`, `\sectionmark`, and `\subsectionmark` commands[5]. You must put the redefinition after the first call of `\pagestyle{fancy}` as this sets up the defaults.

Let us illustrate this with chapter info. It is made up of three parts:

- the number (say, 2), displayed by the macro `\thechapter`
- the name (in English, Chapter), displayed by the macro `\chaptername`
- the title, contained in the argument of `\chapter`.

We combine these below with `\markboth` in `\chaptermark`.

For the lower-level sectioning information, we do the same with `\markright` in `\sectionmark`.

So if "2. Implementation" is the current chapter and "2.1. First steps" is the current section, then

[5]There are similar commands for `paragraph` and `subparagraph` but they are seldom used.

```
\renewcommand{\chaptermark}[1]{%
  \markboth{\chaptername\ \thechapter.\ #1}{}}
\renewcommand{\sectionmark}[1]{\markright{\thesection.\ #1}}
```

will give "Chapter 2. Implementation" and "2.1. First steps"

Redefining the `\chaptermark` and `\sectionmark` commands may not eliminate all uppercaseness. E.g. the bibliography will have a title of BIBLIOGRAPHY in the header, as the `\MakeUppercase` is explicitly given in the definition of `\thebibliography`. Similar for INDEX etc. If you don't want to redefine these commands, you can use the `\nouppercase` command that fancyhdr makes available in the header and footer fields. Note that this may screw other things, like uppercase roman numerals in your headers, so it should be used with care. Essentially this command typesets its argument in an environment where `\MakeUppercase` and `\uppercase` are changed into do-nothing operations.

```
\fancyhead[L]{\nouppercase{\rightmark}}
\fancyhead[R]{\nouppercase{\leftmark}}
```

Figure 3 shows some variants for "Chapter 2. Implementation" (the last example is appropriate in some non-English languages). The `%` signs at the end of the lines are to prevent unwanted space. Normally you would continue the lines and remove these `%` signs[6].

It should be noted that the LaTeX marking mechanism works fine with chapters (which always start on a new page) and sections (which are reasonably long). It does not work quite as well with short sections and subsections. This is a problem with LaTeX, not with fancyhdr.

As an example let's take a page layout where the leftmarks are generated by the sections and the rightmarks by the subsections (as is default in the `article` class). Take a page with some short sections, e.g.
Section 1.
subsection 1.1
subsection 1.2
Section 2.

As the leftmark contains the *last* mark of the page it will be "Section 2.", and the rightmark will be "subsection 1.1" as it will be the *first* mark of the page. So the page header info will combine section 2 with subsection 1.1 which isn't very nice. One thing you can do in these cases is use only the `\rightmark`s and redefine `\sectionmark` accordingly.

However, the extramarks package described in section 25 contains a command `\firstleftmark` that can be used to get the first of the leftmarks on the page in the header. This might be the best solution in this situation. Now the header will contain "Section 1." in the situation described above.

[6] The `\MakeUppercase` command is used in LaTeX to generate uppercase text, while `\uppercase` is the plain TeX command for this. The difference is that `\MakeUppercase` also deals with non-ASCII letters.

Code:	Prints:

`\renewcommand{\chaptermark}[1]{%`
` \markboth{\chaptername`
` \ \thechapter.\ #1}{}}`
	Chapter 2. Implementation

`\renewcommand{\chaptermark}[1]{%`
` \markboth{\MakeUppercase{%`
` \chaptername}\ \thechapter.%`
` \ #1}{}}`
	CHAPTER 2. Implementation

`\renewcommand{\chaptermark}[1]{%`
` \markboth{\MakeUppercase{%`
` \chaptername\ \thechapter.%`
` \ #1}}{}}`
	CHAPTER 2. IMPLEMENTATION

`\renewcommand{\chaptermark}[1]{%`
` \markboth{#1}{}}`
	Implementation

`\renewcommand{\chaptermark}[1]{%`
` \markboth{\thechapter.\ #1}{}}`
	2. Implementation

`\renewcommand{\chaptermark}[1]{%`
` \markboth{\thechapter.%`
` \ \chaptername.\ #1}{}}`
	2. Chapter. Implementation

Figure 3: Marker variants

```
\usepackage{extramarks}
  . . .
\fancyhead[R]{\firstleftmark}
```

Another problem with the marks in the standard LaTeX classes is that the higher level sectioning commands (e.g. `\chapter`) call `\markboth` with an empty right argument. This means that on the first page of a chapter (or a section in article style) the `\rightmark` will be empty. The underlying problem is that the TeX machinery has only one `\mark`. All the marks must be packed together in this one. So there are no independent left or right marks. That also applies to the extra marks as described in section 25. If this is a problem you must manually insert extra `\markright` commands or redefine the `\chaptermark` (`\sectionmark`) commands to issue a `\markboth` command with two decent parameters.

As a final remark you should also note that the `*` forms of the `\chapter` etc. commands do *not* call the mark commands. So if you want your preface to set the header info but not be numbered nor be put in the table of contents, you must issue the `\markboth` command yourself, e.g.

```
\chapter*{Preface}
```

```
\markboth{Preface}{}
```

Or in a documentclass without chapters:

```
\section*{Preface}
\markboth{Preface}{}
```

16 Dictionary style headers

Dictionaries and concordances usually have a header containing the first word defined on the page or both the first and the last words. This can easily be accomplished with fancyhdr and LaTeX's `mark` mechanism. Of course if you use the marks for dictionary style headers, you cannot use them for chapter and section information, so if there are also chapters and sections present, you must redefine the `\chaptermark` and `\sectionmark` to make them harmless:

```
\renewcommand{\chaptermark}[1]{}
\renewcommand{\sectionmark}[1]{}
```

Now you do a `\markboth{#1}{#1}` for each dictionary or concordance entry `#1` and use `\rightmark` for the first entry defined on the page and `\leftmark` for the last one.

If you want to use a header entry of the form firstword–lastword it would be nice if this would be reduced to just the form firstword if both are the same. This could happen if there is just one entry on the page. In this case a test must be made to check if the marks are the same. However, TeX's marks are strange beasts, which cannot be compared out of the box with the plain TeX `\if` commands. Fortunately the ifthen package works well:

```
\newcommand{\mymarks}{
  \ifthenelse{\equal{\leftmark}{\rightmark}}
    {\rightmark} % if equal
    {\rightmark--\leftmark}} % if not equal
\fancyhead[LE,RO]{\mymarks}
\fancyhead[LO,RE]{\thepage}
```

17 Fancy layouts

You can make a multi-line field with the `\\` command. It is also possible to put extra space in a field with the `\vspace` command. Note that if you do this you will probably have to increase the height of the header (`\headheight`) and/or of the footer (`\footskip`), otherwise you may get error messages "Overfull `\vbox` ... has occurred while `\output` is active"[7]. See the warning below. See also Section 4.1 of the LaTeX *Companion* for detail.

For instance, the following code will place the section title and the subsection title of an article in two lines in the upper right hand corner:

```
\documentclass{article}
\usepackage{fancyhdr}
\pagestyle{fancy}
\addtolength{\headheight}{\baselineskip}
\renewcommand{\sectionmark}[1]{\markboth{#1}{}}
\renewcommand{\subsectionmark}[1]{\markright{#1}}
\fancyhead[R]{\leftmark\\\rightmark}
```

You can also customize the decorative lines. You can make the decorative line in the header quite thick with

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

or you can make the decorative line in the footer disappear with

```
\renewcommand{\footrulewidth}{0pt}
```

The decorative lines, themselves, are defined in the two macros `\headrule` and `\footrule`. For instance, if you want a dotted line rather than a solid line in the header, redefine the command `\headrule`:

```
\renewcommand{\headrule}{\vbox to 0pt
    {\makebox[\headwidth]{\dotfill}\vss}}
```

The redefined `\headrule` should preferably take up no vertical space, as in the example above, and as in the standard definition. If it does take vertical space, the header may come too close to the text, or even intrude in the text. In that case fancyhdr will give you a warning that `\headheight` is too small. Like

```
Package fancyhdr Warning: \headheight is too small (12.0pt):
(fancyhdr)                Make it at least 14.0pt, for example:
(fancyhdr)                \setlength{\headheight}{14.0pt}.
(fancyhdr)                You might also make \topmargin smaller to compensate:
(fancyhdr)                \addtolength{\topmargin}{-2.0pt}.
```

[7]If you use `11pt` or `12pt` you will probably also have to do this, because LaTeX's defaults are quite small

You will probably get this warning on every page. **Note:** Before version 4.0, `fancyhdr` would change the `\headheight` itself, causing the text on the following pages to come out lower than on this page. This appeared to be confusing, so since version 4.0 this is no longer done (except when you give the `compatV3` package option. You should not give this as a permanent solution, however, but solve the problem). Therefore you are strongly advised to redefine `\headheight` in the preamble, like this:

```
\setlength{\headheight}{14pt}
```

This would cause the main text to be put 2pt lower on the page, which might be undesirable. You can compensate this by making `\topmargin` correspondingly smaller, for example

```
\addtolength{\topmargin}{-2pt}
```

A similar change would be necessary for `\footskip` if the footer comes out too tall.

You can also eliminate this check completely by using the `nocheck` option of the package. But this may risk unwanted run-ins of the header or footer with other text. So this is generally discouraged. It is better to change `\headheight`, `\footskip`, and/or `\topmargin`. But in cases where you generate the LaTeX code automatically, and the software does not know how tall the header or footer will be, this may be handy.

As an alternative to changing `\headrulewidth` to 0 to have the rule disappear, you can also make it empty with

```
\renewcommand{\headrule}{}
```

Visually this makes no difference, but it is more difficult to restore it later to its default value.

Finally, let us make a real 'decorative' line[8].

```
\usepackage{fourier-orns}
...
\renewcommand\headrule{%
      \hrulefill
      \raisebox{-2.1pt}
          {\quad\decofourleft\decotwo\decofourright\quad}%
      \hrulefill}
```

This gives us the following headrule:

[8]Based upon an idea by Wayne Chan.

Note that we haven't taken care to make this decorative line occupy zero vertical space. The consequence is that it will extend towards the text and that we will get the warning about `\headheight` too small. So we should change `\headheight` as given above. Another problem is that the distance between the line and the header text is quite big. We can reduce this by putting a negative `\vspace` above it, like

```
\renewcommand\headrule{%
      \vspace{-6pt}
      \hrulefill
      \raisebox{-2.1pt}
          {\quad\decofourleft\decotwo\decofourright\quad}%
      \hrulefill}
```

We can use the same code for the `\footrule`, but we wouldn't need the `\vspace`. If you want to change the distance between that decorative line and the footer text you need to adjust the parameter `\footruleskip`. It defines the distance between the decorative line in the footer and the top of the footer text line. By default it is set to 30% of the normal line distance. You may want to adjust it if you use unusally large or small fonts in the footer. Change it with `\renewcommand`.

`\footruleskip`

You can also change the distance between the baseline of the header text and the decorative line in the header. Normally this distance is determined by the maximum depth of possible descenders in the text, which is 30% of the normal line distance. You can increase or decrease this distance by defining the macro `\headruleskip`, similar to `\footruleskip` [9]. This defines the extra distance. The default value is 0pt, and positive values make the distance larger, and negative values make the distance shorter. Please note that this does not change the position of the decorative line with respect to the page, but it shifts the header text. If you want to keep the header text fixed, but move the decorative line, then you must also change the parameter `\headsep` (see figure 1).

`\headruleskip`

The header and footer in this page show the *strut* (the amount of space in the text area above and below the baseline), and the `\headruleskip` and `\footruleskip`. For this page `\headruleskip` is 4pt.

18 Two book examples

The following definitions give an approximation of the style used in L. Lamport's LaTeX book.

Lamport's header overhangs the outside margin. This is done as follows.

The width of headers and footers is `\headwidth`, which by default equals the width of the text: `\textwidth`. You can make the width wider (or narrower) by redefining `\headwidth` with the `\setlength` and `\addtolength` commands. To overhang the outside margin where the marginal notes are printed, add both `\marginparsep` and `\marginparwidth` to `\headwidth` with the commands:

[9] (But `\headruleskip` is only available since version 4.0.)

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

It is safest to issue these commands *after* the first `\pagestyle{fancy}` command.

And now a complete definition of Lamport's book style. The header has the width of the text plus the marginpar area. The header on even pages has the page number on the left, and the chapter title on the right. On odd pages it has the section title preceded by the section number on the left and the page number on the right. All in boldface. There is no footer. The `plain` style is redefined to have no header and no footer. (In the LaTeX book this makes sense because each chapter begins with a page that contains only a drawing. In most other cases you probably would want a page number on the page.)

```
\documentclass{book}
\usepackage{fancyhdr}
\pagestyle{fancy}
\addtolength{\headwidth}{\marginparsep}
\addtolength{\headwidth}{\marginparwidth}
\renewcommand{\chaptermark}[1]{\markboth{#1}{}}
\renewcommand{\sectionmark}[1]{\markright{\thesection\ #1}}
\fancyhf{}
\fancyhead[LE,RO]{\textbf{\thepage}}
\fancyhead[LO]{\textbf{\rightmark}}
\fancyhead[RE]{\textbf{\leftmark}}
\fancypagestyle{plain}{%
   \fancyhead{} % get rid of headers
   \renewcommand{\headrulewidth}{0pt} % and the line
}
```

Notice that the `\chaptermark` and `\sectionmark` commands have been redefined to eliminate the chapter numbers and the uppercaseness.

For more control about the horizontal position of the headers and/or footers, fancyhdr has additional commands to specify the offset of the header and/or footer elements. Use `\fancyhfoffset[place]{length}` to offset one or more elements. The `place` parameter is like the optional parameter of `\fancyhf`, like `L R E O`, except that `C` cannot be used. It specifies for which elements the offset should be applied. The `length` parameter specifies the actual offset. Positive values move the element outward (into the margin), negative values inward. There are also specialised commands `\fancyheadoffset` and `\fancyfootoffset`, which have the `H` and `F` parameter pre-applied, respectively.

When you use these commands, LaTeX will recalculate `\headwidth`, based on the given parameters.

So the above example could also have been done with (N.B. You can only use such an expression as a length parameter if the calc package is used):

```
\fancyheadoffset[LE,RO]{\marginparsep+\marginparwidth}
```

For the second example, we take the $\mathcal{A}\mathcal{M}\mathcal{S}$-LaTeX book[10].
Chapter pages have no headers or footers. So we declare

```
\thispagestyle{empty}
```

for every chapter page, and we do not need to redefine plain.

Chapter and section titles appear in the form: 2. IMPLEMENTATION, so we have to redefine `\chaptermark` and `\sectionmark` as follows (see Section 15):

```
\renewcommand{\chaptermark}[1]%
   {\markboth{\MakeUppercase{\thechapter.\ #1}}{}}
\renewcommand{\sectionmark}[1]%
   {\markright{\MakeUppercase{\thesection.\ #1}}}
```

On an even page, the page number is printed as the left header and the chapter info as the right header; on an odd page, the section info is printed as the left header and the page number as the right header. The center headers are empty. There are no footers.

There is a decorative line in the header. It is 0.5pt wide, so we need the commands:

```
\renewcommand{\headrulewidth}{0.5pt}
\renewcommand{\footrulewidth}{0pt}
```

The font used in the headers is 9 pt bold Helvetica. The PSNFSS system by Sebastian Rahtz uses the short (Karl Berry) name `phv` for Helvetica, so this font is selected with the commands:

```
\fontfamily{phv}\fontseries{b}\fontsize{9}{11}\selectfont
```

(See Sections 7.6.1 and 11.9.1 of the LaTeX *Companion* or Sections 7.6 and 7.9.1 of *The LaTeX Companion, Second Edition.*) Let us define a shorthand for this:

```
\newcommand{\helv}{%
   \fontfamily{phv}\fontseries{b}\fontsize{9}{11}\selectfont}
```

Now we are ready for the page layout:

```
\documentclass{book}
\usepackage{fancyhdr}
```

[10]George Gratzer, *Math into LaTeX, An Introduction to LaTeX and $\mathcal{A}\mathcal{M}\mathcal{S}$-LaTeX*, Birkhauser, Boston.

```
\pagestyle{fancy}
\renewcommand{\chaptermark}[1]%
   {\markboth{\MakeUppercase{\thechapter.\ #1}}{}}
\renewcommand{\sectionmark}[1]%
   {\markright{\MakeUppercase{\thesection.\ #1}}}
\renewcommand{\headrulewidth}{0.5pt}
\renewcommand{\footrulewidth}{0pt}
\newcommand{\helv}{%
   \fontfamily{phv}\fontseries{b}\fontsize{9}{11}\selectfont}
\fancyhf{}
\fancyhead[LE,RO]{\helv \thepage}
\fancyhead[LO]{\helv \rightmark}
\fancyhead[RE]{\helv \leftmark}
```

19 Special page layout for float pages

Some people want to have a special layout for float pages (pages only containing floats). As these pages are generated autonomically by LaTeX, the user doesn't have any control over them. There is no `\thispagestyle` for float pages and any change of the page style will at least also affect the page before the float page. With fancyhdr, however, you can specify in each of the header- or footer fields

`\iffloatpage{`⟨*value for float page*⟩`}{`⟨*value for other pages*⟩`}`

You can even use this to get rid of the decorative line on float pages only by defining:

```
\renewcommand{\headrulewidth}{\iffloatpage{0pt}{0.4pt}}
```

Sometimes you may want to change the layout also for pages that contain a float on the top of the page, a float on the bottom of the page or a footnote on the bottom of the page.

fancyhdr gives you the commands `\iftopfloat`, `\ifbotfloat` and `\iffootnote` similar to `\iffloatpage`. For example:

```
\fancyhead[R]{\iftopfloat{This page has a topfloat}
                         {There is no topfloat here}}
```

Note: Marks in floats will not be visible in LaTeX's output routine, so it is not useful to put marks in floats. So there is currently no way to let a float (e.g. a figure caption) influence the page header or footer.

20 Those blank pages

In the `book` class when the `openany` option is not given or in the `report` class when the `openright` option is given, chapters start at odd-numbered pages, half of the time causing a blank page to be inserted. Some people prefer this page to be completely empty, i.e. without headers and footers. This cannot be done with `\thispagestyle` as this command would have to be issued on the *previous* page. There is, however, no magic necessary to get this done:

```
\clearpage\begingroup\pagestyle{empty}\cleardoublepage\endgroup
```

As the `\pagestyle{empty}` is enclosed in a group it only affects the page that may be generated by the `\cleardoublepage`. You can of course put the above in a private command. If you want to have this done automatically at each chapter start or when you want some other text on the page then you must redefine the `\cleardoublepage` command.

```
\makeatletter
\def\cleardoublepage{\clearpage\if@twoside \ifodd\c@page\else
 \begingroup
  \mbox{}
  \vspace*{\fill}
  \begin{center}
    This page intentionally contains only this sentence.
  \end{center}
  \vspace{\fill}
  \thispagestyle{empty}
  \newpage
  \if@twocolumn\mbox{}\newpage\fi
 \endgroup\fi\fi}
\makeatother
```

21 N of M style page numbers

Some document writers prefer the pages to be numbered as n of m where m is the number of pages in the document. There is a package lastpage available which you can use with fancyhdr as follows:

```
\usepackage{lastpage}
...
\fancyfoot[C]{\thepage\ of \pageref{LastPage}}
```

Because you want the pages with pagestyle `plain` to contain the same style of page numbers, you will have to redefine this pagestyle too.

```
\fancypagestyle{plain}{\fancyhead{}\renewcommand{\headrule}{}}
```

We clear all the headers including its rule. The footer will be "inherited" from the pagestyle `fancy`.

The value of the `LastPage` label can be used to make different headers or footers on the last page of a document. E.g. if you want the footer of every odd page, except if it is the last one, to contain the text "please turn over", this can be done as follows:

```
\usepackage{lastpage}
\usepackage{ifthen}
...
\fancyfoot[R]{\ifthenelse{\isodd{\value{page}} \and \not
     \(\value{page}=\pageref{LastPage}\)}{please turn over}{}}
```

In order to get the number of pages correctly used, you usually have to do one additional LaTeX run.

22 Chapter or section related page numbers

In technical documentation very often page numbers are used of the form 2-10 where the first number is the chapter number and the second is the pagenumber relative to the chapter. Sometimes section is used rather than chapter. The package `chappg` can be used to get this format.

Basically this package redefines `\thepage` as `\thechapter\chappgsep\arabic{page}`, where `\chappgsep` by default is '-'. If you want do use a different separator, you must redefine `\chappgsep`, for example to use an en-dash:

```
\renewcommand{\chappgsep}{--}
```

To use a different prefix, for example the section number, use the `\pagenumbering{bychapter}` command with an optional argument specifying the prefix.

```
\pagenumbering[\thesection]{bychapter}
```

What the package also does is reset the page number to 1 at the beginning of each chapter.

In the frontmatter of your document (for example the Table of Contents) there will be no chapter numbers. Therefore a simple page number will be used there. This may be confusing, so you might prefer to use roman pagenumbers in the front matter. Do this by using `\pagenumbering{roman}` in the beginning of the document and `pagenumbering{bychapter}` after the first `\chapter` command. If you want to do it before the `\chapter` command you must precede it by a `\newpage` command (see the next section).

```
\pagenumbering{roman}
```

```
\tableofcontents
\newpage
\pagenumbering{bychapter}
\chapter{Introduction}
```

There is a caveat when you have appendices in your document. Before the `\appendix` command you should give a `\clearpage` or `\cleardoublepage`. See the `chappg` documentation for details.

There is a fundamental difference between the page numbering of the style "m of n" as described in the previous section and the current one. The m of n style is only used in the page header or footer, but not in the table of contents, index, or references like "*See page* xx". Therefore it does not change the command `\thepage`. The page numbering style "2-10", however should be used in all references to the page number, therefore it must be done by redefining `\thepage`.

23 Switching page styles

Page style `fancy`, if not redefined, does not have the definitions of the headers and footers built-in, but they are defined in the document, globally, or locally in a group. This also applies to the definitions of the `\chaptermark` and/or `\[sub]sectionmark` commands. So if you want to switch from another page style to the `fancy` page style later in the document, and that other page style has changed for example the `\chaptermark` and/or `\[sub]sectionmark` commands, you will have to redefine these yourself and maybe also the definitions of the headers and footers, at that point. For example

```
\pagestyle{fancy}
\renewcommand{\chaptermark}[1]{\markboth{Chapter \thechapter. ##1}{}}
\renewcommand{\sectionmark}[1]{\markright{\thesection\ ##1}}
```

Please note that you have to double the `#` signs, because the definitions are inside a macro.

If the previous page style was one of the standard LaTeX page styles, or some page style that is not based on `fancyhdr`, then the definitions of `\fancyhead` or `\fancyfoot` are not affected. So strictly you don't have to include them. But if it was based on `fancyhdr` and had different definitions, you will get the wrong headers and/or footers when you switch back to page style `fancy`. So it is safer to include them anyway.

A better possibility is to define your own page style, and include these definitions in that page style:

```
\fancypagestyle{myfancy}{
  \renewcommand{\chaptermark}[1]{\markboth{Chapter \thechapter. ##1}{}}
  \renewcommand{\sectionmark}[1]{\markright{\thesection\ ##1}}
  \fancyhead{...}
}
```

```
...
\pagestyle{myfancy}
```

In general, when you use only one page style `fancy` in your document, with the occasional `\thispagestyle` excursion to page style `plain` or `empty`, you can just keep the definitions globally in your document, but as soon as you use more than one page style, and swicth between them, it is highly advisable to define them (including page style `fancy`) with `\fancypagestyle` and put all the relevant definitions inside them.

There is another caveat, when switching page styles, if they have different definitions of `\chaptermark` in the `book` or `report` document class or similar ones. When you put the `\pagestyle` command *after* the `\chapter` command, then the `\chapter` command calls the `\chaptermark` of the previous page style, which is probably not what you intended. So you must issue the `\pagestyle` command *before* the `\chapter` command. But this would probably change the page style of the previous page, which is too early. Therefore you would have to give a `\newpage`, `\clearpage` or `\cleardoublepage` command before the `\pagestyle` command, so that the last page will be finished with the previous page style. I.e., the proper sequence is:

```
\newpage % (or \clearpage or \cleardoublepage)
\pagestyle{newstyle}
\chapter{My New Chapter}
```

24 When to change the headers and footers?

In the previous section we switched page styles at a point that has a clear page break (the beginning of a chapter). Sometimes you want to change only a header or footer without changing the whole page style.

It should be noted that although the fancyhdr commands like `\fancyhead` take effect immediately, this does not mean that any "variables" used in these commands get the value they have at the place where these commands are given. E.g. if `\fancyfoot[C]{\thepage}` is given the page number that will be inserted in the footer is not the page number of the page where this command is given, but rather the page number of the actual page where the footer is constructed. Of course for the page number this is what you expect, but it is also true for other commands. There is a difference, however. The page number is incremented *after* the page has been constructed. When we have our own "variables", however, these are usually changed in the middle of our text.

As an example we take a book where each chapter is written by a different author. If we want the name of the author in the header opposite the chapter title, we can use the following commands:

```
\newcommand{\TheAuthor}{}
\newcommand{\Author}[1]{\renewcommand{\TheAuthor}{#1}}
```

```
\fancyhead[LE,RO]{\TheAuthor}
```

and start each chapter with the command `\Author{Real Name}`. If, however, the author name would be changed before a page is completed the wrong author could come in the header. This would be the case if you gave the above command *before* the `\chapter` command rather than after it. So we give the `\Author` command after the `\chapter` command:

```
\chapter{Chapter Title}
\Author{Author Name}
```

As a chapter starts on a new page, we can be sure that the `\Author` command comes at the same page as the chapter start.

Another source of problems is the fact that TEX's output routine processes commands ahead, so it may already have processed some commands that produce text that will appear on the next page. So if our book was not divided into chapters, but into sections, we cannot use the similar system:

```
%%% NOTE: This may not work %%%%
\section{Chapter Title}
\Author{Author Name}
```

because in this case, when this command comes at the end of a page, the "variable" `\TheAuthor` could be set at that page, but then TEX could decide to move the section title to the next page. And then the author name would appear one page too early. This problem can be solved using marks. In fact this is the whole reason the mark mechanism was developed in TEX. See section 25.

The same applies to other changes in the middle of a page, e.g. to change the page numbering from roman to arabic (with `\pagenumbering`). For the same reason `\thispagestyle{mystyle}` will not always work in the middle of a page.

Some of these changes can be accomplished by using the mark mechanism as may be seen in section 15 and the next section.

In the remainder of this section we look at two different cases of changing the page style in the middle of a page: changing the style of the current page and changing the style of the next page.

24.1 Changing the page style of the current page

So now we are giving an example how to change the headers and footers, only on the current page. In some cases this can be done by the `\thispagestyle` command. This changes the page style for the "current" page only. But then we may be hit by the problem mentioned above. LATEX may have a different idea about the "current" page than you. The use of `\thispagestyle` is OK if you can be sure that the text where the command `\thispagestyle` is executed is the same page as where the surrounding text appears. So for example directly after a `\chapter` command, or after a `\newpage`. However, when the command is given near the end of a page, LaTeX may execute the command, and then decide that the page is full and move the text that contains the command to the next page. So now the page style is changed on one page earlier than was intended.

A good solution to this problem is to put a label, like `\label{otherpagestyle}` in the text where you want the different page style, and then in the header and/or footer definitions compare the page number with the label page number and choose the proper value. For example, if we want to replace the section title on the special page with "MYFANCY SECTION", like in

```
\fancypagestyle{myfancy}{
  \fancyhead[LE,RO]{MYFANCY SECTION}
}
```

we define a new pagestyle that makes the choice:

```
\usepackage{ifthen}
. . .
\fancypagestyle{switch}{
  \fancyhead[LE,RO]{%
    \ifthenelse{\value{page}=\pageref{otherpagestyle}}
      {MYFANCY SECTION}
      {\textsl{\rightmark}}}
}
```

where `\textsl{\rightmark}` is the normal value of the header field from `\pagestyle{fancy}`. Now we choose `\pagestyle{switch}` before our text, or even for the whole document.

There can still be some ambiguity on which page gets the different header. For example, if the text says:

> This page gets a different header than the surrounding pages.

where do you put the `\label`? LaTeX could break the page between "This" and "page", and then would you want the special heading on the page where "This" appears, or on the page where "page" appears. It depends on the positioning of the `\label` command. Probably it is safer to make sure the sentence isn't broken. This can be dome by putting the text in a `\parbox` or `minipage` environment.

```
\noindent
\begin{minipage}{\textwidth}
  This page should have a different header than the surrounding pages.
  \label{otherpagestyle}
  It is done with the \verb|\pagestyle{switch}| command, that
  has tests in the header field definitions. This chooses the actual
  header depending on the page number.
\end{minipage}
```

The `\noindent` is necessary, otherwise the whole `minipage` will be shifted right by the paragraph indentation.

Note that you cannot reset the page style immediately after this code, as this may still influence the current page. If you want to reset it, for example to `\pagestyle{fancy}`, you must be sure that it happens on a following page. But in this case it isn't even necessary, as the special page style acts as the default on all pages except the special page.

The special header and footer in page 23, which show the struts are done in a similar way, although the header and footer are a bit more elaborated there. Also there is another complication there, as we also want to make both `\headruleskip` and `\footrulewidth` dependent on the page number. Unfortunately, this cannot be done with a simple `\ifthenelse` command. Both `\headruleskip` and `\footrulewidth` are eventually used as length parameters, and this requires that they are *expandable*. However, the `\ifthenelse` construct is not expandable, so you will get strange error messages if you use something like

```
%%% NOTE: This does not work %%%%
\renewcommand{\footrulewidth}{%
  \ifthenelse{\value{page}=\pageref{otherpagestyle}}{0.4pt}{0pt}%
}
```

`\fancyheadinit` `\fancyfootinit` `\fancyhfinit`

For cases like this fancyhdr version 4.0 and later has some new commands `\fancyheadinit`, `\fancyfootinit` and `\fancyhfinit`. With `\fancyheadinit{`⟨*code*⟩`}` you can define some code that will be executed just before the construction of the header. As it is executed in the header, it can test the correct page number, because the counter `page` is guaranteed to have the correct value in the headers and footers. Similarly, the code in `\fancyfootinit{`⟨*code*⟩`}` is executed in the footer. And `\fancyhfinit{`⟨*code*⟩`}` sets its code for both the header and the footer. Now we can set for example `\headruleskip` or `\footrulewidth` depending on the page number. So instead of putting the test inside the definition of `\headruleskip`, we can put it outside, and then we can use the command `\ifthenelse`. So we put the following in `\pagestyle{switch}`:

```
  \fancyheadinit{%
    \ifthenelse{\value{page}=\pageref{otherpagestyle}}
      {\renewcommand{\headruleskip}{4pt}}
      {\renewcommand{\headruleskip}{0pt}}
  }
  \fancyfootinit{%
    \ifthenelse{\value{page}=\pageref{otherpagestyle}}
      {\renewcommand{\footrulewidth}{0.4pt}}
      {\renewcommand{\footrulewidth}{0pt}}
  }
```

Now here is the definition of the page style used for page 23.

```
\fancypagestyle{showstruts}{%
  \fancyhead[L]{%
    \ifthenelse{\value{page}=\pageref{showstruts}}%
      {\strutheader}%
      {\rightmark}%
  }
  \fancyfoot[L]{%
    \ifthenelse{\value{page}=\pageref{showstruts}}%
      {\strutfooter}%
      {}%
  }
  \fancyheadinit{%
    \ifthenelse{\value{page}=\pageref{showstruts}}%
      {\renewcommand{\headruleskip}{4pt}}%
      {\renewcommand{\headruleskip}{0pt}}%
  }
  \fancyfootinit{%
    \ifthenelse{\value{page}=\pageref{showstruts}}%
      {\renewcommand{\footrulewidth}{0.4pt}}%
      {\renewcommand{\footrulewidth}{0pt}}%
  }
}
```

The label used on that page is `showstruts`. `\strutheader` and `\strutfooter` are macros that contain the code to draw these pictures. In this example the values for `\headruleskip` and `\footrulewidth` in the *else* case are the same as the global values. So we could have left these *else* parts empty. Then they would keep the global values. However, often explicit is better than implicit.

These initialisation commands cannot be used to make global changes to the page, for example to `\headheight`. Neither can you use them to change `\fancyhead` or `\fancyfoot`, because these have already been set up. But you can use it to set the color and font of the header and/or footer, for example to get large, red text in the headers and footers on this specific page:

```
\fancyhfinit{%
  \ifthenelse{\value{page}=\pageref{otherpagestyle}}
    {\color{red}\Large}
    {}
  }
```

24.2 Changing the page style of the next page

If you want the change of the page style to take effect at the next page you must make sure that the current page is finished. In most cases this can be done by issuing a `\newpage` or `\clearpage` command before any changes. However, this will immediately end the current page, possibly leaving you with a half-empty page, which may be undesirable.

If this is not what you want, you can use the afterpage package with:

`\afterpage{\fancyhead[L]{new value}}` or
`\afterpage{\pagenumbering{roman}}`.

You cannot use `\afterpage` to change the `\pagestyle` as the commands issued by `\afterpage` are local in a group, and the `\pagestyle` command makes only local changes. The `\pagenumbering` and the `\thispagestyle` command make global changes, as well as changes to LaTeX's counters, such as `\setcounter` and `\addtocounter`. So these can be used[11]. Here is an example to change the page style of the next page with `\afterpage`:

```
\usepackage{afterpage}
\usepackage{fancyhdr}
\fancypagestyle{myfancy}{
  \fancyhead[LE,RO]{\textbf{MYFANCY SECTION}}
  \fancyhead[LO,RE]{\textbf{MYFANCY CHAPTER}}
  \fancyfoot[C]{\textbf{--~\thepage~--}}
}
. . .
\afterpage{\thispagestyle{myfancy}}
```

Then the page after this code will have the page style `myfancy`.

25 Headers and footers induced by the text

We have seen how we can use LaTeX's marks to get information from the document contents to the headers and footers. The marks mechanism is the only reliable mechanism that you can use to get changing information to the headers or footers. This is because LaTeX may be processing your document ahead before deciding to break the page.

Sometimes the two marks that LaTeX offers are not enough. An example is the following:

> If a solution to an exercise goes across a page break, then I would like to have "(Continued on next page...)" at the bottom of the first page and "(Continued...)" at the top in the margin of the next page.

You cannot use LaTeX's mark mechanisms for this if you also want to use chapter and section information.

The extramarks package gives you two extra marks that can be used in this situation. Here is a way to use this package:

```
\usepackage{extramarks}
```

[11]In fancyhdr version 3 and earlier the commands like `\fancyhead` and `\fancyfoot` also made global changes. This is no longer the case in version 4.0 and later.

```
...
\pagestyle{fancy}
\fancyhead[L]{\firstxmark}
\fancyfoot[R]{\lastxmark}
\fancypagestyle{plain}{\fancyhead{}\renewcommand{\headrule}{}}
...
\extramarks{}{}% 1
\extramarks{Continued\ldots}{Continued on next page\ldots}% 2
...
Some text that may or may not cross a page boundary...
...
\extramarks{Continued\ldots}{}% 3
\extramarks{}{}% 4
```

Note that we redefine the `plain` page style, so that on the first page of a chapter also the footer will be given if necessary. We assume that a 'Continued' block will not cross chapter boundaries, so no header will be necessary on these pages. Also the `\extramarks` command must be close to the text, i.e no empty lines (paragraph boundaries) should intervene. Otherwise the page may be broken at that boundary and the extramarks would come on the wrong page. The final `\extramarks{}{}` is to prevent the 'Continued. . . ' header to appear on the following pages.

Explanation: There are two new marks that can be used in the page layout with this package: If commands of the form `\extramarks{`m_1`}{`m_2`}` are given `\firstxmark` gives you the first m_1 value and `\lastxmark` gives you the last m_2 value of the current page. In the above example, when the complete block falls on the same page, the `\firstxmark` will be the empty parameter of the first `\extramarks` command (indicated by `% 1`), and the `\lastxmark` will be the empty parameter from the last `\extramarks` command (indicated by `% 4`).

However, when the page break falls inside the block, the mark generated by `% 2` will be the last one on the first page. Therefore on that page `\lastxmark` will be 'Continued on next page. . . '. On the following pages, there are two possibilities: (1) when the block ends on that page the first mark will be `% 3`, therefore `\firstxmark` will be 'Continued. . . '; (2) the block ends at a later page, therefore it does not contribute any marks to that page, and the marks are 'inherited' from the last values of the previous page, i.e. those from `% 2`. On all of the pages after the block the values of `% 4` will be used, i.e. empty ones. This final `\extramarks{}{}` is to prevent the 'Continued. . . ' header to spill over to the following pages. Of course in real life you would leave out the numbers.

In case you want the last m_1 value or the first m_2 value, you can use the `\lastleftxmark` or `\firstrightxmark`, respectively. For symmetry reasons there are also commands `\firstleftxmark` (=`\firstxmark`), `\lastrightxmark` (=`\lastxmark`), `\topleftxmark` (=`\topxmark`) and `\toprightxmark`. The topmarks are basically the last-marks of the previous page.

The package also gives you the `\firstleftmark` and `\lastrightmark` commands that complement the standard LaTeX marks.

To stress the point that marks are the correct way to do this, let me give you a "solution" that will not work[12]:

```
\fancyhead[L]{Continued}
\fancyfoot[R]{Continued on next page\ldots}
Some text that may or may not cross a page boundary...
\fancyhead[L]{}
\fancyfoot[R]{}
```

You may be tempted to think that the first `\fancyhead` and `\fancyfoot` will be in effect when TEX breaks the page in the middle of the text, and the last ones when the page breaks after the text. This is not true as the whole paragraph (including the last definitions) will be processed before TEX considers the page break, so at the time of the page break the last definitions are effective, whether the page break occurs inside the text or outside of it. Putting a paragraph boundary between the text and the last definitions will not work either, because you don't want the first definitions to be in effect when TEX decides to break the page exactly at this boundary. Actually the marks mechanism was invented to get rid of these kinds of problems.

In the above example the text "Continued" appears in the page header. It may be nicer to put it in the margin. This can be easily accomplished by positioning it at a fixed place relative to the page header. In plain TEX you would use a concoction of `\hbox to 0pt`, `\vbox to 0pt`, `\hskip`,`\vskip`, `\hss` and `\vss` but fortunately LATEX's `picture` environment gives a much cleaner way to do this. In order not to disturb the normal header layout we put the text in a zero-sized `picture`. Generally this is the best way to position things on fixed places on the page. You can then also use the normal headings. See also section 27 for another example of this technique.

```
\fancyhead[L]{\setlength{\unitlength}{\baselineskip}%
\begin{picture}(0,0)
  \put(-2,-3){\makebox(0,0)[r]{\firstxmark}}
  \end{picture}\rightmark} % \rightmark = section title
```

This solution can of course also be used for the footer. Make sure you put the `picture` as the first thing in left-handside entries and last in right-handside ones.

Finally you may want to put "(Continued...)" in the *text* rather than in the header or the margin. Then you have to use the afterpage package. We also decide to make a separate environment for it.

```
\newenvironment{continued}{\par
  \extramarks{}{}%
  \extramarks{(Continued\ldots)}{Continued on next page\ldots}%
  \afterpage{\noindent\firstxmark\vspace{1ex}}%
  }{\extramarks{(Continued\ldots)}{}%
```

[12]Actually there is another way but it requires two LATEX passes: you can put `\label` commands before and after the text and compare the `\pageref`s.

```
  \extramarks{}{}\par
}
```

It is a bit dangerous to use `\firstxmark` outside the page layout routine, but apparently with `\afterpage` this works. If you would need the information further on in the page you must remember the state of the marks in your own variable. You can set this in one of the fancyhdr fields. For example if you want to add something *after* the broken piece of text you can use the following:

```
\newcommand{\mysaved}{}

\newenvironment{continued}{\par
  \extramarks{}{}%
  \extramarks{(Continued\ldots)}{Continued on next page\ldots}%
  }{\extramarks{(Continued\ldots)}{}%
  \extramarks{}{}\par\vspace{1ex}\mysaved}%
}
\fancyhead[L]{\leftmark}
\fancyhead[C]{\ifthenelse{\equal{\lastxmark}{}}
  {\gdef\mysaved{}}
  {\gdef\mysaved{\noindent[Continued from previous page]}}}
```

If you want to include one of the marks or other varying information in the saved text, you must use `\xdef` rather than `\gdef`.

26 A movie

If you put at each page on the same place a picture that slightly changes from page to page you can get a movie-like effect by flipping through the pages. You can create such a movie easily with fancyhdr. For simplicity we assume that we use a PDF-producing LaTeX (such as `pdflatex`) and each picture is in a PNG file called `pic`$\langle n \rangle$`.png` [13] where $\langle n \rangle$ is the page number and that we use the graphics or graphicx package. To put the movie in the righthandside bottom corner the following will work:

```
\fancyfoot[R]{\setlength{\unitlength}{1mm}
  \begin{picture}(0,0)
    \put(5,-20){\includegraphics[width=1cm]{pic\thepage}}
  \end{picture}}
```

If the document is two-sided, it would be better to put them only on the odd pages, by specifying `\fancyfoot[RO]`.

Notice that the `\unitlength` parameter should be set locally in the fancyhdr field in order to avoid unwanted interference with its value in the text.

[13] With `pdflatex` we could also use PDF or JPG pictures. With a DVI based `latex` we could use PS or EPS pictures. Or any other supported image format.

27 Thumb-indexes

Some railroad guides and expensive bibles have so called *thumb-indexes*, i.e. there are marks on the sides of the pages that indicate where the chapters are. You can create these by printing black blobs in the margin of the pages. The vertical position should be determined by the chapter number or some other counter. As the position is independent of the contents of the page, we print these blobs as part of the header in a zero-sized `picture` as described in the previous section.

Of course we have to take care of two-sided printing, and we may want to have an index page with all the blobs in the correct position. The solution requires some hand-tuning to get the blobs nicely spaced out vertically. For the application that I had there were 12 sections, so I made the blobs 18 mm apart, i.e. 9 mm blob separated by 9 mm whitespace. In order to avoid calculations they are set in a `picture` environment with the `\unitlength` set to 18 mm. Page numbers are set in the headers at the outer sides, and the blobs are attached to these. In this example the section numbers are used to position the blobs, but you can replace this with any numeric value. See figure 4 for the resulting overview page and figure 5 for the code.

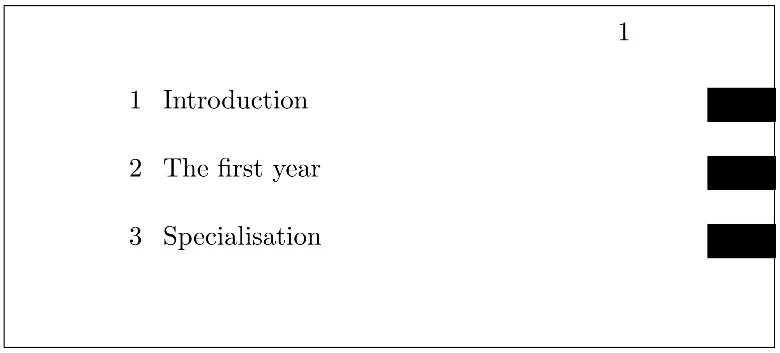


Figure 4: Thumb-index overview page

28 Float placement

Floats are page elements that float with respect to the rest of the document. Standard floats are tables and figures, but with the `float` package you can easily make new ones, like algorithms. Most of the time floats work satisfactory, but sometimes LaTeX seems too stubborn to do what you want. This section describes how you can influence LaTeX so that it will do most of the time what you want. There might, however be some pathological cases where it is impossible to convince LaTeX to do things your way. In the following we will use figures as an example but everything applies to other floats as well.

The most encountered problems with floats are:

```
\setlength{\unitlength}{18mm}
\newcommand{\blob}{\rule[-.2\unitlength]{2\unitlength}{.5\unitlength}}

\newcommand\rblob{\thepage
  \begin{picture}(0,0)
    \put(1,-\value{section}){\blob}
  \end{picture}}

\newcommand\lblob{%
  \begin{picture}(0,0)
    \put(-3,-\value{section}){\blob}
  \end{picture}%
  \thepage}

\pagestyle{fancy}
\fancyfoot[C]{}

\newcounter{line}
\newcommand{\secname}[1]{\addtocounter{line}{1}%
  \put(1,-\value{line}){\blob}
  \put(-7.5,-\value{line}){\Large \arabic{line}}
  \put(-7,-\value{line}){\Large #1}}

\newcommand{\overview}{\thepage
  \begin{picture}(0,0)
    \secname{Introduction}
    \secname{The first year}
    \secname{Specialisation}
...etc...
  \end{picture}}

\begin{document}
\fancyhead[R]{\overview}\mbox{}\newpage % This produces the overview page
\fancyhead[R]{} % Front matter may follow here
\clearpage
\fancyhead[RE]{\rightmark}
\fancyhead[RO]{\rblob}
\fancyhead[LE]{\lblob}
\fancyhead[LO]{{\leftmark}
...
```

Figure 5: Thumb-index code

1. You want a float at a certain position in the text, but LaTeX moves it, usually to the next page.
2. From a certain point, LaTeX moves all your floats to the end of the document or the end of a chapter.
3. LaTeX complains about "Too many floats".

In the first two cases you must first check if you have given the correct "placement" parameter to you float, e.g. `\begin{figure}[htp]` specifies that your figure may be placed either: Here (i.e. in the text position where the command is given), on the Top of a page (which may be the page where you put the command), or on a separate Page of floats. You could also have specified "b" for Bottom of the page. The order of the letters is insignificant, you cannot force LaTeX to try Bottom first and then Top by specifying `[bt]`.

If LaTeX doesn't put the float at the place where you expected it, it is usually caused by the following:

1. The float didn't fit on the page. In this case it has to move to the next page or even further. If you didn't specify either `[t]` or `[b]` in the position parameter, LaTeX must save it until it has enough for a page of floats. So don't specify only `[h]`. If you want to give LaTeX a chance to put the float on a page of floats, you must also specify "p".
2. The placement would violate the constraints imposed by LaTeX's float placement parameters. This is one of the most occurring causes and it can easily be corrected by changing the parameters. Here is a list of them:

Counters – change with `\setcounter`		
`topnumber`	max. number of floats at top of page	2
`bottomnumber`	max. number of floats at bottom of page	1
`totalnumber`	max. number of floats on a page	3
Other – change with `\renewcommand`		
`\topfraction`	max fraction of page for floats at top	0.7
`\bottomfraction`	max fraction of page for floats at bottom	0.3
`\textfraction`	min fraction of page for text	0.2
`\floatpagefraction`	min fraction of floatpage that should have floats	0.5

There are also some others for double column floats in two-column documents.

The values in the righthand column are the defaults for the standard LaTeX classes. Other classes could use different defaults. As you see with the default values a float will not be put in the bottom of a page if its height is more than 30% of the page height. So if you specify [hb] for a float which is taller it has to move to a float page. But if it is less than 50% of the page height it will have to wait until some more floats are given before a float page can be filled to satisfy the `\floatpagefraction` parameter. If you have this kind of behaviour you can easily adapt the parameters, e.g. with:

```
\renewcommand{\textfraction}{0.05}
\renewcommand{\topfraction}{0.95}
\renewcommand{\bottomfraction}{0.95}
\renewcommand{\floatpagefraction}{0.35}
\setcounter{totalnumber}{5}
```

You may want to be careful not to make `\floatpagefraction` too small, otherwise you may get too many small floatpages.

You can force LaTeX to ignore most of the parameters for one specific float occurrence by including an exclamation mark (!) in the placement parameters, e.g.

```
\begin{figure}[!htb]
```

Floats which contain a "`t`" in the position parameter could be placed before the place where they are referenced (but on the same page). This is normal behaviour for LaTeX but some people just don't like it. There are a number of ways to prevent this:

1. Of course deleting the "`t`" will help, but in general this is undesirable, as you may want the float to be placed at the top of the next page.
2. use the flafter package which causes floats never to be placed "backwards".
3. use the command `\suppressfloats[t]`. This command will cause floats for the top position *on this page* to be moved to the next page. This can also be done with `[b]` or without parameter for all floats on this page.

If in spite of all your attempts LaTeX still moves your floats to the end of the document or the end of a chapter, you can insert a `\clearpage` command. This will start a new page and insert all pending floats before continueing. If it is undesirable to have a pagebreak you can use the afterpage package and the following command:

```
\afterpage{\clearpage}
```

This will wait until the current page is finished and then flush all outstanding floats. In some pathological circumstances afterpage may give strange results, however.

Finally, if you want a float only at the place where you define it, without LaTeX moving it whatsoever, you can use the float package and give the command:

```
\restylefloat{figure}
```

in the preamble. Now you will be able to specify `[H]` as the position parameter, which will mean "HERE and only HERE". This may cause an unwanted page break however. If you want to avoid the unwanted pagebreak, i.e. let LaTeX move the float only if it doesn't fit on the page, then use the afterpage package with:

```
\afterpage{\clearpage \begin{figure}[H] ... \end{figure}}
```

Complaints from LaTeX about "Too many floats" are usually caused by one of the above problems: floats not being able to be placed and LaTeX collecting too many of them. The solutions given above, especially those with `\clearpage` in them will usually help. In some cases there really are too many floats, as LaTeX has a limited number of "boxes" to store the floats. The package morefloats can be used to increase this number. If you need still more then you must edit a private copy of this file, but even then there will be some limit that you cannot pass. Then your only resort will be to change your document.

29 Multipage Floats

LaTeX's floats cannot be split across pages. Sometimes, however, you want to have a table or figure that doesn't fit on one page. The easiest way is to split these into multiple table or figure environments, but this has a number of undesirable effects:

- Where do you split it? This is in general more difficult for tables than for figures.
- How do you keep them together?
- You don't want more than one entry in the list of figures/tables.

Although these problems are not fully solvable in all cases, here are a couple of suggestions:

29.1 Tables

For tables longer than a page you can use the longtable package. This package defines a `longtable` environment that is a kind of amalgamation of `table` and `tabular`. It has approximately the same syntax as the `tabular` environment, but it adds some features of `table`, like captions. Longtables will be automatically split when they don't fit on the page. And they will be entered in the list of tables when a caption is given. They will not float, however, and cannot be used inside a float environment. This could mean that another `table` environment, which was defined before the `longtable`, will float past it, and therefore the numbers may get out of order. Another problem could be that the `longtable` starts rather far down the page, which isn't a pleasant sight. If you want the `longtable` to start at the top of the page, the best thing to do is to include it in an `\afterpage` command (using the afterpage package). As a `longtable` is by definition large, it is best to put it in a separate file, and `\input` it in the `\afterpage` command:

```
\afterpage{\input{mytable}}
```

```
\afterpage{\clearpage\input{mytable}}
```

The last form has the additional advantage that most of the outstanding floats will be printed first.

29.2 Figures

There isn't an equivalent `longfigure` solution, so for figures you will usually have to split it yourself. In general this is less of a problem. However, the problem you get now is how to keep them together, i.e. how to get the parts on subsequent pages, and how to get a single entry in the list of figures.

You will have to split the figure into pieces and put each part in a separate `figure` environment. To keep them together it is best to use only the `[p]` placement, so that they will be put on floatpages. As they are bigger than a page this is appropriate. The first part would then get a `\caption`, the subsequent parts would be used without a caption. If you want to add a caption-like text, enter it as normal text rather than a `\caption`, so that it will not be entered in the list of figures. It may also be desirable to issue a `\clearpage` first, just like we did for the `longtable`, and to encapsulate this in the `\afterpage` command. E.g.

```
\afterpage{\clearpage\input{myfigure}}
```

where `myfigure.tex` contains:

```
\begin{figure}[p]
\includegraphics{myfig1.eps}
\caption{This is a multipage figure}
\label{fig:xxx}
\end{figure}
\begin{figure}[p]
\includegraphics{myfig2.eps}
\begin{center}
Figure~\ref{fig:xxx} (continued)
\end{center}
\end{figure}
```

You have to make sure that the last part is big enough, otherwise LaTeX could decide to postpone it until it has collected some more floats. This can be done either by making the figure big enough (e.g. by adding some `\vspace`), or by tweaking the `\floatpagefraction` parameter.

If you want your multipage figure to start at a lefthand-side (even-numbered) page you can use a test in the `\afterpage` command (using the ifthen package):

```
\afterpage{\clearpage
\ifthenelse{\isodd{\value{page}}}{\afterpage{\input{myfigure}}} % odd page
    {\input{myfigure}}}} % even page
```

If there are too many floats on the skipped page, this may still fail to start your multipage figure on an even page, however.

30 Deprecated commands

This section contains the description of deprecated commands. These were parts of the original implementation of fancyheadings. They continue to work for compatibility reasons, but it is recommended not to use them anymore. This description is given so that you know what they mean and how to convert them to the standard commands. To be honest, I use these sometimes myself in quick examples, because `\lhead` is less typing than `\fancyhead[L]`.

These commands for specifying the header or footer fields and their translation to the modern commands are given in table 1.

`\lhead{xx}`	`\fancyhead[L]{xx}`	
`\lhead[xx]{yy}`	`\fancyhead[LE]{xx}`	`\fancyhead[LO]{yy}`
`\chead{xx}`	`\fancyhead[C]{xx}`	
`\chead[xx]{yy}`	`\fancyhead[CE]{xx}`	`\fancyhead[CO]{yy}`
`\rhead{xx}`	`\fancyhead[R]{xx}`	
`\rhead[xx]{yy}`	`\fancyhead[RE]{xx}`	`\fancyhead[RO]{yy}`
`\lfoot{xx}`	`\fancyfoot[L]{xx}`	
`\lfoot[xx]{yy}`	`\fancyfoot[LE]{xx}`	`\fancyfoot[LO]{yy}`
`\cfoot{xx}`	`\fancyfoot[C]{xx}`	
`\cfoot[xx]{yy}`	`\fancyfoot[CE]{xx}`	`\fancyfoot[CO]{yy}`
`\rfoot{xx}`	`\fancyfoot[R]{xx}`	
`\rfoot[xx]{yy}`	`\fancyfoot[RE]{xx}`	`\fancyfoot[RO]{yy}`

Table 1: Deprecated commands and their translation

As you see, if there is an optional parameter, this one applies to the even pages, whereas the required parameter applies to the odd pages. Of course this only works if the `twoside` option is given in the documentclass. If there is no optional parameter, the required parameter applies to both even and odd pages.

There was also a special pagestyle `fancyplain` that could be used to define both the pagestyle `fancy` and to redefine the pagestyle `plain` at the same time. In order to use that you say

```
\pagestyle{fancyplain}}
```

and then in the headers/footers you use for example:

```
\fancyhead[L]{\fancyplain{value for ‘plain’ page}{value for other pages}}}
```

The `\fancyplain` command is only useful within the pagestyle `fancyplain`. Nowadays you would just redefine pagestyle `plain` with the `\fancypagestyle{plain}{xxxx}` command (see section 11).

There are also `\plainheadrulewidth` and `\plainfootrulewidth` commands to define the values of `\headrulewidth` and `\footrulewidth` to be used on ‘`plain`’ pages. This also only works with the pagestyle `fancyplain`, not when you redefine pagestyle `plain` with the `\fancypagestyle` command.

31 Contact information

Pieter van Oostrum
E-mail: pieter@vanoostrum.org
WWW: http://pieter.vanoostrum.org

The source code can be found on Github:
https://github.com/pietvo/fancyhdr
Bugs and suggestions for improvements can be reported at
https://github.com/pietvo/fancyhdr/issues
Example files can be found at
https://github.com/pietvo/fancyhdr/tree/Examples

32 Version information

- Version 1.0. March 11, 2003. This is the version that was distributed for a long time on CTAN. Version history before this has been lost.
- Version 2.0. August 27, 2016:
  - Removed references to fixmarks.sty as that is no longer used.
  - References to older LaTeX versions removed.
  - Removed obsolete source code of extramarks.sty
  - Changed font commands to `\textbf` and `\textsl`.
  - Added description of the `\fancy...offset` commands.
  - Added various `\...xmark` commands from extramarks.sty.
  - Various corrections applied.
  - Updated contact information.
  - Added Version information. :)
- Version 2.1. August 28, 2016
  - Explain what the top-marks are.
- Version 2.1. Sept. 6, 2016
  - Add `\string` to special indexing commands to get a neater index file.
  - Add a decorative headrule example.
- Version 3.9, October 13, 2016.

- Documentation integrated in `fancyhdr.dtx`.
- Version number unified with fancyhdr.sty.
- All deprecated commands moved to a separate section (30).
- Documentation expanded.

- Version 3.9a, June 30, 2017.
  - Updated contact information.
  - Restore `\newtoks\@temptokenb`
- Version 3.10, Januari 25, 2019
  - Distribution based on fancydhr.dtx.
  - Use `\f@nch@ifundefined` instead of `\ifx` or `\@ifundefined`.
  - Replace `\def` with `\newcommand` in several places.
  - Don't use `\global\setlength`.
  - Put `\footrule` in a `\vbox` to accommodate for flexible footrules, and then `\unvbox` that. Move the `\footruleskip` vertical space outside of the definition of `\footrule`.

32.1 Changes in version 4

Version 4 is a significant rewrite of the package. It also introduces a number of new features.

- Version 4.0, March 15, 2019–Jan 04, 2021
  - Options introduced on the `\usepackage` command.
  - The check whether the header or footer fits in `\headheight` and `\footskip`, respectively, no longer adjusts these values for the following pages. This appeared to be too confusing. However, when the package option `compatV3` is given, the old behaviour is kept.
    The `nocheck` option now eliminates these checks completely, on your own risk. (See section 17 on page 22.)
  - Eliminated global definitions. All definitions are now local. The `\global` case was originally so that you could do definitions in a group and they would be applied globally. This was a mistake. If you make them locally they should stay local. And it caused problems with switching page styles, because then the global style would be changed, which you generally don't want.
    However, when the package option `compatV3` is given, the old behaviour is kept. (See section 13.)
  - The page style `fancydefault`.
  - The `\headruleskip` parameter.

  - The `\fancyheadinit`, `\fancyfootinit`, and `\fancyhfinit` commands.

    Note: The following changes were mostly copied from the `nccfancyhdr` package by Alexander I. Rozhenko.
  - The `\fancycenter` command (section 9).
  - The `headings` and `myheadings` package options (see section 13).
  - The `\fancypagestyle` command has an optional parameter `[`⟨*base-style*⟩`]`.

- Version 4.0.1, Jan 28, 2021
  - Some documentation corrections, especially in sections 25 and 26.

Part III
Questions & Answers

This part contains answers to questions that have been emailed to me, or have been asked at various internet forums, and don't have a logical place in the other documentation. It is expected to grow gradually.

33 Large chapter/section titles

Sometimes a chapter or section title is too large to fit in the header or footer. It may take more than one line in the header/footer, or it may overwrite other parts. How can we shorten these titles in the header/footer without changing the actual title?

Here is an example:

```
\fancyhead[LE,RO]{\nouppercase{\rightmark}} % Section title
\fancyhead[LO,RE]{\nouppercase{\leftmark}} % Chapter title
\fancyfoot[C]{\thepage}
 . . .
\chapter{This is a very long chapter title}
 . . .
\section{This is a very long section title that will not fit in the header}
 . . .
```

With these settings the header will come out as:

Chapter 1. This is a very long chapter title 1.2. This is a very long section title that will not fit in the header

which isn't very nice. There are basically three options to solve this problem.

33.1 Using optional arguments

As we have seen in section 15, the header info comes from the marks. So if we want the text in the header to be smaller we have to supply smaller marks. This can be done by giving these as optional arguments in the `\chapter` and `\section` commands.[14]

```
\chapter[Short chapter title]{This is a very long chapter title}
 . . .
\section[Short section title]
        {This is a very long section title that will not fit in the header}
```

The short titles will now appear in the header. However, these will also appear in the table of contents. If that is what you want than you are ready. But if you want to use the long titles in the table of contents, you have to use some trickery. In particular you have to supply the marks yourself.

33.2 Using explicit marks

First we show how you can supply a different value for the chapter title in the heading, because this is the easiest. Remember from section 15 that this mark is defined by calling `\chaptermark`. Also, because it is used as `\leftmark`, the last value of this mark on the page is used. So we can easily overrule the value that is supplied by the `\chapter` command, by supplying an additional `\chaptermark` command after the `\chapter` command, like this:

```
\chapter{This is a very long chapter title that does not fit in the header}
\chaptermark{This is a not so long chapter title}
```

For the section titles the situation is more complicated. Here we use the `\rightmark`, which uses the first mark of its kind on the page. So you might think putting a `\sectionmark` before the `\section` command would be the solution. Unfortunately, it is not that simple. In many cases, this will work, but not when there is a page break just before the section title, because in that case the `\sectionmark` will stay behind on the previous page. However, we can put the `\sectionmark` inside the argument of the `\section` command. Because LaTeX first typesets the title (which will execute the included `\sectionmark` command), and after that executes its own `\sectionmark`, our `\sectionmark` will be the first. But there is one case in which this fails: if the next page does not have any `\sectionmark` commands, it will inherit the **last** mark from the page before it, which will be the long title. To correct this we must also give an additional `\sectionmark` with the short title **after** the `\section` command.

As if this isn't enough, there is still a problem with this setup. Our section title is not only used to typeset the title in the text, but it is also included in the

[14] At least in the `book` and `report` documentclasses. In the `article` class this would be the `\section` and `\subsection` commands.

table of contents. But the table of contents does not accept a `\sectionmark` in its title. It will generate an ugly error message. To prevent this we must give the long title (that we want to appear in the table of contents) also as the optional argument to the `\section` command. Of course this will also generate a mark for the header, but this will be overruled by our included `\sectionmark` commands

So the complete code would be:

```
\section[Long title]{Long title\sectionmark{Short title}}
\sectionmark{Short title}
```

To avoid all the repetitions, it is better to make a macro:

```
\newcommand{\Section}[2]{\section[#1]{#1\sectionmark{#2}}\sectionmark{#2}}
  . . .
\Section{Text title}{Header title}
```

And if you want to use yet a different text in the table of contents, you can make a macro with three parameters. The third parameter is the text to be put in the table of contents. We use this parameter as the optional argument for the `\section` command.

```
\newcommand{\Sectioniii}[3]{\section[#3]{#1\sectionmark{#2}}\sectionmark{#2}}
  . . .
\Sectioniii{Text title}{Header title}{TOC title}
```

Please note that if you use the `article` class, instead of `\chaptermark` and `\sectionmark`, you would probably use `\sectionmark\subsectionmark`.

33.3 Using automatic truncation

For this solution we use the truncate package by Donald Arseneau. This has a `\truncate` command that truncates a text to a maximum size, when it exceeds that size. We put both headers in `\truncate` to limit it to half the `\headwidth`. Of course it is also possible to make asymmetric arrangements.

```
\usepackage[fit]{truncate}
\fancyhead[LE,RO]{\nouppercase{\truncate{0.5\headwidth}{\rightmark}}}
\fancyhead[LO,RE]{\nouppercase{\truncate{0.5\headwidth}{\leftmark}}}
```

We don't have to make any changes to the chaper and section titles because `\truncate` will take care of this. This arrangement gives the following header when both titles are too big, like in the example above:

Chapter 1. This is a very long chapter . . .   1.2. This is a very long section title . . .

Note that we have used the `[fit]` option of the truncate package. Otherwise the right header will not be right aligned, but it will start at halfway the header.

Note also that, as each part can occupy half of the available width, they could theoretically touch each other. This can be prevented by making the widths slightly smaller. And when there is only one title in the header, you can make the width equal to or slightly smaller than `\headwidth`.

34 I lost my chapter/section titles

Some time ago I got a question like this (edited to get the essentials):

"I redefined the `\pagestyle{fancy}` to get my own kind of headings. Also, I redefined the `\chaptermark`. I need the `fancy` style from chapter 1 and on (mainmatter part), but, until the Introduction chapter (that I included into the frontmatter part) I need the `myheadings` style.

When I set the `myheadings` style into the frontmatter the `fancy` style doesn't show the chapter title any more.

What can I do in order to reestablish the right behavior of the `fancy` style?"

The solution to this problem is actually very simple. The pagestyle `myheadings` (as well as `headings`) redefines the `\chaptermark` and `\sectionmark`, so when you return to pagestyle `fancy`, the definitions you had given before (or the ones that fancyhdr provided) are lost. You just have to repeat them at the point where you switch back to pagestyle `fancy`.

```
\begin{document}
\frontmatter
\pagestyle{myheadings}
 . . .
\mainmatter
\pagestyle{fancy}
\renewcommand{\chaptermark}[1]{....}
```

Part IV Implementation

35 fancyhdr.sty

<*fancyhdr>

`\if@nch@mpty` This macro tests if its argument is empty.

```
\newcommand\if@nch@mpty[1]{\def\temp@a{#1}\ifx\temp@a\@empty}
```

`\iff@nch@check` Boolean for the `nocheck` option.

```
\newif\iff@nch@check
\f@nch@checktrue
\DeclareOption{nocheck}{%
  \f@nch@checkfalse
}
```

`\iff@nch@compatViii` Define `\iff@nch@compatViii` to track the `compatV3` option.

```
\newif\iff@nch@compatViii
```

`\f@nch@gbl` Initialise `\f@nch@gbl` to do nothing (except with the `compatV3` option).

```
\let\f@nch@gbl\relax
\DeclareOption{compatV3}{%
  \let\f@nch@gbl\global
  \f@nch@compatViiitrue
}
```

`\f@nch@def` This macro defines another macro (usually a header or footer field). Depending on the value of `\f@nch@gbl` the definition will be global or local. Default it is always local. But with the `compatV3` option it is `\global` in the normal definitions, and local in `\fancypagestyle`. The `\global` case is now considered a bug (or at least undesirable).

If the value (argument 2) is empty, a `\leavevmode` will be substituted. If it is not empty, a `\strut` will be added.

```
\newcommand\f@nch@def[2]{\if@nch@mpty{#2}\f@nch@gbl\def#1{\leavevmode}\else
                                   \f@nch@gbl\def#1{#2\strut}\fi}
```

`\f@nch@ifundefined` This macro tests if a command is undefined. Older versions of fancyhdr used `\@ifundefined`, but this had an undesired side effect in the original LaTeX (the command was made equal to `\relax` if it was undefined). Another way was `\ifx\thecommand\undefined ...` or `\ifx\thecommand\@undefined ...` but that could conflict with packages that use the `\@ifundefined` method. LaTeX versions later than 2018 have a definition of `\@ifundefined` that avoids these problems, but not everybody may have such a version installed. Therefore we define our own version `\f@nch@ifundefined`. This definition is copied from the tocloft package by Peter Wilson and Will Robertson.

```
\newcommand{\f@nch@ifundefined}[1]{%
  \begingroup\expandafter\expandafter\expandafter\endgroup
  \expandafter\ifx\csname #1\endcsname\relax
    \expandafter\@firstoftwo
  \else
    \expandafter\@secondoftwo
    \fi}
% \end{macrocode}
% \end{macro}
%
% \changes{fancyhdr v4.0}{2019/03/17}{Added \texttt{headings} and
% \texttt{myheadings} options.}
% Standard styles are redefined optionally. These definitions are
% borrowed from the \Package{nccfancyhdr} package by by Alexander I. Rozhenko.
%
% \begin{macro}{\ps@myheadings}
% The redefinition of the \texttt{myheadings} style is conditional.
% We test the existence of the \cs{chapter} command and redefine the
% style accordingly.
%
% \OPTindex{myheadings}\PSindex{myheadings}
%    \begin{macrocode}
\DeclareOption{myheadings}{%
  \f@nch@ifundefined{chapter}{%
```

An article-like class without chapters:

```
    \def\ps@myheadings{\ps@f@nch@fancyproto \let\@mkboth\@gobbletwo
      \fancyhf{}
      \fancyhead[LE,RO]{\thepage}%
      \fancyhead[RE]{\slshape\leftmark}%
      \fancyhead[LO]{\slshape\rightmark}%
      \let\sectionmark\@gobble
      \let\subsectionmark\@gobble
    }%
  }%
```

A book/report-like class with chapters:

```
  {\def\ps@myheadings{\ps@f@nch@fancyproto \let\@mkboth\@gobbletwo
      \fancyhf{}
      \fancyhead[LE,RO]{\thepage}%
      \fancyhead[RE]{\slshape\leftmark}%
      \fancyhead[LO]{\slshape\rightmark}%
      \let\chaptermark\@gobble
      \let\sectionmark\@gobble
    }%
  }%
}
```

\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}{%
  \f@nch@ifundefined{chapter}{%
    \if@twoside
```

An article in two-side mode:

```
      \def\ps@headings{\ps@f@nch@fancyproto \let\@mkboth\markboth
        \fancyhf{}
        \fancyhead[LE,RO]{\thepage}%
        \fancyhead[RE]{\slshape\leftmark}%
        \fancyhead[LO]{\slshape\rightmark}%
        \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@f@nch@fancyproto \let\@mkboth\markboth
        \fancyhf{}
        \fancyhead[LE,RO]{\thepage}%
        \fancyhead[RE]{\slshape\leftmark}%
        \fancyhead[LO]{\slshape\rightmark}%
        \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@f@nch@fancyproto \let\@mkboth\markboth
        \fancyhf{}
        \fancyhead[LE,RO]{\thepage}%
        \fancyhead[RE]{\slshape\leftmark}%
        \fancyhead[LO]{\slshape\rightmark}%
        \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@f@nch@fancyproto \let\@mkboth\markboth
        \fancyhf{}
        \fancyhead[LE,RO]{\thepage}%
        \fancyhead[RE]{\slshape\leftmark}%
        \fancyhead[LO]{\slshape\rightmark}%
        \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
  }%
}
```

Process the options.

```
\ProcessOptions*
```

`\f@nch@errmsg` This macro generates an error message.

```
\newcommand\f@nch@errmsg[1]{%
  \f@nch@ifundefined{PackageError}{\errmessage{#1}}{\PackageError{fancyhdr}{#1}{}}}
```

`\f@nch@warning` This macro generates a warning.

```
\newcommand\f@nch@warning[1]{%
  \f@nch@ifundefined{PackageWarning}{\errmessage{#1}}{\PackageWarning{fancyhdr}{#1}{
```

`\f@nch@forc` Usage: `\f@nch@forc \var {charstring}{body}`.
Execute the body for each character in `charstring` bound to `\var`. This is similar to LaTeX's `\@tfor`, but it expands the `charstring`.

```
% \changes{fancyhdr v3.10}{2019/01/25}{Use \cs{newcommand} instead of \cs{def}.}
\newcommand{\f@nch@forc}[3]{\expandafter\f@nchf@rc\expandafter#1\expandafter{#2}{#3}
\newcommand{\f@nchf@rc}[3]{\def\temp@ty{#2}\ifx\@empty\temp@ty\else
                                    \f@nch@rc#1#2\f@nch@rc{#3}\fi}
\def\f@nch@rc#1#2#3\f@nch@rc#4{\def#1{#2}#4\f@nchf@rc#1{#3}{#4}}
```

`\f@nch@for` Usage: `\f@nch@for\var{list}{body}`
Execute the body for each element of the list, bound to `\var`. List elements are separated by commas. This is like LaTeX's `\@for` but an empty list is treated as a list with an empty element.

```
\newcommand{\f@nch@for}[3]{\edef\@fortmp{#2}%
  \expandafter\@forloop#2,\@nil,\@nil\@@#1{#3}}
```

`\f@nch@default` Usage: `\f@nch@default \var{defaults}{argument}`
Sets `\var` to the characters from `defaults` appearing in `argument`, or to `defaults` if it would be empty. All characters are lowercased first.

```
\newcommand\f@nch@default[3]{%
  \edef\temp@a{\lowercase{\edef\noexpand\temp@a{#3}}}\temp@a \def#1{}%
  \f@nch@forc\tmpf@ra{#2}%
  {\expandafter\f@nch@ifin\tmpf@ra\temp@a{\edef#1{#1\tmpf@ra}}{}}%
  \ifx\@empty#1\def#1{#2}\fi}
```

`\f@nch@ifin` Usage: `\f@nch@ifin` ⟨*char*⟩ ⟨*set*⟩ ⟨*truecase*⟩ ⟨*falsecase*⟩
If ⟨*char*⟩ is in ⟨*set*⟩, then ⟨*truecase*⟩ else ⟨*falsecase*⟩.

```
\newcommand{\f@nch@ifin}[4]{%
  \edef\temp@a{#2}\def\temp@b##1#1##2\temp@b{\def\temp@b{##1}}%
  \expandafter\temp@b#2#1\temp@b\ifx\temp@a\temp@b #4\else #3\fi}
```

`\fancyhead`
`\fancyfoot`
`\fancyhf`
These are the principal user macros. Pick up the parameters, and supply an 'h' (`\fancyhead`) or 'f' (`\fancyfoot`).

```
\newcommand{\fancyhead}[2][]{\f@nch@fancyhf\fancyhead h[#1]{#2}}%
\newcommand{\fancyfoot}[2][]{\f@nch@fancyhf\fancyfoot f[#1]{#2}}%
\newcommand{\fancyhf}[2][]{\f@nch@fancyhf\fancyhf {}[#1]{#2}}%
```

`\fancyheadoffset`
`\fancyfootoffset`
`\fancyhfoffset`
The commands for offsets. Pick up the parameters, and supply an 'h' (`\fancyheadoffset`) or 'f' (`\fancyfootoffset`).

```
\newcommand{\fancyheadoffset}[2][]{\f@nch@fancyhfoffs\fancyheadoffset h[#1]{#2}}%
\newcommand{\fancyfootoffset}[2][]{\f@nch@fancyhfoffs\fancyfootoffset f[#1]{#2}}%
\newcommand{\fancyhfoffset}[2][]{\f@nch@fancyhfoffs\fancyhfoffset {}[#1]{#2}}%
```

`\f@nch@fancyhf` This macro interprets the parameters for the headers and footers.
Parameters:
(1) The user command that was used (like `\fancyhead`). This is used for errors/warnings.
(2) `h` (for `\fancyhead`), `f` (for `\fancyfoot`), or `{}` (for `\fancyhf`).
(3) The optional parameter that was given to these commands (default `[]`).
(4) The required parameter that was given to these commands.
The header and footer fields are stored in command sequences with names of the form: `\f@nch@`⟨*x*⟩⟨*y*⟩⟨*z*⟩ with ⟨*x*⟩ from `[eo]`, ⟨*y*⟩ from `[lcr]` and ⟨*z*⟩ from `[hf]`.

```
\def\f@nch@fancyhf#1#2[#3]#4{%
  \def\temp@c{}%
  \f@nch@forc\tmpf@ra{#3}%
  {\expandafter\f@nch@ifin\tmpf@ra{eolcrhf,EOLCRHF}%
    {}{\edef\temp@c{\temp@c\tmpf@ra}}}%
  \ifx\@empty\temp@c\else \f@nch@errmsg{Illegal char `\temp@c' in
    \string#1 argument: [#3]}%
  \fi \f@nch@for\temp@c{#3}%
  {\f@nch@default\f@nch@@eo{eo}\temp@c \if@twoside\else \if\f@nch@@eo
    e\f@nch@warning {\string#1's `E' option without twoside option is
```

```
      useless}\fi\fi \f@nch@default\f@nch@@lcr{lcr}\temp@c
    \f@nch@default\f@nch@@hf{hf}{#2\temp@c}%
    \f@nch@forc\f@nch@eo\f@nch@@eo
        {\f@nch@forc\f@nch@lcr\f@nch@@lcr
          {\f@nch@forc\f@nch@hf\f@nch@@hf
            {\expandafter\f@nch@def\csname
              f@nch@\f@nch@eo\f@nch@lcr\f@nch@hf\endcsname {#4}}}}}}
```

`\f@nch@fancyhfoffs` This macro interprets the parameters for the header and footer offsets.
Parameters:
(1) The user command that was used (like `\fancyheadoffset`). This is used for errors/warnings.
(2) `h` (for `\fancyheadoffset`), `f` (for `\fancyfootoffset`), or `{}` (for `\fancyhfoffset`).
(3) The optional parameter that was given to these commands (default `[]`).
(4) The required parameter that was given to these commands.
The header and footer offsets are stored in command sequences with names of the form: `\f@nch@O@`$\langle x\rangle\langle y\rangle\langle z\rangle$ with $\langle x\rangle$ from `[eo]`, $\langle y\rangle$ from `[lr]` and $\langle z\rangle$ from `[hf]`.

```
\def\f@nch@fancyhfoffs#1#2[#3]#4{%
  \def\temp@c{}%
  \f@nch@forc\tmpf@ra{#3}%
  {\expandafter\f@nch@ifin\tmpf@ra{eolrhf,EOLRHF}%
    {}{\edef\temp@c{\temp@c\tmpf@ra}}}%
  \ifx\@empty\temp@c\else \f@nch@errmsg{Illegal char `\temp@c' in
    \string#1 argument: [#3]}%
  \fi \f@nch@for\temp@c{#3}%
  {\f@nch@default\f@nch@@eo{eo}\temp@c \if@twoside\else \if\f@nch@@eo
    e\f@nch@warning {\string#1's `E' option without twoside option is
      useless}\fi\fi \f@nch@default\f@nch@@lcr{lr}\temp@c
    \f@nch@default\f@nch@@hf{hf}{#2\temp@c}%
    \f@nch@forc\f@nch@eo\f@nch@@eo
        {\f@nch@forc\f@nch@lcr\f@nch@@lcr
          {\f@nch@forc\f@nch@hf\f@nch@@hf
            {\expandafter\setlength\csname
              f@nch@O@\f@nch@eo\f@nch@lcr\f@nch@hf\endcsname {#4}}}}}%
  \f@nch@setoffs}
```

`\lhead` `\chead` `\rhead` `\lfoot` `\cfoot` `\rfoot` Fancyheadings version 1 commands. These are deprecated, but they continue to work for compatibility reasons. They have an optional parameter that is used as the value for even pages in a two-sided document. If this is not given (or if the document is not two-sided) the required parameter is used for both even and odd pages. Therefore the default value for the optional parameter is the required parameter. It is not possible to express this directly in the definition. Therefore we use a trick. Both parameters are store in a macro. For example for `\lhead` the parameter for even pages is stored in `\f@nch@elh`, and the one for odd pages in `\f@nch@olh`. For the others it is similar, just replace the `l` with `c` or `r`, and the `h` with `f`. In the body of the macro we first store the required parameter in

\f@nch@olh, and we use this macro as default for the optional parameter. The optional parameter is then stored in \f@nch@elh. The order of the assignments is therefore important.

```
\newcommand{\lhead}[2][\f@nch@olh]%
                    {\f@nch@def\f@nch@olh{#2}\f@nch@def\f@nch@elh{#1}}
\newcommand{\chead}[2][\f@nch@och]%
                    {\f@nch@def\f@nch@och{#2}\f@nch@def\f@nch@ech{#1}}
\newcommand{\rhead}[2][\f@nch@orh]%
                    {\f@nch@def\f@nch@orh{#2}\f@nch@def\f@nch@erh{#1}}
\newcommand{\lfoot}[2][\f@nch@olf]%
                    {\f@nch@def\f@nch@olf{#2}\f@nch@def\f@nch@elf{#1}}
\newcommand{\cfoot}[2][\f@nch@ocf]%
                    {\f@nch@def\f@nch@ocf{#2}\f@nch@def\f@nch@ecf{#1}}
\newcommand{\rfoot}[2][\f@nch@orf]%
                    {\f@nch@def\f@nch@orf{#2}\f@nch@def\f@nch@erf{#1}}
```

\f@nch@headwidth

Length parameter to be used for \headwidth. We use this rather than defining \headwidth as a length paramater directly to protect ourself to someone saying: \let\headwidth\textwidth.

```
\newlength{\f@nch@headwidth} \let\headwidth\f@nch@headwidth
```

\f@nch@O@elh
\f@nch@O@erh
\f@nch@O@olh
\f@nch@O@orh
\f@nch@O@elf
\f@nch@O@erf
\f@nch@O@olf
\f@nch@O@orf

Length parameters for the offsets.

```
\newlength{\f@nch@O@elh}
\newlength{\f@nch@O@erh}
\newlength{\f@nch@O@olh}
\newlength{\f@nch@O@orh}
\newlength{\f@nch@O@elf}
\newlength{\f@nch@O@erf}
\newlength{\f@nch@O@olf}
\newlength{\f@nch@O@orf}
```

\headrulewidth
\footrulewidth

```
\newcommand{\headrulewidth}{0.4pt}
\newcommand{\footrulewidth}{0pt}
```

\headruleskip

Don't define \headruleskip if it is already defined.

```
\f@nch@ifundefined{headruleskip}%
      {\newcommand{\headruleskip}{0pt}}{}
```

\footruleskip

Memoir also defines \footruleskip. Don't define \footruleskip if it is already defined.

```
\f@nch@ifundefined{footruleskip}%
      {\newcommand{\footruleskip}{.3\normalbaselineskip}}{}
```

\plainheadrulewidth
\plainfootrulewidth

Fancyplain stuff shouldn't be used anymore (rather \fancypagestyle{plain} should be used), but we keep it for compatibility reasons.

```
\newcommand{\plainheadrulewidth}{0pt}
\newcommand{\plainfootrulewidth}{0pt}
```

`\if@fancyplain` Boolean for the implementation of `\fancyplain`

```
\newif\if@fancyplain \@fancyplainfalse
```

`\fancyplain` Deprecated macro

```
\def\fancyplain#1#2{\if@fancyplain#1\else#2\fi}
```

`\headwidth` Initialise `\headwidth` with a magic constant.

```
\headwidth=-123456789sp
```

`\f@nch@raggedleft` `\f@nch@raggedright` `\f@nch@centering` `\f@nch@everypar` Save the standard definitions of `\raggedleft`, `\raggedright`, `\centering` and `\everypar` so that we can reset them when we are typesetting the headers and footers. Some packages change these to incompatible values.

```
\let\f@nch@raggedleft\raggedleft
\let\f@nch@raggedright\raggedright
\let\f@nch@centering\centering
\let\f@nch@everypar\everypar
```

`\f@nch@reset` Command to reset various things in the headers: a.o. single spacing (taken from setspace.sty) and the catcode of `\endlinechar` (so that epsf files in the header work if a verbatim crosses a page boundary) It also defines a `\nouppercase` command that disables `\uppercase` and `\Makeuppercase`. It can only be used in the headers and footers. Set `\hsize` to `\headwidth` (this helps for multicol); reset `\\`, `\raggedleft`, `\raggedright` and `\centering` to their default values (for tabu), and `\everypar` to empty.
The font is reset to `\normalfont`.

```
\def\f@nch@reset{\f@nch@everypar{}\restorecr\endlinechar=13
  \let\\\@normalcr \let\raggedleft\f@nch@raggedleft
  \let\raggedright\f@nch@raggedright \let\centering\f@nch@centering
  \def\baselinestretch{1}%
  \hsize=\headwidth
  \def\nouppercase##1{{\let\uppercase\relax\let\MakeUppercase\relax
      \expandafter\let\csname MakeUppercase \endcsname\relax##1}}%
  \f@nch@ifundefined{@newbaseline} % NFSS not present; 2.09 or 2e
  {\f@nch@ifundefined{@normalsize} {\normalsize} % for ucthesis.cls
   {\@normalsize}}%
  {\@newbaseline}% NFSS (2.09) present
  }
```

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

```
\newcommand*{\fancycenter}[1][1em]{%
  \@ifnextchar[{\f@nch@center{#1}}{\f@nch@center{#1}[3]}%
}
\def\f@nch@center#1[#2]#3#4#5{%
```

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

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

All 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 lengths of ⟨*left-mark*⟩ and ⟨*right-mark*⟩. We calculate which mark is shorter and increase the natural size of the corresponding register by the difference between their lengths.

```
  \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` have been calculated. Put everything in the box.

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

`\fancyheadinit` This macro can be used to define initialisation code that will be run before the construction of the header. It can for example set the color or the font, or change `\headrulewidth` or `\headruleskip`. It cannot make global changes, just changes for the header.

`\f@nch@headinit` Storage for the header initialisation code.

```
\newcommand{\f@nch@headinit}{}
\newcommand{\fancyheadinit}[1]{%
  \def\f@nch@headinit{#1}%
}
```

`\fancyfootinit` This macro can be used to define initialisation code that will be run before the construction of the footer. It can for example set the color or the font, or change `\footrulewidth` or `\footruleskip`. It cannot make global changes, just changes for the footer.

`\f@nch@footinit` Storage for the footer initialisation code.

```
\newcommand{\f@nch@footinit}{}
\newcommand{\fancyfootinit}[1]{%
  \def\f@nch@footinit{#1}%
}
```

`\fancyhfinit` This macro sets both the header and the footer initialisation codes to the same value.

```
\newcommand{\fancyhfinit}[1]{%
  \def\f@nch@headinit{#1}%
  \def\f@nch@footinit{#1}%
}
```

`\f@nch@vbox` Make a `\vbox` with the header or footer. Check whether there is enough space and give a warning if not. Use box 0 as a temp box and dimen 0 as temp dimen. This can be done, because this code will always be used inside another box, and therefore the changes are local.
Parameter 1 is `\headheight` or `\footskip`, respectively.
Parameter 2 is the contents of the box.

```
\newcommand\f@nch@vbox[2]{%
  \setbox0\vbox{#2}%
  \ifdim\ht0>#1\relax
    \iff@nch@check
      \dimen0=#1\advance\dimen0-\ht0
      \f@nch@warning{%
        \string#1 is too small (\the#1): \MessageBreak
        Make it at least \the\ht0, for example:\MessageBreak
        \string\setlength{\string#1}{\the\ht0}%
        \iff@nch@compatViii .\MessageBreak
        We now make it that large for the rest of the document.\MessageBreak
        This may cause the page layout to be inconsistent, however
        \fi
        \ifx#1\headheight .\MessageBreak
          You might also make \topmargin smaller to compensate:\MessageBreak
          \string\addtolength{\string\topmargin}{\the\dimen0}%
        \fi
      \@gobble
    }%
  \fi
  \iff@nch@compatViii
    \dimen0=#1\relax
    \global#1=\ht0\relax
    \ht0=\dimen0 %
  \else
    \ht0=#1%
  \fi
\fi
\box0}
```

`\f@nch@head` Put together a header or footer given the left, center and right text, fillers at left and right and a rule. The `\xlap` commands put the text into an hbox of zero size, so overlapping text does not generate an errormessage.
These macros have 5 parameters:
1. LEFTSIDE BEARING. This determines at which side the header will stick out. When `\fancyhfoffset` is used this calculates `\headwidth`, otherwise it is `\hss` or `\relax` (after expansion).
2. `\f@nch@olh`, `\f@nch@elh`, `\f@nch@olf` or `\f@nch@elf`. This is the left component.
3. `\f@nch@och`, `\f@nch@ech`, `\f@nch@ocf` or `\f@nch@ecf`. This is the center component.
4. `\f@nch@orh`, `\f@nch@erh`, `\f@nch@orf` or `\f@nch@erf`. This is the right component.
5. RIGHTSIDE BEARING. This is always `\relax` or `\hss` (after expansion).
Before constructing the header or footer, the environment is reset to a known state, and then the corresponding initialisation code as given in `\fancyheadinit` or `\fancyfootinit`, respectively, is run.

```
\newcommand\f@nch@head[5]{%
  \f@nch@reset
  \f@nch@headinit\relax
  #1%
  \hbox to\headwidth{%
    \f@nch@vbox\headheight{%
      \hbox{%
        \rlap{\parbox[b]{\headwidth}{\raggedright#2}}%
        \hfill
        \parbox[b]{\headwidth}{\centering#3}%
        \hfill
        \llap{\parbox[b]{\headwidth}{\raggedleft#4}}%
      }%
      \vskip\headruleskip\relax
      \headrule
    }%
  }%
  #5%
}
```

\f@nch@foot We put the `\footrule` in a `\vbox` to accommodate for flexible footrules (e.g. using `\hrulefill`), so that the `\headwidth` will be used as the line width. But to preserve the vertical spacing we then `\unvbox` this box.

```
\newcommand\f@nch@foot[5]{%
  \f@nch@reset
  \f@nch@footinit\relax
  #1%
  \hbox to\headwidth{%
    \f@nch@vbox\footskip{%
      \setbox0=\vbox{\footrule}\unvbox0
      \vskip\footruleskip
      \hbox{%
        \rlap{\parbox[t]{\headwidth}{\raggedright#2}}%
        \hfill
        \parbox[t]{\headwidth}{\centering#3}%
        \hfill
        \llap{\parbox[t]{\headwidth}{\raggedleft#4}}%
      }%
    }%
  }%
  #5%
}
```

\MakeUppercase Define For old LaTeXen. Note: we used `\def` rather than `\let`, so that `\let\uppercase\relax` (from the version 1 documentation) will still work.

```
\f@nch@ifundefined{MakeUppercase}{\def\MakeUppercase{\uppercase}}{}%
```

\@chapapp Define `\@chapapp` for classes that don't have it, e.g. amsbook

```
\f@nch@ifundefined{@chapapp}{\let\@chapapp\chaptername}{}%
```

\f@nch@initialise This macro initialises the headers and footers and `\chaptermark` and/or `\[sub]sectionmark` for pagestyle `fancy`

```
\def\f@nch@initialise{%
```

\chaptermark \sectionmark \subsectionmark Standard definitions for `\chaptermark`, `\sectionmark` and `\subsectionmark`.

```
  \f@nch@ifundefined{chapter}%
   {\def\sectionmark##1{\markboth{\MakeUppercase{\ifnum \c@secnumdepth>\z@
          \thesection\hskip 1em\relax
        \fi ##1}}{}}%
    \def\subsectionmark##1{\markright {\ifnum \c@secnumdepth >\@ne
      \thesubsection\hskip 1em\relax \fi ##1}}}%
   {\def\chaptermark##1{\markboth {\MakeUppercase{\ifnum
        \c@secnumdepth>\m@ne \@chapapp\ \thechapter. \ \fi ##1}}{}}%
    \def\sectionmark##1{\markright{\MakeUppercase{\ifnum \c@secnumdepth >\z@
        \thesection. \ \fi ##1}}}%
  }%
```

\headrule

```
  \def\headrule{{\if@fancyplain\let\headrulewidth\plainheadrulewidth\fi
      \hrule\@height\headrulewidth\@width\headwidth
      \vskip-\headrulewidth}}%
```

\footrule

```
  \def\footrule{{\if@fancyplain\let\footrulewidth\plainfootrulewidth\fi
      \hrule\@width\headwidth\@height\footrulewidth}}%
```

Default values for `\headrulewidth`, `\footrulewidth`, `\headruleskip` and `\footruleskip`.

```
  \def\headrulewidth{0.4pt}%
  \def\footrulewidth{0pt}%
  \def\headruleskip{0pt}%
  \def\footruleskip{0.3\normalbaselineskip}%
```

Initialisation of the head and foot text.

The default values still contain `\fancyplain` for compatibility: lefthead empty on "plain" pages, `\rightmark` on even, `\leftmark` on odd pages; evenhead empty on "plain" pages, `\leftmark` on even, `\rightmark` on odd pages.

```
  \fancyhf{}%
  \if@twoside
    \fancyhead[el,or]{\fancyplain{}{\slshape\rightmark}}%
    \fancyhead[er,ol]{\fancyplain{}{\slshape\leftmark}}%
  \else
    \fancyhead[l]{\fancyplain{}{\slshape\rightmark}}%
    \fancyhead[r]{\fancyplain{}{\slshape\leftmark}}%
  \fi
  \fancyfoot[c]{\rmfamily\thepage}% page number
}
```

Call the initialisation

```
\f@nch@initialise
```

\ps@f@nch@fancyproto

`\ps@f@nch@fancyproto` is the initial value for pagestyle `fancy`. The real page style is `\ps@f@nch@fancycore` but `\ps@f@nch@fancyproto` for the first use of `\pagestyle{fancy}` or any of its derivatives. It initialises `\headwidth`, and then resets itself to `\ps@f@nch@fancycore`. For backwards compatibility it still contains `\@fancyplainfalse`. The reason we have `\ps@f@nch@fancyproto` is so that page style `fancy` can be redefined.

```
\def\ps@f@nch@fancyproto{%
```

Initialise `\headwidth` if the user didn't. If `\headwidth` < 0, then the user did not initialise it, or they just added something to it in the expectation that it was initialised to `\textwidth`. We compensate this now. This loses if the user intended to multiply it by a factor. But that case is more likely done by saying something like `\setlength{\headwidth}{1.2\textwidth}`. The documentation advises to change `\headwidth` after the first call to `\pagestyle{fancy}`. This code is just to catch the most common cases were that is not the case.

```
  \ifdim\headwidth<0sp
    \global\advance\headwidth123456789sp\global\advance\headwidth\textwidth
  \fi
```

Now we reset `\ps@f@nch@fancyproto` to `\ps@f@nch@fancycore` with `\@fancyplainfalse` and call that version.

```
  \gdef\ps@f@nch@fancyproto{\@fancyplainfalse\ps@f@nch@fancycore}%
  \@fancyplainfalse\ps@f@nch@fancycore
}%
```

Let the system know this is a `fancyhdr` pagestyle.

```
\@namedef{f@nch@ps@f@nch@fancyproto-is-fancyhdr}{}
```

`ps@fancy` Define `\ps@fancy` just to call `\ps@f@nch@fancyproto`.

```
\def\ps@fancy{\ps@f@nch@fancyproto}
\@namedef{f@nch@ps@fancy-is-fancyhdr}{}
```

`\ps@fancyplain` The pagestyle `fancyplain` (deprecated). After initializing by calling `\ps@f@nch@fancyproto` it sets page style `plain` to our version `\ps@plain@fancy`, which just sets `\@fancyplaintrue` and then calls the page style fancy implementation.

```
\def\ps@fancyplain{\ps@f@nch@fancyproto \let\ps@plain\ps@plain@fancy}
\def\ps@plain@fancy{\@fancyplaintrue\ps@f@nch@fancycore}
```

`\f@nch@ps@empty` Save the definition of `\ps@empty` (pagestyle `empty`).

```
\let\f@nch@ps@empty\ps@empty
```

`\ps@f@nch@fancycore` The actual implementation of pagestyle `fancy`. For amsbook/amsart, which do strange things with `\topskip`, we start with `\f@nch@ps@empty`. We construct the even and odd headers and footers from all the parts that we have collected.

```
\def\ps@f@nch@fancycore{%
  \f@nch@ps@empty
  \def\@mkboth{\protect\markboth}%
  \def\@oddhead{\f@nch@head\f@nch@Oolh\f@nch@olh\f@nch@och\f@nch@orh\f@nch@Oorh}%
  \def\@oddfoot{\f@nch@foot\f@nch@Oolf\f@nch@olf\f@nch@ocf\f@nch@orf\f@nch@Oorf}%
  \def\@evenhead{\f@nch@head\f@nch@Oelh\f@nch@elh\f@nch@ech\f@nch@erh\f@nch@Oerh}%
  \def\@evenfoot{\f@nch@foot\f@nch@Oelf\f@nch@elf\f@nch@ecf\f@nch@erf\f@nch@Oerf}%
}
```

`ps@fancydefault` This is page style `fancydefault`. It is in fact page style `fancy` with all the defaults embedded. In contrast with page style `fancy` that gets all its settings from the environment. It reruns all initialisations and then calls `\ps@f@nch@fancyproto`.

```
\def\ps@fancydefault{%
```

```
  \f@nch@initialise
  \ps@f@nch@fancyproto
}
\@namedef{f@nch@ps@fancydefault-is-fancyhdr}{}
```

\f@nch@Oolh \f@nch@Oorh \f@nch@Oelh \f@nch@Oerh \f@nch@Oolf \f@nch@Oorf \f@nch@Oelf \f@nch@Oerf

Default definitions for compatibility mode: These cause the header/footer to take the defined `\headwidth` as its width and if required to shift it in the direction of the marginpar area.

```
\def\f@nch@Oolh{\if@reversemargin\hss\else\relax\fi}
\def\f@nch@Oorh{\if@reversemargin\relax\else\hss\fi}
\let\f@nch@Oelh\f@nch@Oorh
\let\f@nch@Oerh\f@nch@Oolh
\let\f@nch@Oolf\f@nch@Oolh
\let\f@nch@Oorf\f@nch@Oorh
\let\f@nch@Oelf\f@nch@Oelh
\let\f@nch@Oerf\f@nch@Oerh
```

\f@nch@offsolh \f@nch@offselh

New definitions for the use of `\fancyhfoffset`, `\fancyheadoffset`, `\fancyfootoffset`. These calculate the `\headwidth` from `\textwidth` and the specified offsets.
First for the header.

```
\def\f@nch@offsolh{\headwidth=\textwidth\advance\headwidth\f@nch@O@olh
                   \advance\headwidth\f@nch@O@orh\hskip-\f@nch@O@olh}
\def\f@nch@offselh{\headwidth=\textwidth\advance\headwidth\f@nch@O@elh
                   \advance\headwidth\f@nch@O@erh\hskip-\f@nch@O@elh}
```

\f@nch@offsolh \f@nch@offselh

The same for the footer.

```
\def\f@nch@offsolf{\headwidth=\textwidth\advance\headwidth\f@nch@O@olf
                   \advance\headwidth\f@nch@O@orf\hskip-\f@nch@O@olf}
\def\f@nch@offself{\headwidth=\textwidth\advance\headwidth\f@nch@O@elf
                   \advance\headwidth\f@nch@O@erf\hskip-\f@nch@O@elf}
```

\f@nch@setoffs

Set the offset parts to be used in the construction of the headers and footers. Depending on `\f@nch@gbl` it will be done globally (for pagestyle `fancy`) or locally (for `\fancypagestyle`). Just in case `\let\headwidth\textwidth` was used, we reset `\headwidth` to the length parameter that it should be.

```
\def\f@nch@setoffs{%
  \f@nch@gbl\let\headwidth\f@nch@headwidth
  \f@nch@gbl\let\f@nch@Oolh\f@nch@offsolh
  \f@nch@gbl\let\f@nch@Oelh\f@nch@offselh \f@nch@gbl\let\f@nch@Oorh\hss
  \f@nch@gbl\let\f@nch@Oerh\hss \f@nch@gbl\let\f@nch@Oolf\f@nch@offsolf
  \f@nch@gbl\let\f@nch@Oelf\f@nch@offself \f@nch@gbl\let\f@nch@Oorf\hss
  \f@nch@gbl\let\f@nch@Oerf\hss
}
```

\iff@nch@footnote \@makecol

Redefine `\@makecol` so that we can capture if there are top/bottom floats, footnotes or if we are on a float page. Because of a clash with the footmisc package

we do this at `\begin{document}`.
We need a boolean `\iff@nch@footnote` to capture if there was a footnote.

```
\newif\iff@nch@footnote
\AtBeginDocument{%
  \let\latex@makecol\@makecol
  \def\@makecol{\ifvoid\footins\f@nch@footnotefalse\else\f@nch@footnotetrue\fi
    \let\topfloat\@toplist\let\botfloat\@botlist\latex@makecol}%
}
```

`\iftopfloat` `\ifbotfloat` `\iffloatpage` `\iffootnote` These can be used in a header/footer field to make them conditional on the presence of floats and/or footnotes.

```
\newcommand\iftopfloat[2]{\ifx\topfloat\empty #2\else #1\fi}%
\newcommand\ifbotfloat[2]{\ifx\botfloat\empty #2\else #1\fi}%
\newcommand\iffloatpage[2]{\if@fcolmade #1\else #2\fi}%
\newcommand\iffootnote[2]{\iff@nch@footnote #1\else #2\fi}%
```

`\fancypagestyle` Define a new page style. The optional second argument is the base page style. It defaults to `\ps@f@nch@fancyproto`.

```
\newcommand{\fancypagestyle}[1]{%
  \@ifnextchar[{\f@nch@pagestyle{#1}}{\f@nch@pagestyle{#1}[f@nch@fancyproto]}%
}
```

`\f@nch@pagestyle` The actual code for `\fancypagestyle`. Build the page style body. Declare it as a fancyhdr page style.

```
\long\def\f@nch@pagestyle#1[#2]#3{%
  \f@nch@ifundefined{ps@#2}{%
    \f@nch@errmsg{\string\fancypagestyle: Unknown base page style `#2'}%
  }{%
    \f@nch@ifundefined{f@nch@ps@#2-is-fancyhdr}{%
      \f@nch@errmsg{\string\fancypagestyle: Base page style `#2' is not fancyhdr-bas
    }%
    {%
      \@namedef{ps@#1}{\let\f@nch@gbl\relax\@nameuse{ps@#2}#3\relax}%
      \@namedef{f@nch@ps@#1-is-fancyhdr}{}%
    }%
  }%
}%
```

</fancyhdr>

36 extramarks.sty

<*extramarks>

`\@temptokenb` A token register to store some marks information

```
\newtoks\@temptokenb
```

`\unrestored@protected@xdef` Define this macro just in case it isn't defined (should be part of LaTeX).

```
\providecommand\unrestored@protected@xdef{%
  \let\protect\@unexpandable@protect \xdef}
```

`\markboth` Our own definition of `\markboth`, mainly because `\@markboth` gets more parameters.

```
\def\markboth#1#2{%
  \begingroup
  \let\label\relax \let\index\relax \let\glossary\relax
  \expandafter\@markboth\@themark{#1}{#2}%
  \@temptokena \expandafter{\@themark}%
  \mark{\the\@temptokena}%
  \endgroup
  \if@nobreak\ifvmode\nobreak\fi\fi}
```

`\markright` We use the standard definition of `\markright`. No use to duplicate here.

`\@markboth` Note: put `#3#4` in toks register.

```
\def\@markboth#1#2#3#4#5#6{\@temptokena{{#3}{#4}}%
  \unrestored@protected@xdef\@themark{{#5}{#6}\the\@temptokena}}
```

`\@markright` Note: put `#1` and `#3#4` in toks registers. Maybe I can get rid of the extra `\@temptokenb` by doing the expansion of `#5` to a temp separately. But then, nowadays registers are plenty.

```
\def\@markright#1#2#3#4#5{\@temptokena{#1}\@temptokenb{{#3}{#4}}%
  \unrestored@protected@xdef\@themark{{\the\@temptokena}{#5}\the\@temptokenb}}
```

`\@leftmark` `\@rightmark` Internal macros to get the standard marks.

```
\def\@leftmark#1#2#3#4{#1}
\def\@rightmark#1#2#3#4{#2}
```

`\leftmark` `\rightmark` `\firstleftmark` `\lastrightmark` `\firstrightmark` `\lastleftmark` The standard marks + the new ones (based on the standard marks info).

```
\def\leftmark{\expandafter\@leftmark
      \botmark\@empty\@empty\@empty\@empty}
\def\rightmark{\expandafter\@rightmark
      \firstmark\@empty\@empty\@empty\@empty}
\def\firstleftmark{\expandafter\@leftmark
      \firstmark\@empty\@empty\@empty\@empty}
\def\lastrightmark{\expandafter\@rightmark
      \botmark\@empty\@empty\@empty\@empty}
\let\firstrightmark \rightmark
\let\lastleftmark \leftmark
```

`\@themark` This is where the marks information is stored.

```
\def\@themark{{}{}{}{}}
```

`\extramarks` This command is used to define the extra marks.

```
\newcommand\extramarks[2]{%
  \begingroup
  \let\label\relax \let\index\relax \let\glossary\relax
  \expandafter\@markextra\@themark{#1}{#2}%
  \@temptokena \expandafter{\@themark}%
  \mark{\the\@temptokena}%
  \endgroup
  \if@nobreak\ifvmode\nobreak\fi\fi}
```

`\@markextra` Internal macro to store the extra marks in the marks storage.
Note: Put `#1#2` in toks register.

```
\def\@markextra#1#2#3#4#5#6{\@temptokena {{#1}{#2}}%
  \unrestored@protected@xdef\@themark{\the\@temptokena{#5}{#6}}}
```

`\firstleftxmark` `\firstrightxmark` `\topleftxmark` `\toprightxmark` `\lastleftxmark` `\lastrightxmark` `\firstxmark` `\lastxmark` `\topxmark` The new extra marks.

```
\def\firstleftxmark{\expandafter\@leftxmark
      \firstmark\@empty\@empty\@empty\@empty}
\def\firstrightxmark{\expandafter\@rightxmark
      \firstmark\@empty\@empty\@empty\@empty}
\def\topleftxmark{\expandafter\@leftxmark
      \topmark\@empty\@empty\@empty\@empty}
\def\toprightxmark{\expandafter\@rightxmark
      \topmark\@empty\@empty\@empty\@empty}
\def\lastleftxmark{\expandafter\@leftxmark
      \botmark\@empty\@empty\@empty\@empty}
\def\lastrightxmark{\expandafter\@rightxmark
      \botmark\@empty\@empty\@empty\@empty}
\let\firstxmark\firstleftxmark
\let\lastxmark\lastrightxmark
\let\topxmark\topleftxmark
```

`\@tleftxmark` `\@rightxmark` Internal macros to extract the extra marks out of the marks storage.

```
\def\@leftxmark#1#2#3#4{#3}
\def\@rightxmark#1#2#3#4{#4}
```

</extramarks>

37 fancyheadings.sty

Fancyheadings.sty was the original style file (as they were called then) to implement fancy headers and footers in LaTeX. This was in the time when MSDOS was stil quite a dominant “Operating System”. It had a nasty property (amongst others): filenames consisted of at most 8 characters + a 3 character extension. This meant that the name ’`fancyheadings.sty`’ was internally truncated in MSDOS to ’`fancyhea.sty`’, although it was perfectly OK to say ’fancyheadings’ in LaTeX.

However, some people started to write also 'fancyhea' in LaTeX documents, which made them unportable to for example Unix systems, unless there a copy or link was made to 'fancyhea.sty'. I found this so annoying that I decided to rename the package to 'fancyhdr.sty'. This package has evolved to a version that is incompatible with the original 'fancyheadings'. Fancyheadings should no longer be used, therefore this package is provided that issues a clear warning and then switches to fancyhdr.

<*fancyheadings>

```
\PackageWarningNoLine{fancyheadings}{%
 Please stop using fancyheadings!\MessageBreak
 Use fancyhdr instead.\MessageBreak
 We will call fancyhdr with the very same\MessageBreak
 options you passed to fancyheadings.\MessageBreak
 \MessageBreak
 fancyhdr is 99 percent compatible with\MessageBreak
 fancyheadings. The only incompatibility is\MessageBreak
 that \protect\headrulewidth\space and \protect\footrulewidth\space
    and\MessageBreak
 their \protect\plain... versions are no longer length\MessageBreak
 parameters, but normal macros (to be changed\MessageBreak
 with \protect\renewcommand\space rather than \protect\setlength).}
\RequirePackage{fancyhdr}
```

</fancyheadings>

Change History

extramarks v1.99e
- General: Added a few % marks to get rid of unwanted spaces, and `\endinput`. Added LPPL license clause. . . 67

extramarks v2.0beta
- General: Adapted for the new implementation of marks in LaTeX to solve bug latex/3203. Added symmetric commands `\firstrightmark`, `\lastleftmark`, `\firstleftxmark`, `\firstrightxmark`, `\lastrightxmark`, `\lastleftxmark`, `\topleftxmark` and `\toprightxmark`. 67

fancyhdr v 2.0
- General: version 2.0 Release. . . . 67

extramarks v2.1
- General: Added a `\ProvidesPackage` line. Updated contact information. 67

extramarks v3.9
- General: Unify version number with fancyhdr.sty. 67

extramarks v3.9a
- General: Restore `\newtoks\@temptokenb` 67

fancyhdr v1.4
- General: Correction for use with `\reversemarginpar` 52

fancyhdr v1.5
- General: Added the `\iftopfloat`, `\ifbotfloat` and `\iffloatpage` commands . . . 52

fancyhdr v1.6
- General: Reset single spacing in

headers/footers for use with setspace.sty or doublespace.sty 52

fancyhdr v1.7

General: Changed `\let\@mkboth\markboth` to `\def\@mkboth{\protect\markboth}` to make it more robust. 52

fancyhdr v1.8

General: corrections for amsbook/amsart: define `\@chapapp` and (more importantly) use the `\chapter/sectionmark` definitions from `ps@headings` if they exist (which should be true for all standard classes). . 52

fancyhdr v1.9

General: The proposed `\renewcommand{\headrulewidth}{\iffloatpage...` construction in the doc did not work properly with the `fancyplain` style. 52

fancyhdr v1.91

General: The definition of `\@mkboth` wasn't restored on subsequent `\pagestyle{fancy}`'s. 52

fancyhdr v1.92

General: The sequence `\pagestyle{fancyplain}` `\pagestyle{plain}` `\pagestyle{fancy}` would erroneously select the plain version. 52

fancyhdr v1.93

General: `\fancypagestyle` command added. 52

fancyhdr v1.94

General: (suggested by Conrad Hughes `<chughes@maths.tcd.ie>`): added `\footruleskip` to allow control over footrule position (old hardcoded value of .3`\normalbaselineskip` is far too high when used with very small footer fonts). 52

fancyhdr v1.95

General: call `\@normalsize` in the reset code if that is defined, otherwise `\normalsize`. This is to solve a problem with ucthesis.cls, as this doesn't define `\@currsize`. Unfortunately for latex209 calling `\normalsize` doesn't work as this is optimized to do very little, so there `\@normalsize` should be called. Hopefully this code works for all versions of LaTeX known to mankind. 52

fancyhdr v1.96

General: Initialise `\headwidth` to a magic (negative) value to catch most common cases that people change it before calling `\pagestyle{fancy}`. Note it can't be initialised when reading in this file, because `\textwidth` could be changed afterwards. This is quite probable. We also switch to `\MakeUppercase` rather than `\uppercase` and introduce a `\nouppercase` command for use in headers. and footers. . . 52

fancyhdr v1.97

General: Two changes:
1. Undo the change in version 1.8 (using the `\pagestyle{headings}` defaults for the chapter and section marks). The current version of amsbook and amsart classes don't seem to need them anymore. Moreover the standard LaTeX classes don't use `\markboth` if twoside isn't selected, and this is confusing as `\leftmark` doesn't work as expected.
2. Include a call to `\ps@empty` in `\ps@@fancy`. This is to solve a problem in the amsbook and amsart classes, that make global changes to `\topskip`, which are reset in `\ps@empty`. Hopefully this doesn't break other things. 52

to change any of these, the value of `\z@skip` will not be changed, rather an errormessage will be given. . . 52

fancyhdr v3.0
  General: Release of version 3.0. . 52

fancyhdr v3.1
  General: Added '`\endlinechar=13`' to `\f@nch@reset` to prevent problems with `\includegraphics` in header/footer when `verbatiminput` is active. 52

fancyhdr v3.10
  `\f@nch@errmsg`: Use `\f@nch@ifundefined` instead of `\ifx`. 55
    Use `\newcommand` instead of `\def`. 55
  `\f@nch@foot`: Move `\footruleskip` outside of the `\footrule` definition. 63
    Put `\footrule` in a `\vbox` to accommodate for flexible footrules. 63
    Use `\unvbox` on the footrule `\vbox` to preserve vertical spacing. 63
  `\f@nch@vbox`: Don't use `\global\setlength`. 61
    Use `\newcommand` instead of `\def`. 61
  `\f@nch@warning`: Use `\f@nch@ifundefined` instead of `\ifx`. 55
    Use `\newcommand` instead of `\def`. 55
  `\footrule`: Move `\footruleskip` outside of the `\footrule` definition and remove useless `\vskip` at the top. 64

fancyhdr v3.2
  General: Reset `\everypar` (the real one) in `\f@nch@reset` because spanish.ldf does strange things with `\everypar` between « and ». 52

fancyhdr v3.3
  General: Replace '`\@ifundefined{chapter}`' with '`\ifx\chapter\@undefined`' because the former subtly makes `\chapter` equal to `\relax`, which may be undesirable in some cases. . . . 52

fancyhdr v3.4
  General: Replace `\rm` by `\normalfont\rmfamily` and `\sl` by `\normalfont\slshape`. 52

fancyhdr v3.5
  General: Don't define `\footruleskip` if it is already defined. 52

fancyhdr v3.6
  General: Added a `\ProvidesPackage` line.
    Updated contact information. 52

fancyhdr v3.7
  General: Removed `\normalfont` from default values, as every field is already initialised with `\normalfont`.
    Set `\hsize` to `\headwidth` in header/footer. 52

fancyhdr v3.8
  General: Reset `\\`, `\raggedleft`, `\raggedright` and `\centering` to their default values to avoid a clash with the tabu package.
    Move the redefinition of `\@makecol` to `\begin{document}` to avoid a clash with the footmisc package (and maybe others).
    Define a working `\iffootnote` command. 52

fancyhdr v3.9
  `\cfoot`: Let `\newcommand` do the handling of the optional parameter. 57
  `\chead`: Let `\newcommand` do the handling of the optional parameter. 57
  `\fancyfoot`: Let `\newcommand` do the handling of the optional parameter. 56
  `\fancyfootoffset`: Let `\newcommand` do the handling of the optional parameter. . . . 56
  `\fancyhead`: Let `\newcommand` do the handling of the optional

parameter. 56
\fancyheadoffset: Let \newcommand do the handling of the optional parameter. . . . 56
\fancyhf: Let \newcommand do the handling of the optional parameter. 56
\fancyhfoffset: Let \newcommand do the handling of the optional parameter. 56
\lfoot: Let \newcommand do the handling of the optional parameter. 57
\lhead: Let \newcommand do the handling of the optional parameter. 57
\rfoot: Let \newcommand do the handling of the optional parameter. 57
\rhead: Let \newcommand do the handling of the optional parameter. 57
General: Put everything in a .dtx file. 52
Rename some macros to have 'f@nch@' in their names, to get a more uniform naming scheme for internal macros. 52

fancyhdr v4.0
\f@nch@foot: \fancyfootinit initialisation code added and \f@nch@reset moved up. . . . 63
\f@nch@gbl: Implement the compatV3 option 52
Remove the \global in definitions 52
\f@nch@head: \fancyheadinit initialisation code added and \f@nch@reset moved up. . . . 62
Parameter \headruleskip. . . . 62
\f@nch@initialise: Put all the initialisation code in \f@nch@initialise 63
\f@nch@pagestyle: Make the definition of \f@nch@pagestyle \long. . . . 67
\f@nch@ps@empty: Rename \ps@@empty to \f@nch@ps@empty 65
\f@nch@vbox: Don't adjust the \headheight/\footskip, except when the compatV3 option is given. 61
Don't check when the nocheck option is given. 61
\fancycenter: Macro \fancycenter added 59
\fancypagestyle: Added optional base style argument. 67
\headruleskip: Parameter \headruleskip. 58
\iff@nch@check: Implement the nocheck option 52
\ps@f@nch@fancycore: Rename \ps@@fancy to \ps@f@nch@fancycore 65
\ps@f@nch@fancyproto: Reorganise \ps@fancy 64
General: Process package options. 55
ps@fancydefault: Added \ps@fancydefault 65

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.