The fancyhdr and extramarks packages

version v5.2.

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

Feb 7, 2025

Abstract

This document describes how to customize the page layout of your LaTeX documents, i.e., how to change page margins and sizes, headers and footers, and the proper placement of figures and tables (collectively called floats) on the page.
This documentation describes version 5.0 or later of the fancyhdr and extramarks packages. The user documentation is also mostly valid for the versions 4.0 or later of the fancyhdr and extramarks packages (except for the changes mentioned in sections 38.1 and 38.2).

Contents

---

[*] 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.

Part III contains Questions and Answers.

Part IV contains the annotated implementation.

This document describes version 5 of fancyhdr. This version is an extension of fancyhdr version 4, which is described in the *The LaTeX Companion, Third Edition*. It just has some additional commands that are not mentioned in *The LaTeX Companion*. The differences between these versions are summarized in section 38.1 on page 73, and section 38.2 on page 74. Throughout this documentation it is mentioned when a specific feature is only available in version 4 or a later version, or when there are differences between version 4 and 5.

This document also describes version 5 of extramarks. This is a new implementation that differs significantly from the previous versions. See section 4 on page 7 for more details.

This documentation contains several examples. Most of the examples are available for download from Github, see section 37. These examples are indicated with their name in the margin. If the margin says “Example ⟨*n*⟩”, where ⟨*n*⟩ is numeric, maybe followed by a letter, then the file name will be `example`⟨*n*⟩`.tex`. When it is followed by a letter in parentheses like (A), it means an item in the file. Other names without the word “Example” are just the file name without extension, for example “with-beamer” indicates the file name `with-beamer.tex`.

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, extramarks-v4.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. To get the documentation, run ‘`pdflatex fancyhdr.dtx`’.

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. See the next section for the details.)

`\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. Although in this documentation always uppercase letters are used in the ⟨*places*⟩, lowercase is also acceptable.

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

`\fancyheadoffset[⟨places⟩]{⟨length⟩}`
`\fancyfootoffset[⟨places⟩]{⟨length⟩}`
`\fancyhfoffset[⟨places⟩]{⟨length⟩}`

These define offsets to let the headers stick into the margin (or to the inside if negative). Places cannot contain the `C` specifier. See sections 21 and 22 for more details.

`\fancyheadwidth`
`\fancyfootwidth`
`\fancyhfwidth`

`\fancyheadwidth[⟨places⟩][⟨alignment⟩]{⟨length⟩}`
`\fancyfootwidth[⟨places⟩][⟨alignment⟩]{⟨length⟩}`
`\fancyhfwidth[⟨places⟩][⟨alignment⟩]{⟨length⟩}`

`\fancyheadwidth*[⟨places⟩][⟨alignment⟩]{⟨length⟩}`
`\fancyfootwidth*[⟨places⟩][⟨alignment⟩]{⟨length⟩}`
`\fancyhfwidth*[⟨places⟩][⟨alignment⟩]{⟨length⟩}`

These define widths and optionally the alignments for the header and footer fields. The fields will be typeset in a `\parbox` of this width, which can be different for each *place*. If the width of a field is not specified, it defaults to `\headwidth`, which may cause them to overlap. The alignment option and the `*` version are available in fancyhdr version 5.2 and later. See section 12 for more details.

`\headrulewidth`
`\footrulewidth`
`\headruleskip`
`\footruleskip`
`\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.) And `\headwidth` is a length parameter that defines the total width of the headers and footers. See section 22 for more details.

`\headrule`
`\footrule`

`\headrule` and `\footrule` are macros to completely redefine these lines.

`\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 28.1.

`\fancyfootalign`

`\fancyfootalign{}`
`\fancyfootalign{⟨length⟩}`

The command `\fancyfootalign` allows you to fine-tune the vertical position of the footer with respect to the page bottom. This command is only available in fancyhdr 5.0 and later. See section 20.

`\fancycenter`

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

(Only in version 4.0 and later.) The command `\fancycenter` packs 3 header fields into a full-width header. See section 13.

`\fancyhdrbox`

`\fancyhdrbox[⟨alignment⟩][⟨width⟩]{⟨lines separated by \\⟩}`

(Only in version 5.0 and later.) The command `\fancyhdrbox` can be used to align multi-line parts vertically and horizontally. See section 14.

`\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 23 for more details.

`\fancypagestyle`
`\fancypagestyle*`

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

This command lets you (re)define page styles for use in special situations. See sections 15 and 16 for more details.

`\fancypagestyleassign`

`\fancypagestyleassign{⟨ps1⟩}{⟨ps2⟩}`

This command assigns page style ⟨ps2⟩ to ⟨ps1⟩. This causes ⟨ps1⟩ to be an exact copy of ⟨ps2⟩, but completely independent of ⟨ps2⟩. Or you could say that ⟨ps1⟩ becomes a new name for page style ⟨ps2⟩. See section 31 for an example.

`\fancyhdrsettoheight`

`\fancyhdrsettoheight{⟨lengthvar⟩}{⟨header/footer⟩}`

Sets ⟨lengthvar⟩ to the height of the ⟨header/footer⟩, which must be one of `oddhead`, `evenhead`, `oddfoot` or `evenfoot`. Please note: You usually use this outside of a header or a footer (for example in the *preamble*, but then if you use marks with a non-standard height in your headers or footers, the calculated height may be wrong, as marks don't work well outside of a header or footer.

3 Package **fancyhdr** 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
`twoside`	use two-sided headers and footers even in one-sided documents for fancyhdr-based page styles (version 4.1 or later)
`headings`	redefine the `headings` page style to be fancy-based
`myheadings`	redefine the `myheadings` page style to be fancy-based

- Options `nocheck` and `compatV3` are described in section 20 on page 32.
- 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.
  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.

  This option is deprecated in version 5.0 of fancyhdr. It will disappear in a later release. Please don't use this option anymore, but rather change your document.
- Option `twoside` implements two-sided headers and footers in one-sided documents (version 4.1 or later). This applies only for fancyhdr-based page styles. This option doesn't do anything special for two-sided documents (`twoside` documentclass option), as these already have that functionality. And with the `twoside` documentclass option that does apply to other page styles as well.
- 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[1].

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.

Please note that these page styles redefine the `\chaptermark` and/or `\[sub]sectionmark` commands (see section 17), 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 16.

4 Using extramarks

Standard LaTeX has two marks: a left one 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 are to be used in the headers and footers of a page. These are derived from information that is given by the `\markboth` and `\markright` commands in the text body.

---

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

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

These macros give you the other combinations, where `\firstrightmark` = `\rightmark` and `\lastleftmark` = `\leftmark`.

`\extramarks`
`\extramarksleft`
`\extramarksright`

`\extramarks{`⟨*left-text*⟩`}{`⟨*right-text*⟩`}`
`\extramarksleft{`⟨*left-text*⟩`}`
`\extramarksright{`⟨*right-text*⟩`}`

The command `\extramarks{`$\langle m_1\rangle$`}{`$\langle m_2\rangle$`}` defines two extra marks, similar to the standard ones by LaTeX, where $\langle m_1\rangle$ is the left mark and $\langle m_2\rangle$ is the right mark.

In versions before 5.0, the `extramarks` are connected to each other and to the original LaTeX marks; they are not independent. For example, if you use `\markboth` or `\markright`, this introduced empty `extramarks` or duplicated existing ones. This is also true in the other direction. This sometimes caused unwanted effects.

Since version 5.0 this is no longer the case. Now the `extramarks` are independent of the traditional marks, and they can also be set independently of each other by the commands `\extramarksleft{`$\langle m_1\rangle$`}` and `\extramarksright{`$\langle m_2\rangle$`}`.

`extramarks-left`
`extramarks-right`

`extramarks-left`
`extramarks-right`

These are the 'mark classes' for the two marks.

NOTE: The implementation of extramarks version 5 only is available if your LaTeX release is the November 2022 LaTeX release or newer. It uses the new LaTeX marks introduced in that release. These marks are described in *The LaTeX Companion, Third Edition*, section 5.3.5 (Part I). Of course you can also use these new marks directly, or use additional ones if you need more. Some examples in this manual use these.

This manual contains several examples of the use of extramarks, where its features are essential, but in future releases of this manual these examples will be rewritten to use the new LaTeX marks directly.

Extramarks commands to be used in the headers or footers

`\firstleftxmark`
`\firstrightxmark`
`\topleftxmark`
`\toprightxmark`
`\lastleftxmark`
`\lastrightxmark`
`\firstxmark`
`\lastxmark`
`\topxmark`

These commands are used to extract the marks defined by `\extramarks{`$\langle m_1\rangle$`}{`$\langle m_2\rangle$`}`, `\extramarksleft{`$\langle m_1\rangle$`}` and `\extramarksright{`$\langle m_2\rangle$`}` described above. They are used in the headers or footers, similar to the ones without the `x`.

If you want to keep the old behaviour of extramarks, you can use:

```
\usepackage{extramarks}[=v4]
```

Please note that in that case the `\topleftxmark`, `\toprightxmark` and `\topxmark` commands may give you unexpected results.

See sections 17 and 30 for more details about the use of the package.

Part II

Page Layout in LaTeX

5 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 page break. If you want to change the height of a single page, you can use the `\enlargethispage` command.

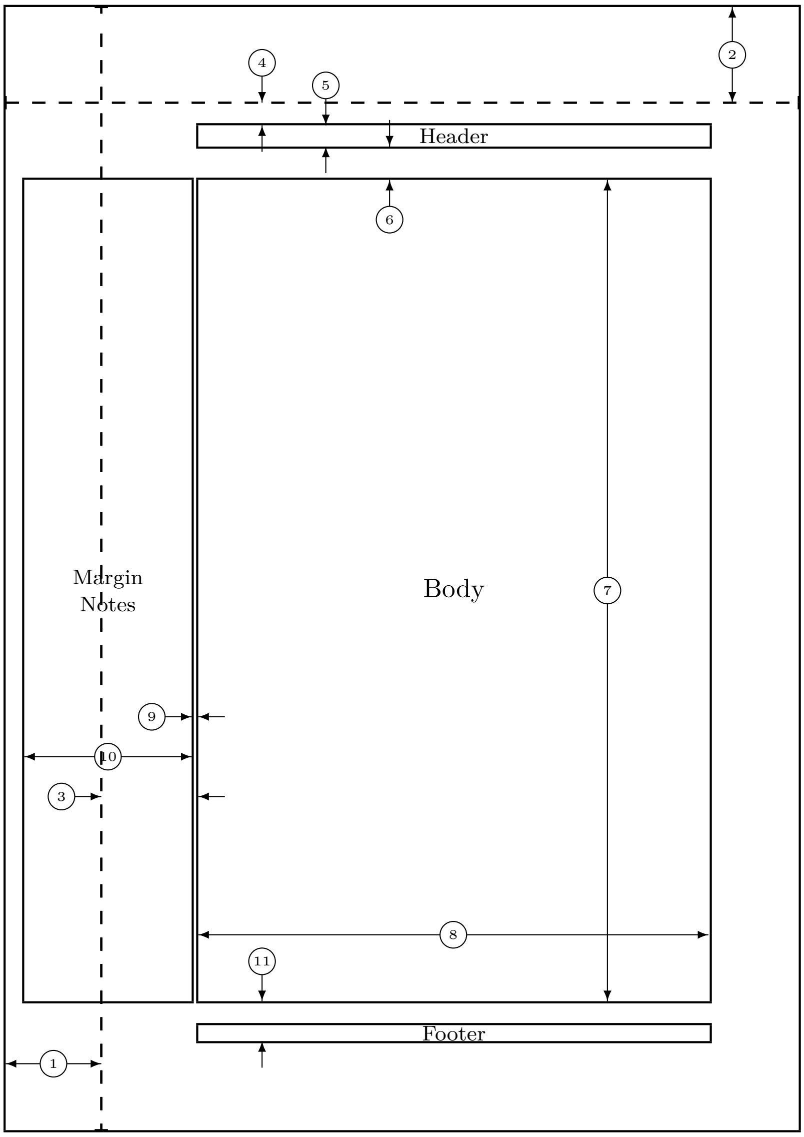
The margin notes area contains small pieces of information created by the `\marginpar` command. On two-sided documents the margin notes appear on the left and right alternatively. The margin notes are not on fixed places with respect to the paper but at approximately the same height as the paragraph in which they appear. Due to the algorithm used to decide the placement of margin notes, in a two-sided 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 32 and 33.

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.

6 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:

`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



1	one inch + \hoffset	2	one inch + \voffset
3	\oddsidemargin = 73pt	4	\topmargin = 17pt
5	\headheight = 16pt	6	\headsep = 25pt
7	\textheight = 618pt	8	\textwidth = 385pt
9	\marginparsep = 5pt	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.

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 page style 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 28).

7 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

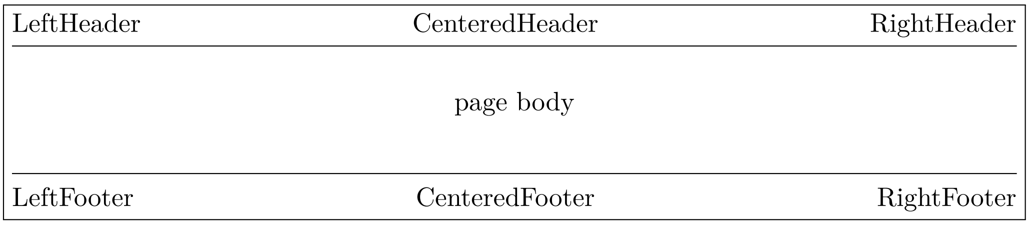
Of course, you also have complete control over fonts, uppercase and lowercase displays, etc.

8 Simple use of fancyhdr

To use this package install it in a place where LaTeX can find it (see section 1)[2], 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:

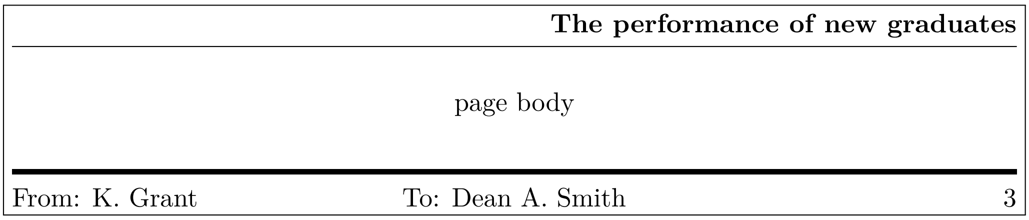


The LeftHeader and LeftFooter are left justified; the CenteredHeader and CenteredFooter are centered; the RightHeader and RightFooter are right justified.

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

9 A simple example

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



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}`[3]:

Example 1

```
\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}
```

---

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

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

(The `\thepage` macro displays the current page number. `\textbf` puts its argument 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

Example 2

```
\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`.

10 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:

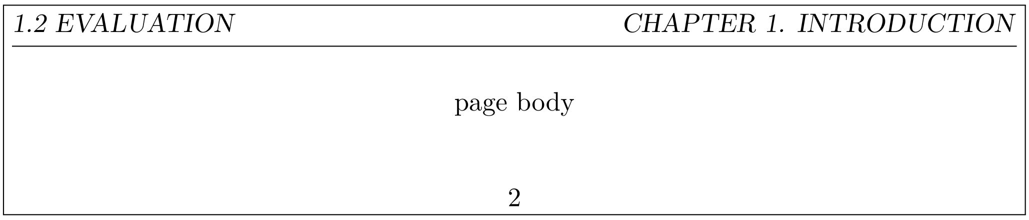
```
\usepackage{fancyhdr}
\pagestyle{fancy}
```

and let fancyhdr take care of everything. As mentioned before, we get a layout that is similar to the page style `headings`, but it is not exactly the same. If you want to have the same layout as the page style `headings`, but with a line under the header, use (you need fancyhdr version 4 or later for this):

```
\usepackage[headings]{fancyhdr}
\pagestyle{headings}
```

On the pages where new chapters start, we get a centered page number in the footer; there is nothing in the header, and there are no decorative lines.

On an even page, we get the layout:



1.2 EVALUATION *CHAPTER 1. INTRODUCTION*

page body

2

On an odd page, we get the layout:

CHAPTER 1. INTRODUCTION		*1.2 EVALUATION*
	page body	
	3	

where the header text is slanted uppercase.

In the `article` documentclass, we get section and subsection instead of chapter and section.

And in a one-sided document, all pages get the same layout as the even pages above. It would probably have been more logical to choose the odd page layout, but changing that now would break some existing documents. Anyway, you can change the layout easily yourself.

This default layout is produced by the following commands:

Example 5

```
\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.

11 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:

The performance of new graduates		
	page body	
4	From: K. Grant	To: Dean A. Smith

where "4" is the page number.

Here are the commands:

Example 3

```
\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. Similarly 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` prespecified, respectively.

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

12 Specifying the widths of the header and footer fields

In fancyhdr version 5.0 and later you can specify the width of each header and footer field individually. In older versions each of the fields was typeset in a `\parbox` of width `\headwidth`, which could cause them to overlap (see e.g., section 39).

In fancyhdr version 5.0 and later, this is still the default but now you can override this with the commands `\fancyheadwidth`, `\fancyfootwidth` or `\fancyhfwidth`. These work exactly like the `\fancyhead` etc. commands but instead of a header/footer value they have a ⟨*length*⟩ and an optional alignment as parameters.

```
\fancyheadwidth[⟨places⟩][⟨alignment⟩]{⟨length⟩}
\fancyfootwidth[⟨places⟩][⟨alignment⟩]{⟨length⟩}
\fancyhfwidth[⟨places⟩][⟨alignment⟩]{⟨length⟩}

\fancyheadwidth*[⟨places⟩][⟨alignment⟩]{⟨length⟩}
\fancyfootwidth*[⟨places⟩][⟨alignment⟩]{⟨length⟩}
\fancyhfwidth*[⟨places⟩][⟨alignment⟩]{⟨length⟩}
```

Field widths that are not specified default to `\headwidth`.

NOTE: In the non-`*` versions of these commands, the widths will be stored as expressions, not as calculated values. The values will be calculated when the header or footer is constructed. So they can change, for example when different pages have different `\headwidth` and you use e.g., `0.3\headwidth` or another expression with a 'variable' as value. Note, however, that at definition time, the width is assigned to a temporary length variable, to check if it is a legal ⟨*`length`*⟩. So any variables used in it should have a value, although this may be different from the value at its final use.

In the `*` versions of these commands, the width will be calculated at the call of the command, and the calculated value will be stored. So for example if you use an expression like '`0.4\headwidth`', and `\headwidth` is 10 cm, the value '4 cm' will be stored. If then then later `\headwidth` is changed to 15 cm when the header or footer is constructed, the value used for the field will still be 4 cm, which no longer is '`0.4\headwidth`'. With the non-`*` version, however, the value '6 cm' will be used, i.e. 0.4 of the then current `\headwidth`.

The fields are typeset in a `\parbox` with the specified width or the default `\headwidth`. It is still possible to get overlaps if the sum of the width in a particular header or footer is larger than `\headwidth`.

The fields will be positioned in a space of width `\headwidth` as follows:

- the left field will be positioned at the left edge of this space
- the right field will be positioned at the right edge of this space
- the position of the center field by default will be in the horizontal center of the header/footer. But it depends on the available space:

  Let W_L, W_C, and W_R be the width of the left, center and right field, respectively.

  - if the total width of the three fields $\sum_{i\in\{L,C,R\}} W_i >$ `\headwidth`, the center field will be centered in the header/footer, i.e., its midpoint will be at $\frac{1}{2}$`\headwidth` from each side.
    NOTE: this also includes the default situation if no widths are specified. This ensures that documents that don't specify widths get the same output as before version 5.0.
  - otherwise ($\sum_{i\in\{L,C,R\}} W_i \leq$ `\headwidth`):
  - if there would be an overlap between any of the fields, i.e.,
    $W_L + \frac{1}{2}W_C > \frac{1}{2}$`\headwidth` or $W_R + \frac{1}{2}W_C > \frac{1}{2}$`\headwidth`, then the center field will be centered between the left and right fields, with equal distances to both.
  - otherwise (there is enough space, and no overlap), the center field is centered in the header/footer, like the first case above.

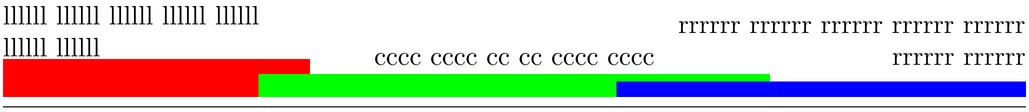
Here are some examples. The header fields have a colored bar in them that indicates their width.

In the first example, the sum of the field widths > `\headwidth`, so the center field will be in the center of the header, but there will be overlaps.

testheadwidth
(1.2)

```
\usepackage{fancyhdr}
\pagestyle{fancy}
\fancyhead[L]{llllll llllll llllll llllll llllll llllll llllll\\
              \color{red}\rule {\linewidth}{4mm}}
```

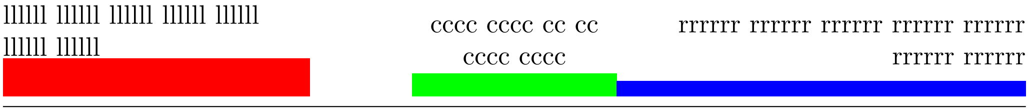
```
\fancyhead[C]{cccc cccc cc cc cccc cccc\\
              \color{green}\rule{\linewidth}{2mm}}
\fancyhead[R]{rrrrrr rrrrrr rrrrrr rrrrrr rrrrrr rrrrrr rrrrrr\\
              \color{blue}\rule {\linewidth}{4mm}}
\fancyheadwidth[L]{0.3\headwidth}
\fancyheadwidth[C]{0.5\headwidth}
\fancyheadwidth[R]{0.4\headwidth}
```


llllll llllll llllll llllll llllll
llllll llllll
cccc cccc cc cc cccc cccc
rrrrrr rrrrrr rrrrrr rrrrrr rrrrrr
rrrrrr rrrrrr

In the second example, the sum of the field widths ≤ `\headwidth`. And there is no overlap between the center field and the other ones (the center and right fields just touch each other), so the center field is still centered in the header.

testheadwidth
(1.3)

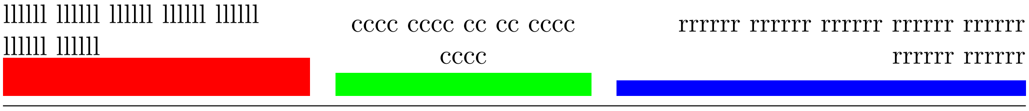
```
\fancyheadwidth[L]{0.3\headwidth}
\fancyheadwidth[C]{0.2\headwidth}
\fancyheadwidth[R]{0.4\headwidth}
```


llllll llllll llllll llllll llllll
llllll llllll
cccc cccc cc cc
cccc cccc
rrrrrr rrrrrr rrrrrr rrrrrr rrrrrr
rrrrrr rrrrrr

In the last example, the sum of the field widths still is ≤ `\headwidth`. But there would be overlap between the center field and the right field if the center field was centered horizontally in the header. So now it is centered between the left and right fields.

testheadwidth
(1.4)

```
\fancyheadwidth[L]{0.3\headwidth}
\fancyheadwidth[C]{0.25\headwidth}
\fancyheadwidth[R]{0.4\headwidth}
```


llllll llllll llllll llllll llllll
llllll llllll
cccc cccc cc cc cccc
cccc
rrrrrr rrrrrr rrrrrr rrrrrr rrrrrr
rrrrrr rrrrrr

The ⟨*alignment*⟩ optional parameter consists of two letters: a vertical alignment, which indicates where the baseline of the `\parbox` will be, followed by a horizontal alignment that specifies how the lines will be positioned horizontally in the `\parbox`. The alignment option is available in `fancyhdr` version 5.2 and later.

The possibilities for the vertical alignment are:

T The baseline of the `\parbox` is on the top of the first line, i.e., just where the top of the tallest character or item in the line is.

t The baseline of the `\parbox` is the baseline of the first line.

c The baseline of the `\parbox` is on the vertical center of the `\parbox`.

b The baseline of the `\parbox` is the baseline of the last line.

`B` The baseline of the `\parbox` is on the bottom of the last line, i.e., just where the bottom of the deepest character or item in the line is.

`-` Use the default vertical alignment.

The letters are the same as the vertical alignment in `\fancyhdrbox` (see section 14 for examples). The `t`, `c` and `b` are the standard options for `\parbox`.

The horizontal alignment can be:

`l` left aligned

`c` centered

`r` right aligned

`j` fully justified

These are the same as the horizontal alignment in `\fancyhdrbox`, except that the `j` option is extra. This one gives the default behavior in a `\parbox`.

The default for the vertical alignment is `b` for a header field, and `t` for a footer field. The default for the horizontal alignment is `l` for an `L` field, `c` for a `C` field, `r` for an `R` field.

Please note that a single `c` as the alignment counts as a vertical alignment, and the horizontal alignment will then be the default. If you want to specify only the horizontal alignment and take the default for the vertical alignment, specify the vertical alignment as `-`. For the horizontal alignment just omit it to get the default.

See section 39.4 for examples.

NOTE: The `\fancyheadwidth`, `\fancyfootwidth` and `\fancyhfwidth` commands are still experimental. This means that they have not been thoroughly tested, so there can still be bugs in them. And the implementation could change in a following release. Use at your own risk, and please, report any bugs.

13 Fancy Centering

Note: This section only applies to fancyhdr version 4.0 and later[4].

The fields in a fancy header and footer are prepared using `\parbox` command. So, you can use multi-line fields. 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 field is by default equal to the `\headwidth` (unless changed by the commands `\fancyheadwidth`, `\fancyfootwidth` or `\fancyhfwidth` from section 12.) This can lead to overlapping of neighbouring fields.

If you want to prepare headers/footers in more traditional way in a line not exceeding the `\headwidth`, you can use the following command in any header/footer command:

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

This command works like

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

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-field*⟩ is less than

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

---

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

Otherwise the ⟨*`center-field`*⟩ will slightly migrate to a shorter item (⟨*`left-field`*⟩ or ⟨*`right-field`*⟩), 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-field`*⟩ is empty, the `\fancycenter` is equivalent to the following command:

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

You would use this in a header for example with

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

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

Note 1: When `\fancycenter` is used inside a header or footer, `\linewidth` usually is the same as `\headwidth`. Only when `\fancycenter` is used inside a box with a different width, `\linewidth` will be the width of that box.

Note 2: If the whole of the `\fancycenter` is wider than `\linewidth` it will stick out on the right. See section 39 for possible solutions.

Note 3: The usage of the `\fancycenter` command is not limited to the argument of headers/footers. You can use it anywhere in your document. Then `\linewidth` will be the width of the box or text in which it is used.

14 The `\fancyhdrbox` command

The `\fancyhdrbox` command can be used to align multi-line header and footer fields and, for example, images. It is modelled after the makecell package by Olga Lapko, but it is a bit simplified, and also has extra vertical alignments `T` and `B`. And the vertical centering of `\fancyhdrbox` is better than the one from makecell. The `\fancyhdrbox` command is primarily meant for use in headers and footers, but can be used anywhere in a document.

The command is used as follows:

`\fancyhdrbox[`⟨*alignment*⟩`][`⟨*width*⟩`]{`⟨*lines separated by* `\\`⟩`}`

Here ⟨*`alignment`*⟩ specifies both the vertical and the horizontal alignment of the contents with respect to other text on the same line (including other `\fancyhdrbox` instances). The result of the command is a box in horizontal mode (in LaTeX parlance an **LR box**), similar to `\parbox` or `\makebox`.

The ⟨*`alignment`*⟩ optional parameter consists of two letters: a vertical alignment, which indicates where the baseline of the complete box will be, followed by a horizontal alignment that specifies how the lines will be positioned horizontally in the box.

The possibilities for the vertical alignment are:

`T` The baseline of the box is on the top of the first line, i.e., just where the top of the tallest character or item in the line is.

`t` The baseline of the box is the baseline of the first line.

`c` The baseline of the box is on the vertical center of the box.

`b` The baseline of the box is the baseline of the last line.

`B` The baseline of the box is on the bottom of the last line, i.e., just where the bottom of the deepest character or item in the line is.

The horizontal alignment can be

`l` left aligned

`c` centered

`r` right aligned

These are the same as, for example, in `tabular` columns.

Each of the vertical and horizontal alignments can be omitted. The default is `c` for the vertical alignment, and `l` for the horizontal alignment. If a single `c` is specified, it counts as both the vertical and horizontal alignment, i.e., as `cc`.

When multiple boxes are put next to each other (i.e., on the same line), their baselines will be aligned. Therefore in general it makes not much sense to specify different vertical alignments for them, unless you want a special effect. And in that case the results may be surprising.

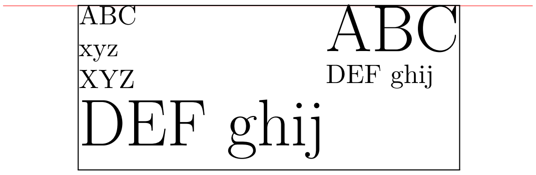
The second optional parameter, ⟨*`width`*⟩, specifies the width of the box. If this is not given, the box has its "natural" width, determined by its contents. With the ⟨*`width`*⟩ parameter, the width of the box is fixed to this value, independent of the contents. Note that there will be no automatic line breaking of the lines if they don't fit in the specified width. If a line is too long it will just stick out of the box, and may overlap the following text. If you want automatic line breaking, use a `\parbox`, a `tabular` with a `p{..}` column, or something similar.

The lines (rows) in the box are separated by `\\` just like in a tabular. You can even use `\\[`⟨*length*⟩`]` to add extra vertical space (or decrease the vertical space with a negative length). Also allowed is `\hline` after `\\`.

Here are examples of all the vertical alignment options, with some variations of the horizontal alignment. Some lines use a bigger font than others, in order to make the alignment non-trivial. All the `\fancyhdrbox` boxes are enclosed in a tight `\fbox` to show how big they are. The red horizontal line is the common baseline.

T-aligned boxes:

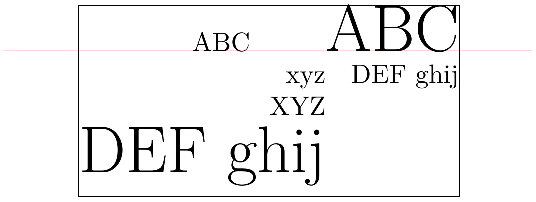
```
\fancyhdrbox[T]{%
   ABC \\
   xyz \\ XYZ \\
  \Huge DEF ghij
}%
\fancyhdrbox[T]{%
  {\Huge ABC} \\
  DEF ghij}
```



t-aligned boxes:

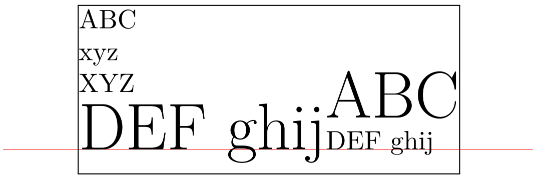
This example also uses right-alignment in the boxes, but the first line in the left box has a 1cm space added to the right, so it is shifted left 1cm.

```
\fancyhdrbox[tr]{%
   ABC\hspace{1cm} \\
   xyz \\ XYZ \\
  \Huge DEF ghij
}%
\fancyhdrbox[tr]{%
  {\Huge ABC} \\
  DEF ghij
}
```



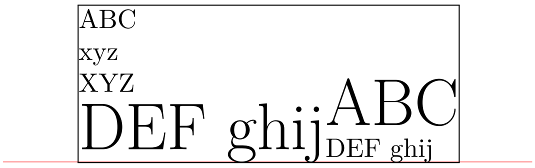
b-aligned boxes:

```
\fancyhdrbox[b]{%
   ABC \\
   xyz \\ XYZ \\
   \Huge DEF ghij
}%
\fancyhdrbox[b]{%
  {\Huge ABC} \\
  DEF ghij}
```



B-aligned boxes:

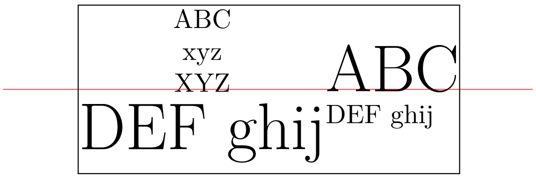
```
\fancyhdrbox[B]{%
   ABC \\
   xyz \\ XYZ \\
  \Huge DEF ghij
}%
\fancyhdrbox[B]{%
  {\Huge ABC} \\
  DEF ghij}
```



c-aligned boxes:

The first box has an explicit `[c]` positioning, which implies both vertical and horizontal centering. The second one uses the default positioning (i.e., it is not explicitly specified), which make it `[cl]`, i.e., horizontally left aligned.

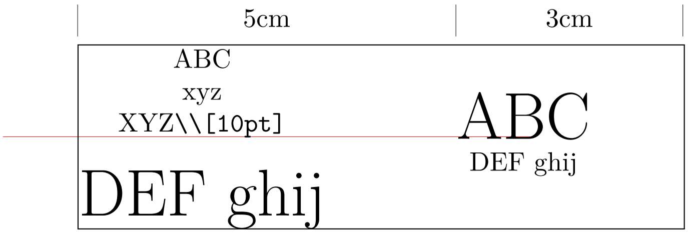
```
\fancyhdrbox[c]{%
   ABC \\
   xyz \\ XYZ \\
  \Huge DEF ghij
}%
\fancyhdrbox{%
  {\Huge ABC} \\
  DEF ghij}
```



c-aligned with ⟨*width*⟩:

This example shows the use of the second optional argument of `\fancyhdrbox`, the width of the box.

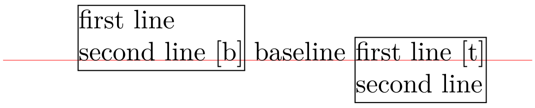
```
\fancyhdrbox[c][5cm]{%
  ABC \\ xyz \\ XYZ\texttt{\textbackslash\textbackslash[10pt]} \\[10pt]
  \Huge DEF ghij%
}%
\fancyhdrbox[c][3cm]{%
  {\Huge ABC}\\
  DEF ghij}
```



Different vertical alignments

Here is an example with two different vertical alignments in boxes next to each other, one with the `[b]` alignment and the other one with `[t]`.

```
\fbox{\showbaseline\fancyhdrbox[b]{%
   first line \\
   second line [b]
}}
baseline
\fbox{\fancyhdrbox[t]{%
  first line [t] \\
  second line}}
```



It may be surprising that the `[b]` box is on top and the `[t]` box on the bottom of the total line, but if you look at the baselines, it should become clear why this is so. This is just the way vertical alignment works in LaTeX. So if that is what you want, just use it.

Two headers with `\fancyhdrbox` parts

Finally, two headers with `\fancyhdrbox` parts. Note that these are in different header fields (left and right).

image+
twolineheader

```
\setlength{\headheight}{68pt}
\pagestyle{fancy}
\fancyhf{}
\rhead{\fancyhdrbox{\Large First Long Title \\ \large Second title}}
\lhead{\fancyhdrbox{\includegraphics[width=3cm]{example-image}}}
```

Image
First Long Title
Second title

In the next example the left header has an image in a `\fancyhdrbox` with the default alignment. The right header has two `\fancyhdrbox`|es, one with an explicit width of 4cm, the second one with its natural width.

threeboxes

```
\setlength{\headheight}{20pt}
\pagestyle{fancy}
\fancyhf{}
\fancyhead[L]{\sffamily
```

```
  \fancyhdrbox{\includegraphics[height=3\normalbaselineskip]
                               {example-image}}
}
\fancyhead[R]{%
  \fancyhdrbox[][4cm]{Our Office \\ Street 1 \\ City1 }%
  \fancyhdrbox[l]{ Our Factory \\ Street 2 \\ City2 }
}
```

Image	Our Office Street 1 City1	Our Factory Street 2 City2

15 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 and an optional one in between: the first parameter is the name of the page style to be defined, then an optional parameter of an existing base page style can be given, and the last parameter 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 optional base style, and finally the definitions in the last parameter. For details see the next section. In particular, if the last parameter is empty, i.e., given as `{}`, then the new page style is equal to base style.

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

```
\fancypagestyle{plain}[fancy]{}
```

If you have not redefined page style `fancy` with `\fancypagestyle`, this is equivalent to:

```
\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 11 by making the page number bold and enclosing it in en-dashes without any rules.

Example 4

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

16 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. With `*` it defines a "*closed*" page style, otherwise an "*open*" one. The difference is that the *open* page style does not necessarily have all the information in itself that is necessary to construct the headers and footers. So it will need to pick up the remaining elements from the environment of the text. The *closed* page style, however, will pick up all necessary elements from the environment at the moment it is **defined**, rather than when it is used, and carries that with it. The information that is picked up consists of:

- The header and footer fields in all variants (EO,LRC,HF) (12 items)
- The header and footer field widths in all variants (EO,LRC,HF) (12 items)
- The header and footer field alignments in all variants (EO,LRC,HF) (12 items)
- The header and footer offsets (EO,LR,HF) (8 items), see section 21 and 22
- The header and footer init values (2 items), see section 28.1
- `\headrule`, `\headrulewidth`, `\footrule`, `\footrulewidth` (4 items)
- the `[nocheck]` option

The *closed* versions can come handy when you are switching back and forth between different page styles, as explained in section 28.

Here is an example of a simple (*open*) definition:

```
\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 17 for an example). The headers and footers and marks that are not redefined inside the `\fancypagestyle` definition, are taken from the global fancy page style values.

The general form of the command is:

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

As you see, there is an optional `[`⟨*base-style*⟩`]` argument between the two mandatory arguments.

If you give this optional base page style to the `\fancypagestyle` command, then the new page style will be based on that base style. This base style must be a `fancyhdr`-defined style. Also you should take care not to create circular dependencies. When no base style is given, an internal base style, which has the default values is used. This is the same as page style `fancy`, unless the latter has been redefined. The order of picking up the definitions (headers, footers, marks, etc.) 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 rules above, are taken from the environment, for an *open* page style at the time the new page style is used, for a *closed* page style at the time it is defined.

In an *open* page style, i.e., if you use the form `\fancypagestyle[`⟨*base-style*⟩`]{`⟨*style-name*⟩`}`... only the first two parts are embedded in the page style.

The optional base style argument is only available in `fancyhdr` version 4.0 and later. In these versions it is also possible to redefine page style `fancy` in this way. In version 3.x and earlier this was not possible. The starred (*closed*) version is only available in `fancyhdr` version 5.0 and later.

The page style `fancydefault`

If you want to restore the original default definitions from page style `fancy` as described in section 10, 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, including the relevant definitions of `\chaptermark` and `\[sub]sectionmark`. 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. It is in fact the *closed* version of page style `fancy`, defined with `\fancypagestyle*{fancydefault}{`⟨*initialisation code*⟩`}` just after `fancyhdr`'s initialisation. However, `fancydefault` is only available since `fancyhdr` version 4.0.

If you don't like the defaults, you can redefine it yourself. For example if you don't want to include the `\...mark` definitions, just put `\fancypagestyle*{fancydefault}{}` after `\usepackage{fancyhdr}`. Or if you want to include your own header and/or footer definitions, use `\fancypagestyle*{fancydefault}{`⟨*your definitions*⟩`}`.

16.1 The `\fancypagestyleassign` command

The `\fancypagestyleassign` command is only available in `fancyhdr` version 5.0 and later. The command `\fancypagestyleassign{⟨ps1⟩}{⟨ps2⟩}` makes page style ⟨ps1⟩ an exact copy of page style ⟨ps1⟩. The effect is similar to the command `\fancypagestyle{⟨ps1⟩}[⟨ps2⟩]{}`, but there are important differences:

- with `\fancypagestyleassign{⟨ps1⟩}{⟨ps2⟩}` the page style ⟨ps1⟩ will be completely independent from ⟨ps2⟩. On the other hand, with `\fancypagestyle{⟨ps1⟩}[⟨ps2⟩]{}` the page style ⟨ps1⟩ will depend on ⟨ps2⟩. If ⟨ps2⟩ later changes (for example with a redefinition with `\fancypagestyle`), the page style ⟨ps1⟩ will change accordingly.
- with `\fancypagestyle{⟨ps1⟩}[⟨ps2⟩]{}` you must take care that you don't get cyclical dependencies, whereas with `\fancypagestyleassign` you can't create cyclical dependencies.
- with `\fancypagestyle{⟨ps1⟩}[⟨ps2⟩]{}` the page style ⟨ps2⟩ must be a page style that is defined by `fancyhdr` (with `\fancypagestyle` or predefined), but `\fancypagestyleassign{⟨ps1⟩}{⟨ps2⟩}` can also be used if ⟨ps2⟩ is not defined by `fancyhdr`, for example a standard LaTeX page style like `plain`.

If ⟨ps2⟩ is defined by `fancyhdr`, then also ⟨ps1⟩ is considered to be defined by `fancyhdr`. If ⟨ps2⟩ is a *closed* page style, then ⟨ps1⟩ is also *closed*.

`\fancypagestyleassign` comes especially handy if you want to temporarily redefine a page style, and later to restore it to its original value. For example, if we have a page style `special`, and we want temporarily to define page style `plain` to be equal to this, but later to restore it to its original definition, you can do this as follows:

```
\fancypagestyleassign{origplain}{plain}
\fancypagestyleassign{plain}{special}
. . .  code where plain is equal to special
\clearpage
\fancypagestyleassign{plain}{origplain}
. . . code where plain has its original meaning
```

Note that you couldn't do this with `\fancypagestyle` because (1) this would introduce a cyclical dependency, (2) you cannot use `plain` as the base page style, because it is not `fancyhdr`-based.

See section 31 for an example.

17 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, Third Edition*, section 5.3.4 (Part I).

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 10.

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

Example 6

```
\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 identity operations.

Example 7

```
\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].

---

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

[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:

Example 8

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

Chapter 2. Implementation

Example 9

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

CHAPTER 2. Implementation

Example 10

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

CHAPTER 2. IMPLEMENTATION

Example 11

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

Implementation

Example 12

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

2. Implementation

Example 13

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

2. Chapter. Implementation

Figure 3: Marker variants

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 30 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.

Example 14

```
\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 original TeX machinery had only one `\mark`. All the marks had to be packed together in this one. So there were no independent left or right marks. Modern LaTeX distributions, however, do have independent marks, so this problem can be solved. See Example 35 in section 42 for an example.

18 Headers for unnumbered chapters, sections, etc.

In the standard LaTeX documentclasses the `*` forms of the `\chapter` etc. commands do *not* call the mark commands. So these don't appear in the header. Neither are they put in the Table of Contents. So, for example, if you want your Preface to set the header info but not be numbered, you must issue the `\markboth` command yourself, e.g.,

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

Or for a section:

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

It can be a bit annoying to have to repeat the title. If you don't want that, it is possible to redefine the `\chapter` and/or `\section` command, in such a way that the `*` version *does* set the header info. For a chapter this is usually done with the `\markboth` command. For a section in a chapter-oriented documentclass with `\markright`, otherwise also with `\markboth`.

Here is a definition that accomplishes this. Redefine the `\chapter` command:
`\chapter[`⟨*header*⟩`]{`⟨*title*⟩`}`
`\chapter*{`⟨*title*⟩`}`
For the `\chapter*` version, we insert a `\markboth` command. For the non-`*` version we just pass the arguments to the original `\chapter` command.

We use the `\RenewDocumentCommand` to redefine the `\chapter` command because it allows us to redefine also the `*` variant, which is much more difficult with `\renewcommand`[7].

The `{som}` in the definition defines the arguments of the `\chapter` command:

1. `s` - a `*` which can be present or absent. This is checked with `\IfBooleanTF{#1}`
2. `o` - an optional argument. the presence of the optional argument is checked with `\IfNoValueTF{#2}`
3. `m` - a mandatory argument

We first save the original definition of `\chapter` in `\originalchapter` with the `\let` statement. The `\newcommand\originalchapter{}` is just a precaution to get an error message if `\originalchapter` was already defined, for example by another package.

---

[7] If you have an older LaTeX distribution that doesn't have the `\RenewDocumentCommand`, include `\usepackage{xparse}` in your preamble. Or better: update your LaTeX installation.

unnumbered

```
\newcommand\originalchapter{}% check that we can define this name
\let\originalchapter\chapter
\RenewDocumentCommand \chapter {som}{%
  \IfBooleanTF{#1}
    {% \chapter*
      \originalchapter*{#3}%
      \markboth{#3}{}%
      % we can also put it in the Table of Contents
      \addcontentsline{toc}{chapter}{#3}
    }%
    {% normal \chapter
      \IfNoValueTF{#2}
        {\originalchapter{#3}}
        {\originalchapter[#2]{#3}}%
    }%
}
```

We can do the same for the `\section` command, but we use `\markright` instead of `\markboth`. Note that the `\mark..` commands are called **after** the original command, because the `\chapter` command begins with a page break, and a `\section` could have a page break before it, but not after it.

NOTE: We don't use `\chaptermark` or `\sectionmark` here because these often include the chapter/section number, which doesn't make sense for an unnumbered one.

unnumbered

```
\newcommand\originalsection{}% check that we can define this name
\let\originalsection\section
\RenewDocumentCommand \section {som}{%
  \IfBooleanTF{#1}
    {% \section*
      \originalsection*{#3}%
      \markright{#3}%
      % we can also put it in the Table of Contents
      \addcontentsline{toc}{section}{#3}
    }%
    {% normal \section
      \IfNoValueTF{#2}
        {\originalsection{#3}}
        {\originalsection[#2]{#3}}%
    }%
}
```

Please note that, contrary to the original LaTeX commands, these new command do accept an optional argument with the `*` version, but if it is given, they don't use it. It is not difficult to add additional code to process this optional argument similar to the non-`*` case. This is left as an exercise for the reader, or look at the example files `unnumberedart1.tex` and `unnumberedart2.tex`.

19 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:

Example 15

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

20 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"[8]. See the warning below. See also section 5.1 and 5.2 of the LaTeX *Companion, Third Edition*, (Part I) 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:

Example 16

```
\documentclass{article}
\usepackage{fancyhdr}
\pagestyle{fancy}
\addtolength{\headheight}{\baselineskip}
\renewcommand{\sectionmark}[1]{\markboth{#1}{}}
```

---

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

```
\renewcommand{\subsectionmark}[1]{\markright{#1}}
\fancyhead[R]{\leftmark\\\rightmark}
```

Note that if you want to use header or footer layouts with multi-line parts that have to be aligned, you can do this with the `\fancyhdrbox` command. See section 14.

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:
(fancyhdr)                \addtolength{\topmargin}{-2.0pt}.
```

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[9].

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

This gives us the following headrule:

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

Example 17

```
\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 unusually large or small fonts in the footer. Change it with `\renewcommand`.

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

---

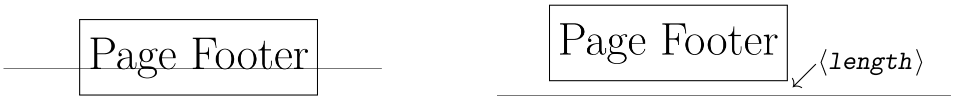
[9] Based upon an idea by Wayne Chan.

to `\footruleskip`. [10] 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).

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 and `\footruleskip` is 3.6pt (0.3`\normalbaselineskip`).

The code for this can be found in section 28.1.

Fine-tuning the footer position. By default LaTeX positions the baseline of the footer on the bottom edge of the bottom margin (the lower line of the footer box in figure 1). Most of the time this is what you want, but it means that any descenders in the footer (symbols that extend below the baseline, e.g., p and g or parentheses). See figure 4a, where the horizontal line denotes the bottom border.



(a) default footer positioning

(b) footer positioning with `\fancyfootalign{`⟨*length*⟩`}`

Figure 4: Vertical footer positioning

In some cases this is undesirable, for example when the bottom border is completely missing (`0pt`). In that case the descenders are cut off because they are outside of the paper, and even if there are no descenders, the resulting layout with the footer at the edge of the paper isn't esthetic. The beamer class has this layout.

In this case we can shift the footer up with the `\fancyfootalign` command (only available in fancyhdr 5.0 and later). This command has two versions:

`\fancyfootalign{}` – This selects the default alignment, as in figure 4a.

`\fancyfootalign{`⟨*length*⟩`}` – This gives extra space of ⟨*length*⟩ between *the bottom of the footer, (including the space for descenders and the interline space)*, and the border. See figure 4b.

Usually a ⟨*length*⟩ of `0pt` is sufficient; this means that the bottom of the footer box coincides with the bottom border. You can also use negative ⟨*length*⟩ values, so that the footer box only partially sticks out under the border. A given ⟨*length*⟩ applies to all subsequent footers (but is subject to the local group structure). It can be cancelled by `\fancyfootalign{}`. See section 41 for an example.

21 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

---

[10] But `\headruleskip` is only available since version 4.0.

outside margin where the marginal notes are printed, add both `\marginparsep` and `\marginparwidth` to `\headwidth` with the commands:

```
\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.)

Example 18

```
\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):

Example 19

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

NOTE: If you change the `\textwidth` in the middle of your document, for example by using the `geometry` package, by default the `\headwidth` will not change, as it picks up the value of `\textwidth` at the beginning of the document. If you want it to track the changes to `\textwidth`, you should use the command `\fancyhfoffset{0pt}` in the neighborhood of your header/footer definitions, unless you already use such an `...offset` command, of course. For the second example, we take the AMS-LaTeX book[11].

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 17):

```
\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 (originally by the late Sebastian Rahtz) uses the short (Karl Berry) name `phv` for Helvetica. The more modern LaTeX solution is to use the TeX Gyre font Heros, which uses the short name `qhv` so this font is selected with the commands[12]:

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

Let us define a shorthand for this:

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

Now we are ready for the page layout:

Example 20

```
\documentclass{book}
\usepackage{fancyhdr}
\pagestyle{fancy}
\renewcommand{\chaptermark}[1]%
   {\markboth{\MakeUppercase{\thechapter.\ #1}}{}}
```

---

[11] George Grätzer, *Math into LaTeX, An Introduction to LaTeX and AMS-LaTeX*, Birkhauser, Boston.

[12] See *The LaTeX Companion, Third Edition*, Part I, section 9.5.2, and Part II, section 10.8.16.

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

22 Summary of `\headwidth` calculation

Here is a summary of the calculation of the widths of headers and footers, as illustrated in the previous section.

- If no `\fancy...offset` commands are given, the default value for `\headwidth` is `\textwidth`. This is used for the width of both the header and the footer. It is possible to change the value of `\headwidth`, for example with `\setlength` or `\addtolength`. The excess or deficit will be applied to the right for a one-sided document, and for a two-sided document to the right on odd pages and to the left on even pages. The header and the footer will have the same width, `\headwidth`.
- If some `\fancy...offset` command is given, the header and footer widths are independently calculated by adding the appropriate offsets to `\textwidth`. Any changes made to `\headwidth` will not be taken into account. The header/footer will stick in/out at the proper side(s) specified by the offsets.

The file `example-headwidth.tex` in the `Examples` branch of the repository illustrates this.

23 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:

Example 21

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

NOTE: There is also a package floatpag[13] by Vytas Statulevičius and Sigitas Tolušis that has a command `\floatpagestyle{`*⟨pagestyle⟩*`}`, that applies *⟨pagestyle⟩* to all float pages, where *⟨pagestyle⟩* can be defined with `\fancypagestyle` (or by any other

---

[13] https://www.ctan.org/pkg/floatpag

means). In some cases this might be simpler than putting `\iffloatpage` in various headers or footers.

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.

24 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
```

25 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:

Example 22

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

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

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

We clear all the headers including its rule. The footer will be “inherited” from the page style `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 by checking if the page number is odd, and if it is equal to the number of the last page.

We use the macro `\getpagerefnumber` from the package refcount, because `\pageref` isn't always usable in a numerical context (it is meant for typesetting only). This is also done in following similar examples.

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

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

26 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 page number 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.

Example 23

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

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

In general it is advisable to give a `\clearpage` or `\cleardoublepage` before changing the page numbering.

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 page numbers 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`.

27 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}}
```

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}
```

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

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 switch 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}
```

Finally, in this section, we give an example that illustrates why using *closed* page styles is recommendable.

Suppose we have a part of our document, maybe one or more chapters, that need a different style headers and/or footers than the rest of the document. We can do this by defining a new page style for this part with `\fancypagestyle`. First we use the traditional (*open*) form:

switchstyle1

```
\pagestyle{fancy}
\fancyhf{}
\fancyhead[L]{\leftmark}
\fancyhead[R]{\rightmark}
\fancyfoot[C]{\thepage}

\fancypagestyle{special}{%
  \fancyhf{}
  \renewcommand{\headrulewidth}{0pt}
  \fancyhead[L]{Special Page Style \nouppercase\leftmark}
  \fancyfoot[R]{\thepage}
}
. . .
\chapter{Special Chapter}
\pagestyle{special}
Chapter text

\chapter{Another Chapter}
\pagestyle{fancy}
Chapter text
```

Now the last chapter will not use the headers and footers that we defined in the beginning, but those that are defined in page style `special`. This is because the command `\pagestyle{special}` will just execute the definitions inside it, and so it changes the definitions of `\fancyhead[]` etc. Also the definition of `\headrulewidth` will not be restored.

To remedy this we would need to put the relevant definitions inside the page style `fancy`. First we try this with the *open* `\fancypagestyle`.

switchstyle2

```
\fancypagestyle{fancy}{%
  \renewcommand{\headrulewidth}{0.4pt}
  \fancyhf{}
  \fancyhead[L]{\leftmark}
  \fancyhead[R]{\rightmark}
  \fancyfoot[C]{\thepage}
}

\fancypagestyle{special}{%
  \fancyhf{}
  \renewcommand{\headrulewidth}{0pt}
  \fancyhead[L]{Special Page Style \nouppercase\leftmark}
  \fancyfoot[R]{\thepage}
}
\pagestyle{fancy}
. . .
\chapter{Special Chapter}
\pagestyle{special}
Chapter text
```

```
\chapter{Another Chapter}
\pagestyle{fancy}
Chapter text
```

We now have the relevant definitions also embedded in page style `fancy`. Note that we have to include the (default) definition of `\headrulewidth`, although it looks unlogical that we have to do this. But we need it because page style `special` changes it. And if we had another page style that would change for example the `offsets` (see section 21) then we would also have to include these. This is the reason for the existence of the *closed* form `\fancypagestyle*`. So now we give the solution with these. This solves the problem in an elegant and robust way.

switchstyle3

```
\fancypagestyle*{fancy}{%
  \fancyhf{}
  \fancyhead[L]{\leftmark}
  \fancyhead[R]{\rightmark}
  \fancyfoot[C]{\thepage}
}

\fancypagestyle*{special}{%
  \fancyhf{}
  \renewcommand{\headrulewidth}{0pt}
  \fancyhead[L]{Special Page Style \nouppercase\leftmark}
  \fancyfoot[R]{\thepage}
}
\pagestyle{fancy}
. . .
\chapter{Special Chapter}
\pagestyle{special}
Chapter text

\chapter{Another Chapter}
\pagestyle{fancy}
Chapter text
```

28 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:

Example 24

```
\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 30.

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 17 and section 30.

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.

28.1 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 page style that makes the choice:

Example 25 (a)

```
\usepackage{ifthen}
\usepackage{refcount}
. . .
\fancypagestyle{switch}{
  \fancyhead[LE,RO]{%
    \ifthenelse{\value{page}=\getpagerefnumber{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 done 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 34, 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}=\getpagerefnumber{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}`[14]:

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

Now here is the definition of the page style used for page 34.

---

[14] Assuming we have already loaded package refcount.

Example 25 (b)

```
\fancypagestyle{showstruts}{%
  \fancyhead[L]{%
    \ifthenelse{\value{page}=\getpagerefnumber{showstruts}}%
      {\strutheader}%
      {\rightmark}%
  }
  \fancyfoot[L]{%
    \ifthenelse{\value{page}=\getpagerefnumber{showstruts}}%
      {\strutfooter}%
      {}%
  }
  \fancyheadinit{%
    \ifthenelse{\value{page}=\getpagerefnumber{showstruts}}%
      {\renewcommand{\headruleskip}{4pt}}%
      {\renewcommand{\headruleskip}{0pt}}%
  }
  \fancyfootinit{%
    \ifthenelse{\value{page}=\getpagerefnumber{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}=\getpagerefnumber{otherpagestyle}}
    {\color{red}\Large}
    {}
}
```

28.2 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[15]. Here is an example to change the page style of the next page with `\afterpage`:

Example 26

```
\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`.

28.3 Changing the page style in a TeX group

Special care has to be taken when you change the page style inside a TeX group. This can be any environment, text between `\begingroup` and `\endgroup`, between { and }, and other similar situations. TeX definitions inside such a group are local to this group, unless they are declared to be global. All definitions pertaining to the page style (i.e., `\fancypagestyle`, `\pagestyle`, `\fancyhead`, etc.) are local definitions, i.e., they disappear at the end of the group. The only exception is `\thispagestyle`, which is global, i.e., its setting survives the end of the group.

An example is the `appendices` environment of the package appendix[16] that you use to get special layout for your appendices. If you also want to change the page headers and/or footers for the appendices, you could use

```
\clearpage
\begin{appendices}
  \pagestyle{appendices}
  \chapter{My Appendix}
  Appendix text.
\end{appendices}
\chapter*{Bibliograpy}
```

Note that we put a `\clearpage` before the environment to prevent that the page before this environment gets the new page style, as indicated in sections 28.1 and 28.2. In the example above, it is probable that the `appendices` environment does not end with a `\newpage` or `\clearpage`. Then a page break will be given by the following `\chapter` command, but then the ‘Special’ page style will no longer be current, so the last page of the `appendices` environment will have the headers and footers that were current before the environment started. If there were still floats to be output at the end of

---

[15] 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.

[16] Use the command ‘`texdoc appendix`’ to see its documentation.

the `appendices` environment, this could even be several pages. So we should put a `\clearpage` before the `\end{appendices}`.

Here follows a more stylized example. The intention is to give the pages of the environment the header “Special Header”. First, the “wrong” implementation.

Example 26G (a)

```
\fancypagestyle{Special}{
  % setting the header in beginning of environment
  \fancyhead[C]{Special Header}
}%

\newenvironment{Special}[1]{%
  \pagestyle{Special}%
  \section*{Special Environment #1}
}{%
}
...
\begin{Special}{a}

Some text or a lot of text.

\end{Special}
```

Now the last page of this environment, which may be the first page if the environment fits on one page, will get the wrong page header.

The first solution would be to end the environment with a `\newpage` or `\clearpage` as described above. Generally, it is best to use `\clearpage`, because it also takes care of extra pages with floats.

Example 26G (b)

```
\begin{Special}{b}

Some text or a lot of text.

\clearpage
\end{Special}
```

It is also possible to add the `\clearpage` to the definition of the `Special` environment if you define this environment yourself. If you use an existing environment you may use a LATEX environment hook to inject a `\clearpage`, for example in the case of the `appendices` environment:

```
\AddToHook{env/appendices/end}{\clearpage}
```

Of course this will always cause a page break. If you don’t want a page break at the end of your environment, you will have to decide what to do with the page that is partially filled with the special environment and partially with the following text. Which page style to use: the Special page style or the normal page style? If you do nothing it will be the normal page style. If you still want the Special page style, you can put a `\thispagestyle{Special}` at the end of the environment. Again, at the use of the environment, at the definition, or using a hook.

Example 26G (c)

```
\begin{Special}{c}

Some text or a lot of text.

\thispagestyle{Special}
\end{Special}
```

Note, however, that this only works if the `Special` page style is defined outside of the environment, as is done in this example. However if the `Special` page style was defined inside the environment, it will have disappeared at the end of the page, and LaTeX will silently ignore it. It doesn't even give an error message. The following pages will then get the normal header again.

29 Fancyhdr hooks

LaTeX has a system of *hooks* since the 2020/10/01 release. This allows packages and classes (and other LaTeX software) to define points in its code where other LaTeX code can insert a piece of code. For more details, see *The LaTeX Companion, Third Edition*, part I, pp. 671 ff. or the documentation that can be read with the command '`texdoc lthooks-doc`'.

Fancyhdr version 4.5 or later defines a number of hooks to be executed at the beginning or end of the header and/or footer, if your LaTeX version supports it. The hooks are defined in mirrored pairs, which means the second one of the pair is executed in the reverse order compared to the first one (see the hooks documentation).

`fancyhdr/before`, `fancyhdr/after` these are executed before the header or footer is constructed, and after the header or footer is finished, respectively.

`fancyhdr/head/begin`, `fancyhdr/head/end` these are run at the beginning and the end of the header construction, respectively

`fancyhdr/foot/begin`, `fancyhdr/foot/end` these are run at the beginning and the end of the footer construction, respectively

The interaction of the hooks and the `\fancyhfinit` code described on page 46 in section 28.1 with the construction of the header and footer is as follows: for the header construction

- first the `fancyhdr/before` hooks are run, then the `fancyhdr/head/begin` hooks, then the `\fancyheadinit` code. Then the header is constructed. Finally, the `fancyhdr/head/end` hooks are run followed by the `fancyhdr/after` hooks.
- For the construction of the footer, it is similar, just replace `head` by `foot`.
- Note that between the construction of the header and the footer, LaTeX builds the body of the page. This process consists mainly of putting boxes next to each other, and fancyhdr does not interfere with this, and neither should the hook code.

The reason there are separate `fancyhdr/before` and `fancyhdr/after` hooks and the `head` and `foot` hooks, is

1. If you want to use the same hooks for headers and footers, use the `fancyhdr/before` and `fancyhdr/after` hooks. This prevents you to have to specify the same hook code twice.

2. If you want to have different hooks for the header and footer, use the `head` and `foot` hooks.

The `after` and `end` hooks are meant to undo changes made in the `before` and `begin` hooks, respectively. If the hooks make only local changes (which is recommended), the TEX grouping mechanism will take care of this, so you can leave out the `after` and `end` hooks in that case.

At first sight it may seem that the `\fancyhfinit` mechanism is no longer useful with the introduction of hooks. One reason it exists is that hooks were not available at the time it was introduced, and for compatibility reasons it remains. However, there are some significant differences between the `\fancyhfinit` mechanism and the hook mechanism, so you should choose carefully which one to use.

- Hooks are global, but the `\fancyhfinit` declarations are local. That is, if `\fancyhfinit` (or its siblings `\fancyheadinit` or `\fancyfootinit`) are given in a TEX group, they last until the end of the group. They will disappear outside of the group, or be reset to the value they had outside of the group.
- `\fancyhfinit` is meant to be used by the user who writes the document, i.e., it is meant for the current document. `\fancyhfinit` *should not be used by package or class writers and similar. They should use the hooks mechanism.* On the other hand the user can also use hooks in the document instead of, or in addition to the `\fancyhfinit` mechanism.
- Hooks can be added multiple times, but the `\fancyhfinit` code can only be given once (i.e., a new one overwrites the previous one).
- The `\fancyhfinit` code is stored in a *closed* page style (see section 16). Hooks are not.
- `\fancyhfinit` has no corresponding `exit` function, so if you need some code to be executed after the construction of the header or footer, you have to use hooks.
- The hooks can also be given if fancyhdr is not used. This can be used as a precautionary measure in packages and classes that may have a bad interaction with fancyhdr otherwise. If fancyhdr is not used in a document, the hooks don't do anything.

NOTE: In fancyhdr version 4.3 and later, paragraph hooks will not work inside fancyhdr headers and footers to avoid unwanted interactions with the main text. However, in version 5.1 and later, the hooks defined in the main text will still be disabled in the headers and footers. But it is possible to use paragraph hooks locally in headers and footers. Se the example below.
NOTE: This is experimental and may change in the future.

parahooks

```
\AddToHook{fancyhdr/before}{%
  \AddToHook{para/begin}{XXXX}%
}
```

30 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:

Example 27

```
\usepackage{extramarks}
...
\pagestyle{fancy}
\fancyhead[L]{\firstleftxmark} % = \firstxmark
\fancyfoot[R]{\lastrightxmark} % = \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.

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 top-marks 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.

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 33 for another example of this technique.

Example 28

```
\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 `continued` for it[17].

The first thought might be to use `\afterpage{\firstxmark}`. But the marks can only be used in the headers and footers, not in the running text [18]. Moreover, we need the value that will become `\firstxmark` (=`\firstleftxmark`) on the next page, but on the current page it will be in `\lastleftxmark`.

Then you might think that the `\afterpage` command could be put in a header or footer, but unfortunately it appears that then the timing is wrong. The `\afterpage` text will appear one page too late.

So what we do is, we put the `\lastleftxmark` in a variable during the footer processing and then use this variable in `\afterpage`. As the footer processing is done inside a TeX group, we must use a global definition. Also the mark must be expanded so that we get the contents of the mark in our variable and not just the name. We can do this with the primitive TeX command `\xdef`. There is no LaTeX 2_ε command for this.

First we give a simple (but incomplete) solution.

Incomplete!

```
\newcommand\ContiText{}
\fancyhead[L]{Example 29a}
\fancyhead[R]{\rightmark}
\fancyfoot[R]{\lastrightxmark}
\fancyfoot[L]{\xdef\ContiText{\lastleftxmark}}
```

---

[17] In the example files for examples 27 and 28 this is also done; it is just not documented here.

[18] NOTE: This used to be different in extramarks version 4 and earlier.

```
\newenvironment{continued}{%
  \par
  \extramarks{}{}%
  \extramarks{\noindent Continued\ldots\\[1ex]}%
             {Continued on next page\ldots}%
  \afterpage{\ContiText}%
  \ignorespaces
 }{%
  \unskip
  \extramarks{\noindent Continued\ldots\\[1ex]}{}%
  \extramarks{}{}\par
}
```

The header contains document information: the name of the document on the left and the section title on the right. The footer contains the "Continued" information like in the previous examples. The `\extramarks` contain essentially the same information as in the previous examples, just formatted a little differently. But the `\...leftxmark` is not put in the header, but is eventually used as the argument in `\afterpage` so that it will appear at the top of the next page body. This is also the reason for the `\\[1ex]`, to separate it from the rest of the page text.

Note how we use `\ignorespaces`, `\unskip` and `%` to prevent unwanted spaces to creep into the text.

However, there are some problems with this simple solution:

1. If the block spans more than one page boundary, the `\afterpage` is not repeated on the following page breaks (`\afterpage` only applies to the next page). So on these pages the "Continued" header will be missing.

   We can solve this by repeating the `\afterpage` command in the `\afterpage` text. To do this we have to put it in a macro (`AP` stands for `afterpage`):

   ```
   \newcommand{\setAP}{\afterpage{\ContiText\setAP}}
   ```

   There is a disadvantage that the `\afterpage` will be continued on all pages after the block has ended. But as `\lastleftxmark` will be empty then, no harm will be done. However, the following subsection (30.1) will give a solution that stops this repetition.

2. If the page break comes out such that the beginning of the block is pushed to the next page, but the `\afterpage` is given while LaTeX was still at the previous page, the `\afterpage` text will be inserted before the block begins.

   Fortunately the `\lastleftxmark` on this page is empty, so the `\afterpage` on this page is essentially harmless, and because we have it made repeating by the previous point, it will be picked up at the proper place.

3. If there is more than one `continue` block on the same page (with the last one crossing the page boundary) there will be an `\afterpage` for each block, thereby repeating the "Continued" text multiple times at the top of the page. Therefore we should start the `\afterpage` only once, not once for each block. As the `\afterpage` is repeated on each page by the previous solution we don't need multiple starts of `\afterpage`.

We could do this by inserting the `\afterpage` command before the first block instead of inside it, but that is error-prone.

The solution is to define a command `\startAP` that sets the `\afterpage` command, and then redefines itself do do nothing. Because the `\startAP` is called inside a TeX group (the `continued` environment) we must do a *global* redefine. LaTeX 2ε does not have a command for this, so we use the low-level TeX command `\gdef` for this.

```
\newcommand{\startAP}{\setAP\gdef\startAP{}}
  . . .
\newenvironment{continued}{%
    . . .
    \startAP
} . . .
```

We also put some thick black rules around the environment. And because the text for the left mark is used twice we put that in a macro `\LM`. The order of the commands is chosen such that the 'Continued' marks don't go to the wrong page. This makes the total solution like this:

Example 29a

```
\newcommand\ContiText{}
\newcommand{\LM}{\noindent\hl{Continued from previous page\ldots}\\[1ex]}
\newcommand{\setAP}{\afterpage{\ContiText\setAP}}
\newcommand{\startAP}{\setAP\gdef\startAP{}}

\fancyhead[L]{Example 29a}
\fancyhead[R]{\rightmark}
\fancyfoot[R]{\lastrightxmark}
\fancyfoot[L]{\xdef\ContiText{\lastleftxmark}}

\newenvironment{continued}{%
  \par\startAP
  \extramarks{}{}%
  \noindent\rule{\textwidth}{1mm}%
  \extramarks{\LM}{Continued on next page\ldots}%
  \\*\ignorespaces
}{%
   \unskip\noindent\rule{\textwidth}{1mm}%
   \extramarks{\LM}{}%
   \extramarks{}{}\par
}
```

30.1 More sophisticated solutions

In this subsection we present some more sophisticated, and therefore a little more tricky solutions and variations to the previous example. If you want to avoid that trickery, you can just skip this subsection.

First we change the example such that the sequence of `\afterpage` invocations will stop as soon as possible. We do this by not using a fixed text as argument for `\afterpage` but by using a macro `\APcommand` as argument. When we want to stop the sequence of `\afterpage` calls, we make this macro empty. To get a proper timing we reset this macro in the right-hand footer field when this is empty, which indicates that we are outside of a 'Continued' block.

We must then take care of restarting the `\afterpage` sequence when a new 'Continued' block is started, and making sure that we don't get more than one such sequence activated. We do this by changing `\startAP` such that it only start an `\afterpage` if `\APcommand` is empty.

Example 29b

```
 1 \newcommand\ContiText{}
 2 \newcommand*{\LM}{\noindent Continued from previous page\ldots\\[1ex]}
 3 \newcommand*{\APcommand}{}
 4 \newcommand*{\setAPcommand}{\gdef\APcommand{\ContiText\setAP}}
 5 \newcommand*{\clearAPcommand}{\gdef\APcommand{}}
 6 \newcommand*{\setAP}{\afterpage{\APcommand}}
 7 \newcommand*{\startAP}{\ifx\APcommand\empty\setAPcommand\setAP\fi}
 8
 9 \fancyhead[L]{Example 29b}
10 \fancyhead[R]{\rightmark}
11 \fancyfoot[R]{\lastrightxmark}
12 \fancyfoot[L]{\xdef\ContiText{\lastleftxmark}}
13 \fancypagestyle{plain}{\fancyhead{}\renewcommand{\headrule}{}}
14
15 \newenvironment{continued}{%
16   \par\startAP
17   \extramarks{}{}%
18   \noindent\rule{\textwidth}{1mm}%
19   \extramarks{\LM}{Continued on next page\ldots}%
20   \\*\ignorespaces
21 }{%
22    \unskip\noindent\rule{\textwidth}{1mm}%
23    \extramarks{\LM}{}%
24   \extramarks{}{\protect\clearAPcommand}\par
25 }
```

We have numbered the lines for easy reference. The changes are in the red lines (3–7 and 24).
3. Here we define `\APcommand`.
4, 5. These are commands to set en clear `\APcommand`, respectively.
6. The `\afterpage` now uses `\APcommand` as argument.
7. `\startAP` now checks if `\APcommand` is empty, and if it is, it first fills `\APcommand` with the required value and then starts a new `\afterpage` (with the `\setAP` command). When `\APcommand` is not empty this means that an `\afterpage` is already active.
24. In the right part of the marks we now call `\clearAPcommand` to clear our variable `\APcommand`. This effectively stops the `\afterpage` sequence.

Note 1. We use `\gdef` to change `\APcommand` because these occur inside a TeX group (`continued` environment and footer). With `\renewcommand` they would be local to these groups but we need them outside of these groups, therefore we use `\gdef` to make the change globally.

Note 2. We define `\APcommand` with `\newcommand*` rather than `\newcommand` to make it compatible with `\gdef`. Without the `*` it would be compatible with `\long\gdef`, but then it would not compare equal to `\empty` in line 7. For the other definitions it does not make a difference, but it looks nicer to also use it there.

Note 3. In line 24 we use `\protect` to delay the expansion of `\clearAPcommand`. The marks in `\extramarks` are expanded at the time they are given, so that they can pick up section numbers and titles and similar information at that point. However, `\clearAPcommand` should not be expanded at that moment, but when it is used in the footer. That is exactly what `\protect` does.

Note 4. We test the value of `\APcommand` with `\ifx`, not with `\ifthenelse` from the ifthen package. The latter completely expands its parameters, and because `\APcommand` has a recursive definition when it is not empty, that would cause TEX to fail. We only want to check the definition of `\APcommand`, not its expansion.

Note 5. For debugging we can add some text in the `\afterpage` command in line 6, to see the difference between an empty `\afterpage` and no `\afterpage` at all. Similarly we can add some text in the footer in line 24, to see where the `\clearAPcommand` is called.

Another use

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 header or footer fields, like in example 29a. For example if you want to add something *after* the broken block of text you can use the following:

Example 29c

```
\newcommand{\ContiText}{}
\fancyhead[L]{Example 29c}
\fancyhead[R]{\rightmark}
\fancyfoot[R]{\lastrightxmark}
\fancyfoot[L]{\xdef\ContiText{\lastleftxmark}}
\fancypagestyle{plain}{\fancyhead{}\renewcommand{\headrule}{}}

\newenvironment{continued}{%
  \par
  \extramarks{}{}%
  \noindent\rule{\textwidth}{1pt}%
  \extramarks{\\[1ex]\noindent\textbf{[Continued
              from previous page]}}{Continued on next page\ldots}%
  \\*\ignorespaces
}{%
   \unskip\noindent\rule{\textwidth}{1pt}%
   \extramarks{}{}%
   \ContiText\par
}
```

Now if the block crosses a page boundary, the `\lastleftxmark` has the text that should be put under the block. In the `[L]` footer field we put this information in the macro `\ContiText`, and this is typeset after the block ends. If the block doesn't cross the page boundary, this text is empty.

NOTE: This example is not completely safe; there can still be timing issues. For example, when the end of the block has already been typeset, including an empty value of `\ContiText`, but then is pushed to the next page. So you have to be very careful in using this kind of mechanism.

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

31 Page styles for Table of Content, List of Figures, Bibliography, etc.

Some special sections of a documents, such as the Table of Contents, List of Figures/Tables, Bibliography, Index, and similar ones sometimes cause difficulties if you want them to have special page styles, especially if you also want the first page of these to have a special page style.

Suppose you have defined a special page style `tocstyle` for the Table of contents. The Table of contents is generated by the command `\tableofcontents` and it can be several pages long, all generated by this simple command. So if you want this to have the page style `tocstyle`, you must give the command `\pagestyle{tocstyle}` before the `\tableofcontents`. But then you have to make sure that the previous page (for example a title page) doesn't get this page style too. As we have seen before we can do this by inserting a `\newpage` first. Like

```
\newpage
\pagestyle{tocstyle}
\tableofcontents
```

If we use a chapter based documentclass, like the standard classes report and book, with this setup the first page of the Table of Contents, and similar parts of the document will still use the `plain` page style. Usually this is the best choice, but there may be cases where you want these also to use the `tocstyle` page style (or another special page style). The `plain` page style is set by a `\thispagestyle{plain}` command embedded in a `\chapter*` command that is used in `\tableofcontents`. So it is not easy to overwrite. It can be overwritten by a `\thispagestyle{tocstyle}` command, but that must be given after the `\chapter*` command, but before the first page of the Table of Contents is finished. So in fact we must break in into the `\tableofcontents` command. The other special parts have similar challenges. In this section we give a number of solutions.

The first solution applies to the Table of Contents and List of Figures/Tables. We can add additional code in these lists with the `\addtocontents` command. We can use this to insert a `\thispagestyle` as the first entry,

tocpagestyle (a)

```
\newpage
\pagestyle{tocstyle}
\addtocontents{toc}{\protect\thispagestyle{tocstyle}}
\tableofcontents
```

and similar for the List of Figures (use `lof`) and List of Tables (use `lot` instead of `toc`). The `\protect` is necessary to prevent the `\thispagestyle` to be executed too early.

NOTE: If you are using the package tocloft, some of the solutions given here for the Table of Contents and List of Figures/Tables may not work (but others may). This is because this package changes the layout of these. In particular, the first page of these will by default have page style `plain`, even in a documentclass that has no chapters, like article. However, the package has a command to set the page style for these first pages. This will set it for all three.

```
\tocloftpagestyle{tocstyle}
```

Alternatively, you can use `\usepackage[titles]{tocloft}` which will keep the original LaTeX code, so then the solutions mentioned in this section will apply.

For the Bibliography and Index the above solutions cannot be used. So we need a different solution for these.

For the Bibliography we can use the LaTeX hook system (available since the 2020/10/01 LaTeX release). The command `\bibliography` reads the file ⟨*jobname*⟩`.bbl`, which contains a `thebibliography` environment, that contains a `\bibitem` for each reference. The `\begin{bibliography}` sets up the chapter header and starts a `list` environment. Unfortunately, the LaTeX hook system doesn't have a hook that is executed just after this point (i.e., before any bibliography items are added).

However, we can add a hook at the first `\bibitem` as follows:

tocpagestyle
(c)

```
\AddToHookNext{cmd/bibitem/before}{\thispagestyle{tocstyle}}
```

An alternative would be to use an '`after`' hook on the `\thebibliography` command. This command is internally used to do the setup work for the `thebibliography` environment, but this is an implementation detail, so theoretically this could change in a future implementation, or an implementation in a different documentclass. But this is the hook that is executed at the right moment: after the setup, and before any bibliography items are added.

tocpagestyle
(c)

```
\AddToHook{cmd/thebibliography/after}{\thispagestyle{tocstyle}}
```

For the Index there is no hook that can be used in a similar manner. Maybe we could use a hook in the `\item` command that is used for the index items, but this is just a too general command that is not exclusively used in the Index. However, we can use a hook on the `\thispagestyle` command that is used in the internal `\chapter*` command in the `\printindex` command (the `\thispagestyle` that gives us these problems). We just use an '`after`' hook to insert another `\thispagestyle` that replaces the built-in one. In fact we could have used this same solution for each of the cases mentioned in this document.

tocpagestyle
(d)

```
\newpage
\pagestyle{tocstyle}
\AddToHookNext{cmd/thispagestyle/after}{\thispagestyle{tocstyle}}
\printindex
```

Finally, we can use the solution from section 15 (redefining page style `plain`). For example:

```
\fancypagestyle{plain}[tocstyle]{}
```

But this would also change the `plain` page style for the chapters in the normal text, which we don't want. So the page style `plain` should be reset in the main text. We can do this with the new (fancyhdr version 5) command `\fancypagestyleassign`. We can use this to 'save' the original `plain` page style and set it equal to `tocstyle`. And later we can reset `plain` to the saved page style.

tocpagestyle (b)

```
\fancypagestyleassign{origplain}{plain}
\fancypagestyleassign{plain}{tocstyle}
\listoftables % \tableofcontents / \listoffigures, etc.

% Here the main document text starts

\clearpage
\fancypagestyleassign{plain}{origplain}
```

All definitions for the page style (including the `\fancypagestyleassign` commands above) are local to the TEX group in which they are defined (see section 28.3). So we could eliminate the saving and restoring to `origplain` if we do the change in a group. For example:

```
{
\fancypagestyleassign{plain}{tocstyle}
% or use the older method \fancypagestyle{plain}[tocstyle]{}
\listoftables
}
```

After the group, page style `plain` has its original value. However, it is not advised to place large parts of your document inside a group.

32 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`[19] where $\langle n\rangle$ is the page number and that we use the graphics or graphicx package. To put the movie in the right-hand side bottom corner the following will work:

Example 30

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

---

[19] 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.

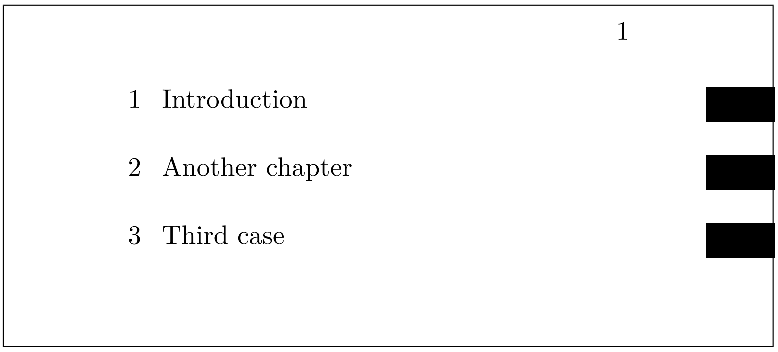
If the document is two-sided, it would be better to put them only on the odd pages, by specifying `\fancyfoot[RO]`.

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.

33 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 originally designed this for, there were 12 sections, so I made the blobs 18 mm apart, i.e., 9 mm blob separated by 9 mm white space. 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 chapter numbers are used to position the blobs, but you can replace this with any numeric value. See figure 5 for the resulting overview page.



1

1 Introduction

2 Another chapter

3 Third case

Figure 5: Thumb-index overview page

Example 31

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

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

\newcommand\lblob{%
  \begin{picture}(0,0)
```

```
    \put(-3,-\value{chapter}){\blob}
  \end{picture}%
  \thepage}

\pagestyle{fancy}
\fancyfoot{}

\newcounter{line}
\newcommand{\chapname}[1]{\addtocounter{line}{1}%
  \put(1,-\value{line}){\blob}
  % Adjust these numbers for the proper indentation
  \put(-5.5,-\value{line}){\Large \arabic{line}}
  \put(-5,-\value{line}){\Large #1}}

\newcommand{\overview}{%
  \begin{picture}(0,0)
    \chapname{Introduction}
    \chapname{Another chapter}
    \chapname{Third case}
    . . .
  \end{picture}
}
```

The overview page:
The page doesn't have 'contents' – all the visual contents is generated by the `\overview` command in the header

Example 31
(continued)

```
\fancyhead[L]{Overview}
\fancyhead[R]{\overview}
\mbox{}\newpage % This produces the overview page

% Front matter -- doesn't have blobs.

\fancyhead[RE]{\rightmark}
\fancyhead[RO,LE]{}
\fancyhead[LO]{\leftmark}

\pagenumbering{roman}
\thispagestyle{plain}
\tableofcontents
 . . .
\newpage

% Here the document begins

\pagenumbering{arabic}

% Now activate the blobs

\fancyhead[RO]{\rblob}
```

```
\fancyhead[LE]{\lblob}

% Page style 'plain' does not have the usual header,
% but it does have the blobs.

\fancypagestyle{plain}{%
  \fancyhead[RE,LO]{}
  \renewcommand{\headrule}{}%
}
```

34 Float placement

Note: This section is not about fancyhdr, but about page layout, especially about the placement of floats.

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:

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 with their default values:

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 default values are 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 continuing. If it is undesirable to have a page break 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 page break, 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.

A much more elaborate article about float placement by Frank Mittelbach appeared in 2014 in TUGboat[20].

35 Multi-page 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 generally a more difficult decision 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:

35.1 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

---

[20] Frank Mittelbach, *How to influence the position of float environments like figure and table in LATEX?*, TUGboat, Volume 35 (2014), No. 3, pp. 248–254.
https://www.latex-project.org/publications/2014-FMi-TUB-tb111mitt-float-placement.pdf
Also on Stackexchange:
https://tex.stackexchange.com/questions/39017/how-to-influence-the-position-of-float-environments-like-figure-and-table-in-lat

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}}
```

or

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

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

35.2 Figures

There isn't an equivalent "`longfigure`" solution, so for figures you will have to split 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. The first part would then get a `\caption`, the subsequent parts would be used without a caption, or a caption that will not go to the list of figures. 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`.

We give a series of possible solutions here, which can be found in Example 33.

First we include the `figure`s with the `[!htbp]` position option to give LaTeX maximum freedom to place them. This way we hope they keep them together, although there is no guarantee.

Example 33
(A)

```
\newcommand{\fakecaption}[2]{% #1 = figure label #2 = caption
  \par Figure~\ref{#1}: #2
}
\begin{figure}[!htbp]
  \centering
  \includegraphics[scale=0.5]{example-image-a}
  \caption[This is a multi-part figure] % For the list of figures
          {This is a multi-part figure (a)}
  \label{fig:first}
\end{figure}
\begin{figure}[!htbp]
  \centering
  \includegraphics[scale=0.5]{example-image-b}
  \fakecaption{fig:first}{This is a multi-part figure (b)}
```

```
\end{figure}
  . . .
```

There will probably be some of the normal text between the figure parts, unless they happen to fit perfectly on the page, which isn't very probable. But, what also can come between them is other floats, such as a `table`. We can prevent that previous floats intrude here by issuing a `\clearpage` command, but this will abruptly end the current page. As we have seen before, we can do better by including the `\clearpage` command in `\afterpage`, and we would also put the figures in the `\afterpage`. To keep the `\afterpage` command more tidy, it is advised to put the code for the figures in a macro, or in a file that is included with `\input`. For example:

Example 33
(B)

```
\newcommand{\myfigures}{%
  \begin{figure}[!htbp]
    \centering
    \includegraphics[scale=0.5]{example-image-a}
    \caption[This is a multi-part figure] % For the list of figures
            {This is a multi-part figure (a)}
    \label{fig:second}
  \end{figure}
  \begin{figure}[!htbp]
    \centering
    \includegraphics[scale=0.5]{example-image-b}
    \fakecaption{fig:second}{This is a multi-part figure (b)}
  . . .
}
  . . .
\afterpage{\clearpage\myfigures}
```

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

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

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

But if there is enough space left on a page, some of the text will go between the figures. Also, if there is still some figure part of a previous sequence that has not yet found a place, it will be forced out because of the `\clearpage` and the a new page will start, with the previous page not optimally filled.

So using `\clearpage` may also not be optimal. We could also try to put the figure parts only on float pages, so that no intervening text will come between them. This can be done by using the position parameter `[p]`. This could cause them to be pushed towards the back of the document. This is because float pages need to be reasonably full before they are generated. You could try to cure this for example by adding some `\vspace` to the last part, or by tweaking the `\floatpagefraction` parameter (see section 34 on page 64). To prevent previous floats to intrude in the float page, we also combine this with the `\afterpage` and `\clearpage`, as in the previous example, but this will probably push the figures even further towards the back.

Example 33 (C)

```
\newcommand{\myfigures}{%
  \begin{figure}[p]
    \centering
    \includegraphics[scale=0.5]{example-image-a}
    \caption[This is a multi-part figure] % For the list of figures
            {This is a multi-part figure (a)}
    \label{fig:third}
  \end{figure}
  \begin{figure}[p]
    \centering
    \includegraphics[scale=0.5]{example-image-b}
    \fakecaption{fig:third}{This is a multi-part figure (b)}
 . . .
}
 . . .
\afterpage{\clearpage\myfigures}
```

So maybe just use the previous example without `\afterpage` and `\clearpage`.

Example 33 (D)

```
\myfigures % (with the [p] placement)
```

The defects of the above approach are

1. It is clumsy to make the captions of all but the first part of the figure
2. It is hard to refer to the parts separately

For this the `subcaption` package comes to the rescue. First it has a `\ContinuedFloat` command to indicate that a figure is a continuation of a previous one, and therefore will not get a new number, and if you wish, neither a separate entry in the list of figures.

Second, it has a `\subcaptionbox` command and a `subfigure` environment for the parts, where a subcaption can be given, that can also have a `\label` to refer to in the document. The `\subcaptionbox` is a specialized `\parbox` but its *width* parameter is optional. The `subfigure` environment is a specialized `minipage`, so it has the same parameters.

These should be used inside a `figure` environment, so all the placement methods of the previous part (Examples 33 A–D) should still apply.

The `subfigure` environment has a `\subcaption` command for the subcaption; the `\subcaptionbox` has the subcaption (with its `\label` if desired) as its first argument. When more than one `\subcaptionbox` is horizontally next to each other, the subcaptions will be aligned.

In the following example (figure 6) we use a `\subcaptionbox` for the first two parts, which are together in a single `figure` environment. We use a `subfigure` environments for the other two, each one in its own `figure` environment. These use a `\caption[]{...}`. The empty optional argument `[]` causes the caption not to appear in the list of figures. The last subfigure (6d on page 70) has a label on the `\subcaption` that we refer to in this sentence.

Example 33
(E)

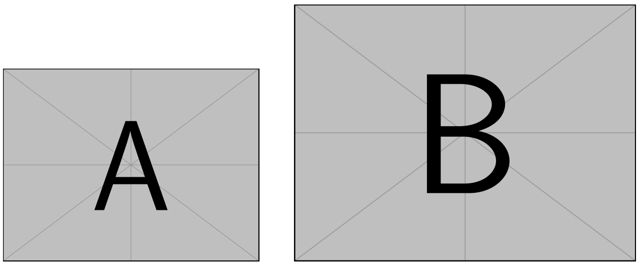
```
\begin{figure}[p]
  \centering
    \subcaptionbox{a subfigure in a \cs{subcaptionbox}}
      {\includegraphics[scale=0.3]{example-image-a}}
  \quad
    \subcaptionbox{another subfigure, also in a \cs{subcaptionbox}}
      {\includegraphics[scale=0.4]{example-image-b}}
  \caption{A figure with subfigures}
  \label{fig:subfigures}
\end{figure}

\begin{figure}[p]\ContinuedFloat
  \begin{subfigure}{\textwidth}
    \centering
    \includegraphics[scale=0.5]{example-image-c}
    \subcaption{subfigure}
  \end{subfigure}
  \caption[]{A figure with subfigures}
\end{figure}

\begin{figure}[p]\ContinuedFloat
  \begin{subfigure}{\textwidth}
    \centering
    \includegraphics[scale=0.5]{example-image}
    \subcaption{last subfigure}
    \label{subfig:last}
  \end{subfigure}
  \caption[]{A fake caption just for demo}
\end{figure}
```

36 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]`.

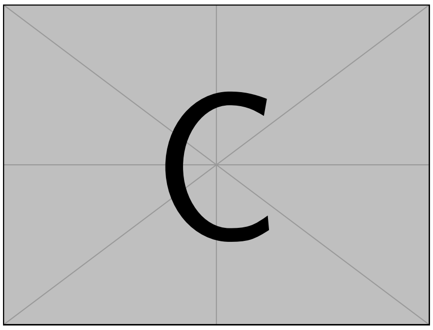
These commands for specifying the header or footer fields and their translation to the modern commands are given in table 1.

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.



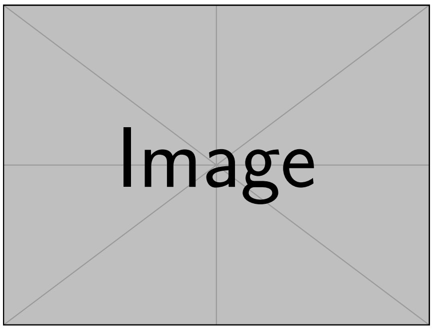
(a) a subfigure in a `\subcaptionbox` (b) another subfigure, also in a `\subcaptionbox`

Figure 6: A figure with subfigures



(c) subfigure

Figure 6: A figure with subfigures (cont.)



(d) last subfigure

Figure 6: A fake caption just for demo

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

`\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

`\fancyplain`

There was also a special page style `fancyplain` that could be used to define both the page style `fancy` and to redefine the page style `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 page style `fancyplain`. Nowadays you would just redefine page style `plain` with the `\fancypagestyle{plain}{xxxx}` command (see section 15).

`\plainheadrulewidth`
`\plainfootrulewidth`

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 page style `fancyplain`, not when you redefine page style `plain` with the `\fancypagestyle` command.

37 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-examples

38 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 (36).
  - Documentation expanded.
- Version 3.9a, June 30, 2017.
  - Updated contact information.
  - Restore `\newtoks\@temptokenb`
- Version 3.10, January 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`.

38.1 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 20 on page 32.)
  - 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 3.)
  - 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 13).
  - The `headings` and `myheadings` package options (see section 3).
  - The `\fancypagestyle` command has an optional parameter `[`⟨*base-style*⟩`]`.
- Version 4.0.1, Jan 28, 2021
  - Some documentation corrections, especially in sections 30 and 32.
- Version 4.0.2, May 9, 2022
  - Added `\leavevmode\ignorespaces` to each header/footer field. The `\leavevmode` prevents a bug when a field starts with a `\color` command. The `\ignorespaces` skips initial spaces in the parameter, as is usual in a `\parbox`, for backwards compatibility. However, there are some rare cases where spurious spaces can still show up in the header/footer fields. In that case the user will have to eliminate these.
- Version 4.0.3, May 18, 2022
  - Initialize `\@mkboth` in extramarks.sty so that it will pick up changes to `\markboth`.
- Version 4.1, Sept 6-Nov 9, 2022
  - Implement `twoside` package option to allow two-sided headers and footers in one-sided documents.
  - Make fancyhdr compatible with the document class newlfm.

  - Make `\nouppercase` compatible with newer definitions of `\MakeUppercase`.

- Version 4.2, April 19, 2024
  - Reset catcodes to their default values in order to facilitate `\input` in headers/footers when `verbatim` is active. (Issue # 8 https://github.com/pietvo/fancyhdr/issues/8.)

- Version 4.3, July 17, 2024
  - Changed `\f@nch@everypar`. If the LaTeX kernel has `expl3`, use `\tex_-everypar:D`, and reset `\par`, `\@@par` and `\endgraf` to their original TEX definitions, so that no paragraph hooks will intrude in fancyhdr code[21]. Therefore paragraph hooks will not work inside fancyhdr headers and footers to avoid unwanted interactions with the main text.

- Version 4.3.1, July 23, 2024
  - Also reset `\everypar` to its original TEX value `\tex_everypar:D` in `\f@nch@resetpar`, otherwise environments based on `\trivlist` will not work properly in fancyhdr headers and footers.

- Version 4.4, Nov 20, 2024
  - Add setting the new style marks for `\leftmark` (`2e-left`) and `\rightmark` (`2e-right` and `2e-right-nonempty`) in extramarks.sty.

- Version 4.5, Nov 21-30, 2024
  - extramarks: Don't redefine `\leftmark` and `\rightmark` in LATEX kernel 2025-06-01 and later.
  - fancyhdr: use a better method to disable paragraph hooks than the v4.3 code.
  - extramarks-v4 (legacy version): add commands `\extramarksleft` and `\extramarksright`.
  - fancyhdr: added hooks.

38.2 Changes in version 5

Version 5 adds several new features. Most notable is a new implementation of the extramarks package, which now has independent marks.

- Version 5.0, Feb 11, 2021-Jan 1, 2025
  - Shorten Warning message about `\headheight`/`\footskip` too large.
  - If the option `[nocheck]` is given, just keep quiet and don't change the `\headheight`/`\footskip` even if the `[compatV3]` option is given.
  - Added `\fancypagestyle*` variant.
  - Added command `\fancyhdrsettoheight`.
  - New implementation of package extramarks with fallback to extramarks-v4.
  - Mark the `compatV3` option deprecated.
  - Added command `\fancyfootalign`.
  - Added command `\fancyhdrbox` (section 14).

---

[21] See https://tex.stackexchange.com/q/691262/113546

  - Added command `\fancypagestyleassign` (section 16.1).
  - Added commands `\fancyheadwidth`, `\fancyfootwidth` and `\fancyhfwidth` (section 12).
  - Many documentation improvements.

- Version 5.1, Jan 4-6, 2025
  - Bug fix in extramarks.
  - Better code to save, clear and restore paragraph hooks in headers/footers.

- Version 5.1.1, Jan 7, 2025
  - Bug fix in save, clear and restore paragraph hooks in headers/footers.

- Version 5.2, Jan 14-Feb 7, 2025
  - Use official interface to reset the paragraph hooks.
  - Conditionally require package xparse in fancyhdr.
  - Require LaTeX version 2018-04-01 or later,
  - therefore cleanup some pre-2018 code.
  - Implement `\fancyhfwidth` etc. ⟨*alignment*⟩ option and the `*` form of these commands.
  - Documentation updates.

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.

39 Long chapter/section titles

Sometimes a chapter or section title is too long 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. Here I give four options to solve this problem.

39.1 Using optional arguments

As we have seen in section 17, the header info comes from the marks. So if we want the text in the header to be shorter we have to supply shorter marks. This can be done by giving these as optional arguments in the `\chapter` and `\section` commands.[22]

Example 34a

```
\chapter[This is a not so long chapter title]
        {This is a very long chapter title to see if we can give
         fancyhdr a shorter one that fits in the header}
. . .
\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 then 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.

39.2 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 17 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:

Example 34b

```
\chapter{This is a 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

---

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

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 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:

Example 34b
(continued)

```
\newcommand{\Section}[2]{%
            \section[#1]{#1\sectionmark{#2}}\sectionmark{#2}}
  . . .
\Section{This is a long section title that will not fit in
         the header}{Shortened section 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.

Example 34b
(continued)

```
\newcommand{\Sectionx}[3]{%
            \section[#3]{#1\sectionmark{#2}}\sectionmark{#2}}
  . . .
\Sectionx{This is another long section title that will not
            fit in the header}{Short section title 3}
           {This is the section title in the table of contents}
```

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

39.3 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.

Example 34c

```
\usepackage[fit]{truncate}
\fancyhead[LE,RO]{\nouppercase{%
         \truncate{0.5\headwidth}{\rightmark}}} % Section title
```

```
\fancyhead[LO,RE]{\nouppercase{%
         \truncate{0.5\headwidth}{\leftmark}}} % Chapter title
```

We don't have to make any changes to the chapter 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 that...

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`. A more sophisticated solution would be to check if one of the header parts is small enough and then truncate the other one for the remaining space.

39.4 Using `\fancyheadwidth`

For this solution we use the `\fancyheadwidth` command to give each part a little less than half of the `\headwidth`. This command is available in `fancyhdr` version 5.0 and later. Of course it is also possible to make asymmetric arrangements.

Example 34d

```
\setlength{\headheight}{35pt}
\fancyhead[LE,RO]{\nouppercase{\rightmark}} % Section title
\fancyhead[LO,RE]{\nouppercase{\leftmark}}  % Chapter title
\fancyfoot[LE,RO]{\thepage}
\fancyheadwidth[L,R]{0.48\headwidth}
. . .
\chapter{This is a very long chapter title to see if we can
         let fancyhdr fit it in the header}
\section{This is a very long section title that will not fit in the header}
```

We don't have to make any changes to the chapter and section titles. 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
title to see if we can let fancyhdr fit it in 1.1. This is a very long section title that
the header will not fit in the header

You may not like that the two parts of the header are aligned at the bottom, which is the default for the header. We can use the second optional parameter of `\fancyheadwidth` to specify a different alignment (in `fancyhdr` version 5.2 or later). See the following example.

Example 34e

```
\setlength{\headheight}{35pt}
\fancyhead[LE,RO]{\nouppercase{\rightmark}} % Section title
\fancyhead[LO,RE]{\nouppercase{\leftmark}}  % Chapter title
\fancyfoot[LE,RO]{\thepage}
```

```
\fancyheadwidth*[L,R][tj]{0.48\headwidth}
\fancyfootwidth*[LE,RO][-c]{0.1\headwidth}
```

We use `t` for the vertical alignment, and also `j` (justified) for the horizontal alignment. Just for fun we use the `*` form of the command, and also add `\fancyfootwidth*`. This puts the page number under the page a little bit away from the edge of the text.

The resulting header looks like:

Chapter 1. This is a very long chapter title to see if we can let fancyhdr fit it in the header	1.1. This is a very long section title that will not fit in the header

Please note, that these solutions also have disadvantages. The header must be quite tall, and when a smaller title is used there is a gap between the title and the line under the header.

40 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 page style `myheadings` (as well as `headings`) redefines the `\chaptermark` and `\sectionmark`, so when you return to page style `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 page style `fancy`.

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

41 Can I use **fancyhdr** with the **beamer** class?

The beamer class has its own provisions for headers and footers with the `headline` and `footline` templates. The advantage of these is that they blend well with the beamer theme in use.

Still people sometimes ask if fancyhdr can be used for header and footers because they are more familiar with this. I would advice to use the standard beamer features if possible, but actually it isn't difficult to use fancyhdr if you take provisions that the header and the footer don't interfere with the beamer layout. This can be done with

```
\setbeamertemplate{headline}{\vspace{⟨headheight⟩}}
\setbeamertemplate{footline}{\vspace{⟨footskip⟩}}
```

Note that beamer sets `\headheight` and `\footskip` to its own vales, so it doesn't make sense to set these in your document. Instead you supply the desired values with `\setbeamertemplate` as above. Also it is advised to add `\fancyfootalign{0pt}` to prevent the footer to be too close to the bottom edge; see section 20 on page 34.
Here is a complete example:

with-beamer

```
\documentclass{beamer}
\usepackage{graphicx}
\usepackage{fancyhdr}
\pagestyle{fancy}
\fancyhead[L]{\includegraphics[width=0.1\textwidth]{example-image}}
\fancyhead[R]{Course Name}
\fancyhead[C]{\textbf{Subject}\\Author}
\fancyfoot[L]{LEFT page footer}
\fancyfoot[R]{RIGHT page footer}
\fancyfoot[C]{{\thepage}}
\fancyfootinit{\tiny}
\renewcommand{\footrulewidth}{0.4pt}

\setbeamertemplate{headline}{\vspace{30pt}}
\setbeamertemplate{footline}{\vspace{14pt}}
\fancyfootalign{0pt}

\begin{document}

\begin{frame}{Subject Title}
Text of the slide
\end{frame}

\end{document}
```

42 I want the first section and the first subsection in my headers

A question that is regularly asked (e.g., on `tex.stackexchange.com`[23]) is how to get both the first section title and the first subsection title in the headers in the article documentclass. Unfortunately, traditional LaTeX (releases before November 2022) can't give you the first subsection on the page. There are two problems:

- Traditional LaTeX uses left marks for the section title and right marks for the subsection title. But it only has commands to extract the last left mark (`\leftmark`) and the first right mark (`\rightmark`). This means that if there are two or more sections on the page you get the last one, which can be counter-intuitive. The newer

---

[23] See for example https://tex.stackexchange.com/q/586066/113546

LaTeX releases (November 2022 or later) have a command (`\FirstMark{2e-left}`) to get the first of the left marks, however. **We assume in the following code that your LaTeX is recent enough.**

- LaTeX uses `\markboth` at a section title in the article class. This also sets an empty right mark. So in some cases you would get an empty subsection title in the header. Also a `\part` command issues a `\markboth{}{}`, so it generates empty marks too. To avoid this, LaTeX now has an additional mark `2e-right-nonempty` in which only the non-empty right marks are saved. This is also set by `\markboth` and `\markright`.

If there is no `\section` command on the page, it 'inherits' the last section title of a previous page. Similarly for subsections. This gives us the following code:

Example 35 (basic)

```
\usepackage{fancyhdr}
\pagestyle{fancy}
\fancyhead[L]{\FirstMark{2e-left}}
\fancyhead[R]{\FirstMark{2e-right-nonempty}}
\fancyfoot[C]{\thepage}

\begin{document}
\section{Section One}
\subsection{Subsection One}
  . . .
```

This mostly solves the problem, but it has some undesirable properties, see below. Therefore, some refinement is possible. But we need an extra mark for this. We are also going to use new marks, instead of using the standard marks to get more control. We need two marks for this, one for the section title, and one for the subsection title. We call these `section` and `subsection` respectively. We replace the marking code above with the following, which gives the same result:

Example 35 (new marks)

```
\usepackage{fancyhdr}
\pagestyle{fancy}
\NewMarkClass{section}
\NewMarkClass{subsection}
\fancyhead[L]{\FirstMark{section}}
\fancyhead[R]{\FirstMark{subsection}}
\renewcommand{\sectionmark}[1]{%
    \InsertMark{section}{\thesection. #1}}
\renewcommand{\subsectionmark}[1]{%
    \InsertMark{subsection}{\thesubsection. #1}}
```

Now when you have a `\section` command on a page, but it doesn't have a subsection for an extended length, then, the previous subsection title is 'inherited' on this page. This may be sub-optimal, because it combines the title of section n with a subsection title that belongs to a previous section, so the subsection isn't even present on the page, which looks unnatural. You may want to suppress the subsection title in the right header in this case. With the new LaTeX marks this is possible.

What we want is essentially the following:

1. If there is at least one subsection on the page, use the first one.
2. Otherwise, if the previous page ended in a subsection (i.e., the page break was inside a subsection), use that subsection title.
3. Otherwise (the page break was inside a section that had no subsections thus far), leave the right header empty.

The first test can be done by comparing the subsection 'topmark' and 'firstmark'. The topmark is the last mark from the previous page (which might even have been inherited from an earlier page). If there is no subsection mark on the current page, firstmark is made equal to topmark. If there is a subsection mark on the page, firstmark will be a different mark. LaTeX now has a command to compare the marks, like this:

`\IfMarksEqualTF{⟨mark⟩}{⟨pos1⟩}{⟨pos2⟩}{⟨true code⟩}{⟨false code⟩}`

where the *pos* arguments are `top`, `first` or `last`. Please note that this test makes only sense in a header or footer[24]. So if we use

`\IfMarksEqualTF{subsection}{top}{first}`...

the ⟨`true code`⟩ is executed when there is no subsection title on the current page, and the ⟨`false code`⟩ when there is at least one.

However, there is no command to see if the page break is at the section or subsection level, because the two are independent. And we cannot use LaTeX variables for this, because of the asynchronous processing of the page breaking. But we can do this if we introduce a new mark '`which`', that is used by both `\section` and `\subsection`. We let `\sectionmark` put "0" in it, and `\subsectionmark` "1". The lastmark of '`which`' on a page then indicates if the page ends within a subsection or not. And as the topmark on the following page is the same as lastmark on the previous page, we can use this topmark to see whether the text at the beginning of the page comes from a section or a subsection (topmark $= 0$ means section, 1 means subsection).

So the interim new code for the right header becomes the following.

```
if there is no subsection on the page
then
    if the previous page ended in a subsection
    then use (inherit) that subsection
    else use an empty header
    fi
else (there is at least one subsection on the page)
    use the first subsection
fi
```

We use the package `ifthen` for the test.

Example 35
(continued)

```
\NewMarkClass{which}  % Preamble
\InsertMark{which}{0} % Initialize so there is always a valid value

\fancyhead[R]{%
  \IfMarksEqualTF{subsection}{top}{first}
    {% no subsection mark on this page
      \ifthenelse{\TopMark{which}=1} % previous page ended in a subsection
        % then use (inherit) that subsection
        {\TopMark{subsection}}%
        {}% otherwise empty right header
```

---

[24] Or more generally, when LaTeX's page building is active.

```
    }
    {% there is a subsectionmark on the page, use it
      \FirstMark{subsection}%
    }%
}
\renewcommand{\sectionmark}[1]{%
  \InsertMark{section}{\thesection. #1}%
  \InsertMark{which}{0}%
}
\renewcommand{\subsectionmark}[1]{%
  \InsertMark{subsection}{\thesubsection. #1}%
  \InsertMark{which}{1}%
}
```

There is one caveat, however. If the page begins immediately with a section title, the topmark may indicate that the previous page ended with a subsection, but that subsection did not extend past the page break. So we want to suppress the 'inheritance' of the subsection if there is a section title at the top of the page and the page contains no subsection title. The information whether a section title is at the top of the page is not available in the marks, so we need some other way to detect this.

TEX has two variables that can help us, `\pagegoal` is the vertical size that is available on the page, and `\pagetotal` is the amount we have used so far[25]. So if `\pagetotal`=`0pt` we are at the top of the page, otherwise somewhat further down. We can use this information to communicate to the header that no inheritance should take place. In reality, sometimes there is already a small amount of white space on the page, so the test should be less strict than `\pagetotal`=`0pt`. We might even choose to not inherit the subsection title if only a few lines of the previous subsection are present at the top of the page. Or maybe your design asks for no inheritance if the subsection at the top of the page before the section header is smaller than 1/3 of the page. The test would then be `\pagetotal`<0.33`\pagegoal`. In the code below we choose a few lines as the limit.

We have to do the test at the beginning of the `\section` command processing. This can be done with a LATEX *hook*.

The next question is how to communicate the fact that the section starts at (or near) the top of the page to the header. A simple way to do this could be to set the '`which`' mark to a different value, e.g, −1 instead of 0. However, below we will present a better way.

```
\AddToHook{cmd/section/before}{%
  % use whatever test your design requires
  \ifthenelse{\lengthtest{\pagetotal<4\baselineskip}}
    {indicate section at top of page}
    {indicate section somewhere else}
}
```

There is another situation where a section title could end up at the top of the page: when it is processed at the bottom of the page, but there isn't enough space left to place it there. It will then be pushed to the next page. In that case the `\pagetotal` will be in the neighbourhood of `\pagegoal`. However, it is very difficult to find a proper value for

---

[25] This doesn't include floats and footnotes.

the cutoff of `\pagetotal`. In many cases it will be too small or too big, depending on the size of the section title. So we need a different way.

If the section title is pushed to the next page, the page number of the page where the section title is processed will be different from the page number that is current when the header is created. (We assume that all page numbers are different.) So by storing the page number atvthe time of processing the section title, and then comparing it in the header with the `page` counter, we can check that the header was moved to another page from an earlier page. This doesn't have to be **the** next page, as there could be float pages in between. The test has to be done while we are in the header, not while we are processing the section title. And it is not enough to store the page number in a variable because there can be subsequent section titles on the next page and these would overwrite the saved value. So it must be stored in yet another mark. Let's call this mark `whichpage`. We can save this in the `\sectionmark` command.

Now, in the right header, we can check if `\FirstMark{whichpage}` is equal to `\thepage`. If they are different, the section title was pushed across a page boundary, so it is at the top of the page. Note that this test makes only sense if there is a section title on the page, so we have to include a test for this also. This can be done with `\IfMarksEqualTF{section}{top}{first}`. A compact way to do this is:

```
          \equal{\IfMarksEqualTF{section}{top}{first}
              {\thepage}{\FirstMark{whichpage}}}{\thepage}}%
```

So if there is no section on the page, we don't compare `whichpage` to `\thepage`, but instead compare `\thepage` with itself, which gives *true*, i.e. no section at the top of the page.

We use a string comparison for the page number, not a numerical comparison, because page numbers can be non-numeric, for example roman numbers, which can't be compared numerically.

This gives us now a way to indicate the previous case for a section title at the top of the page: Just put a value in the `whichpage` mark that is different from the actual page number. But as the test is done in the hook, and the setting of the mark in `\sectionmark`, we must communicate this through a variable. Let's call this variable `\whichpage`. I have chosen to just put the text "top" in front of the page number to make it different, but any value that can't be a page number would be good.

Example 35
(continued)

```
\NewMarkClass{whichpage}
\newcommand\whichpage{}

\AddToHook{cmd/section/before}{%
  % use whatever test your design requires
  \ifthenelse{\lengthtest{\pagetotal<4\baselineskip}}
    {\renewcommand{\whichpage}{top\thepage}}%
    {\renewcommand{\whichpage}{\thepage}}%
}

\renewcommand{\sectionmark}[1]{%
  \InsertMark{section}{\thesection\ #1}%
  \InsertMark{which}{0}%
  \InsertMark{whichpage}{\whichpage}%
}
```

In the example 35 file you can see this in section 6.

The final code for the right header will now be:

Example 35 (continued)

```
\fancyhead[R]{%
    \IfMarksEqualTF{subsection}{top}{first}
      {% no subsection mark on this page
        \ifthenelse{%
          % previous page ended in a subsection
          \TopMark{which}=1\and
                 % Is there a section on the page?
          \equal{\IfMarksEqualTF{section}{top}{first}
                {\thepage}{\FirstMark{whichpage}}}{\thepage}}%
          {
            % use previous subsection unless suppressed by section on top
            \TopMark{subsection}%
          }
        % otherwise use empty right header
        {}%
      }
      {% if there is a subsectionmark on the page, use it
        \FirstMark{subsection}%
      }%
}
```

In the actual code in the Example 35 file there is also some debugging code added.

42.1 The headers in this document

Now we will describe how the headers in the `fancyhdr` documentation (this document) are constructed. This documentation consists of a number of sections, some of which have one or more subsections. There is only one title in the header, which is the title of the [sub]section that is current at the top of the page. In other words, the last one on a previous page. This asks for a ‘top mark’.

Because both sections and subsections are treated the same, we use a single mark for both, and we put the common code in macros. We could have chosen a new mark for the titles, but LaTeX’s standard right mark is sufficient for this[26].

When a [sub]section title is at or near the top of the page, we will use that title in the header instead of the one from the previous page, because this look more natural (the previous one does have no or very little text on the current page), just like in the previous example (35). We use the same mechanism, with a mark `whichpage` to achieve this. Because we put sections and subsections together, we don’t need a mark to distinguish them. Example 36 has essentially the same headers as this documentation.

---

[26] This is better than the left mark, bcause that doesn’t have a way to filter out empty ones which are generated by the `\part` commands in the document, and also `\leftmark` will produce the last title of the current page, which isn’t useful.

The macro `\checkposinpage` checks whether the [sub]section starts at or near the top of the page, and puts this information in the variable `\whichpage`, just like in example 35. The macro `\setmarks`, sets the marks, both the right mark for the title (with `\markright`), and the extra mark `whichpage`.

Example 36

```
\NewMarkClass{whichpage}
\newcommand\whichpage{}

\newcommand\checkposinpage{
  \ifthenelse{\lengthtest{\pagetotal<4\baselineskip}}%
    {\renewcommand{\whichpage}{top\thepage}}%
    {\renewcommand{\whichpage}{\thepage}}%
}

\AddToHook{cmd/section/before}{\checkposinpage}
\AddToHook{cmd/subsection/before}{\checkposinpage}

\newcommand\setmarks[1]{%
  \InsertMark{whichpage}{\whichpage}%
  \markright{#1}%
}

\renewcommand{\sectionmark}[1]{%
  \setmarks{\thesection\quad#1}%
}%

\renewcommand{\subsectionmark}[1]{%
  \setmarks{\thesubsection\quad#1}%
}%
```

Finally, the header code. For the last title of the/a previous page we use `\TopMark{2e-right-nonempty}`, for a title that is at/near the top of the page we use `\rightmark`. The algorithm is similar to example 35, but simplified:

```
if whichpage = \thepage (i.e., no [sub]section at or near top)
then
    use last [sub]section from a previous page
else (whichpage ≠ \thepage)
    if there is no whichpage mark on this page
       (this means the whichpage mark is not valid for this page)
    then use last [sub]section from a previous page
    else (there is a [sub]section title at/near the top of the page)
       use first [sub]section of current page
fi
```

Example 36 (Continued)

```
\fancypagestyle*{fancy}{%
  \fancyhf{}
  \fancyhead[L]{%
    \ifthenelse{\equal{\FirstMark{whichpage}}{\thepage}}%
      {\TopMark{2e-right-nonempty}}%
      % whichpage != page
      {\IfMarksEqualTF{whichpage}{top}{first}%
        {\TopMark{2e-right-nonempty}}% no mark on this page
        {\rightmark}% title at/near the top of the page
      }%
  }
  \fancyhead[R]{\textbf{\thepage}}
}
```

In the example 36 file there is some additional code for debugging that we don't show here.

In this documentation there are some cases that aren't covered by this code: the `\part` commands do not have a corresponding `\partmark`, and there are some hidden `\section*` commands that do not call `\sectionmark`. In these cases we manually add the `\checkposinpage` and `\setmarks` commands.

43 How to change shapes and traits of horizontal lines in headers/footers?

Sometimes one wants a decorative line in the header or footer that is a bit more sophisticated than a straight line[27].

If you just want to change the thickness, redefine `\headrulewidth`. For example:

```
\renewcommand{\headrulewidth}{0.1pt}
```

For more complicated forms you have to redefine `\headrule`. One example has already been given in section 20, and we will repeat it here:

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

This gives us the following headrule:

* * *

Here a simple `\headrule`, but with color, and a bit thicker.

```
\usepackage{xcolor}
. . .
\renewcommand\headrule{%
  \nointerlineskip
  \smash{\color{blue}\rule{\headwidth}{2.5pt}}%
}
```

---

[27] See https://tex.stackexchange.com/q/717266/113546

The `\nointerlineskip` is to prevent LaTeX to insert the normal vertical space between lines, and the `\smash` to let the `\headrule` not occupy any vertical space. Whether you want that is up to you, but if it occupies vertical space it may distort the page layout. This gives us the following headrule:

Now some dotted and dashed headrules:
With spaced dashes:

```
\newbox\dashbox\setbox\dashbox\hbox{-\,}
\renewcommand{\headrule}{%
  \smash{\makebox[\headwidth][c]{\xleaders\copy\dashbox\hfill-}}%
}
```

This gives:

With longer dashes:

```
\newbox\dashbox\setbox\dashbox\hbox{---\,}
\renewcommand{\headrule}{%
  \smash{\makebox[\headwidth][c]{\xleaders\copy\dashbox\hfill---}}%
}
```

This gives:

With spaced dots:

```
\newbox\dashbox\setbox\dashbox\hbox{.\,}
\renewcommand{\headrule}{%
  \smash{\makebox[\headwidth][c]{\xleaders\copy\dashbox\hfill.}}%
}
```

This gives:

With unspaced dots:

```
\newbox\dashbox\setbox\dashbox\hbox{.}
\renewcommand{\headrule}{%
  \smash{\makebox[\headwidth][c]{\xleaders\copy\dashbox\hfill.}}%
}
```

This gives:

Here is an example with a color gradient, using `tikz`:

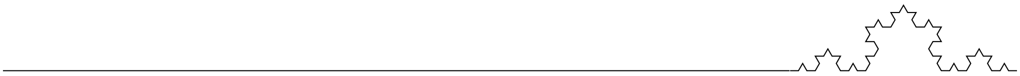
```
\usepackage{tikz}
\renewcommand{\headrule}{%
  \tikz \fill [left color=green,right color=yellow]
              (0,0) rectangle (\headwidth,2.5pt);
}
```

This gives:

Here is a particularly interesting one. It draws a Koch Snowflake at the end of the headrule[28].

```
\usepackage{tikz}
\usetikzlibrary{decorations.fractals}
. . .
\renewcommand\headrule{%
    \vspace{-0.75in}
    \hrulefill
    {\tikz[decoration=Koch snowflake]{%
          \draw decorate{ decorate{ decorate{ (0,-2) -- (3,-2) }}};}}
}
```

This gives:



Of course you would have to make sure not to place anything in the right part of the header.

---

[28] From Logan Weinert, https://tex.stackexchange.com/q/529474/113546

Part IV
Implementation

44 fancyhdr.sty

<*fancyhdr>

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

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

(*End of definition for* `\iff@nch@check`.)

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

```
\let\f@nch@gbl\relax
```

(*End of definition for* `\f@nch@gbl`.)

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

```
\newif\iff@nch@compatViii
\DeclareOption{compatV3}{%
  \PackageWarningNoLine{fancyhdr}{The ‘compatV3’ option is deprecated.\MessageBreak
    It will disappear in one of the following releases.\MessageBreak
    Please change your document to work\MessageBreak
    without this option}
  \let\f@nch@gbl\global
  \f@nch@compatViiitrue
}
```

(*End of definition for* `\iff@nch@compatViii`.)

`\iff@nch@twoside` Boolean for the `twoside` option. This is only set if the document itself is not two-sided.

```
\newif\iff@nch@twoside
\f@nch@twosidefalse
\DeclareOption{twoside}{%
  \if@twoside\else\f@nch@twosidetrue\fi
}
```

(*End of definition for* `\iff@nch@twoside`.)

`\f@nch@def` This macro defines another macro (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 not empty, a `\strut` will be added.

```
\newcommand\f@nch@def[2]{%
  \def\temp@a{#2}\ifx\temp@a\@empty\f@nch@gbl\def#1{}%
                 \else\f@nch@gbl\def#1{#2\strut}\fi}
```

(*End of definition for* `\f@nch@def`.)

Standard styles are redefined optionally. These definitions are borrowed from the nccfancyhdr package by by Alexander I. Rozhenko.

`\ps@myheadings` The redefinition of the `myheadings` style is conditional. We test the existence of the `\chapter` command and redefine the style accordingly.

```
\DeclareOption{myheadings}{%
  \@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
    }%
  }%
}
```

(*End of definition for* `\ps@myheadings`.)

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

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

An article in two-side mode:

```
      \def\ps@headings{\ps@f@nch@fancyproto \def\@mkboth{\protect\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 \def\@mkboth{\protect\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 \def\@mkboth{\protect\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 \def\@mkboth{\protect\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
  }%
}
```

(*End of definition for* `\ps@headings`.)

Process the options.

```
\ProcessOptions*
```

`\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`.

```
\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}
\long\def\f@nch@rc#1#2#3\f@nch@rc#4{\def#1{#2}#4\f@nchf@rc#1{#3}{#4}}
```

(*End of definition for* `\f@nch@forc`.)

`\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}}
```

(*End of definition for* `\f@nch@for`.)

`\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}
```

(*End of definition for* `\f@nch@default`.)

`\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}
```

(*End of definition for* `\f@nch@ifin`.)

`\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}}%
```

(*End of definition for* `\fancyhead`, `\fancyfoot`, *and* `\fancyhf`. *These functions are documented on page 4.*)

`\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}}%
```

(*End of definition for* `\fancyheadoffset`, `\fancyfootoffset`, *and* `\fancyhfoffset`. *These functions are documented on page 4.*)

`\f@nch@fancyhf@Echeck`

Macro for warning if ‘E’ is used without ‘twoside’ option.

```
\def\f@nch@fancyhf@Echeck#1{%
  \if@twoside\else
    \iff@nch@twoside\else
      \if\f@nch@@eo e%
        \PackageWarning{fancyhdr} {\string#1’s ‘E’ option without twoside option is useless.\
          Please consider using the ‘twoside’ option}%
  \fi\fi\fi
}
```

(*End of definition for* `\f@nch@fancyhf@Echeck`.)

`\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@`$\langle x\rangle\langle y\rangle\langle z\rangle$ with $\langle x\rangle$ from `[eo]`, $\langle y\rangle$ from `[lcr]` and $\langle z\rangle$ from `[hf]`.

```
\long\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 \PackageError{fancyhdr}{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
    \f@nch@fancyhf@Echeck{#1}%
    \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}}}}}}
```

(*End of definition for* `\f@nch@fancyhf`*.*)

`\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@offset@`$\langle x\rangle\langle y\rangle\langle z\rangle$ 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 \PackageError{fancyhdr}{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
    \f@nch@fancyhf@Echeck{#1}%
    \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@offset@\f@nch@eo\f@nch@lcr\f@nch@hf\endcsname {#4}}}}}%
  \f@nch@setoffs}
```

(*End of definition for* `\f@nch@fancyhfoffs`*.*)

`\fancyheadwidth`
`\fancyfootwidth`
`\fancyhfwidth`

The commands for field widths. Pick up the parameters, and supply an 'h' (`\fancyheadwidth`) or 'f' (`\fancyfootwidth`).

```
\NewDocumentCommand {\fancyheadwidth}{ s O{} O{} m }
                      {\f@nch@fancyhfwidth{#1}\fancyheadwidth h[#2][#3]{#4}}%
\NewDocumentCommand {\fancyfootwidth}{ s O{} O{} m }
                      {\f@nch@fancyhfwidth{#1}\fancyfootwidth f[#2][#3]{#4}}%
\NewDocumentCommand {\fancyhfwidth}  { s O{} O{} m }
                      {\f@nch@fancyhfwidth{#1}\fancyhfwidth  {}[#2][#3]{#4}}%
```

(*End of definition for* `\fancyheadwidth`*,* `\fancyfootwidth`*, and* `\fancyhfwidth`*. These functions are documented on page 4.*)

`\f@nch@fancyhfwidth`

This macro interprets the parameters for the header and footer field widths.
Parameters:
(1) The optional `*` argument.
(2) The user command that was used (like `\fancyheadwidth`). This is used for errors/warnings.
(3) `h` (for `\fancyheadwidth`), `f` (for `\fancyfootwidth`), or `{}` (for `\fancyhfwidth`).
(4-5) The two optional parameters that were given to these commands (default for each `[]`).
(6) The required parameter that was given to these commands.
The header and footer field widths are stored in command sequences with names of the form: `\f@nch@width@`⟨*x*⟩⟨*y*⟩⟨*z*⟩ with ⟨*x*⟩ from `[eo]`, ⟨*y*⟩ from `[lcr]` and ⟨*z*⟩ from `[hf]`. The header and footer alignments are stored (after defaults have been applied) in command sequences with names of the form: `\f@nch@align@`⟨*x*⟩⟨*y*⟩⟨*z*⟩ with ⟨*x*⟩ from `[eo]`, ⟨*y*⟩ from `[lcr]` and ⟨*z*⟩ from `[hf]`.

First we assign the ⟨*width*⟩ argument to a temporary length variable, to check if it is a legal ⟨*length*⟩.

```
\def\f@nch@fancyhfwidth#1#2#3[#4][#5]#6{%
  \setlength\@tempdima{#6}%
  \def\temp@c{}%
  \f@nch@forc\tmpf@ra{#4}%
  {\expandafter\f@nch@ifin\tmpf@ra{eolcrhf,EOLCRHF}%
    {}{\edef\temp@c{\temp@c\tmpf@ra}}}%
  \ifx\@empty\temp@c\else \PackageError{fancyhdr}{Illegal char ‘\temp@c’ in
    \string#2 argument: [#4]}{}%
  \fi
  \f@nch@for\temp@c{#4}%
  {\f@nch@default\f@nch@@eo{eo}\temp@c
    \f@nch@fancyhf@Echeck{#2}%
    \f@nch@default\f@nch@@lcr{lcr}\temp@c
    \f@nch@default\f@nch@@hf{hf}{#3\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
```

Then we store the ⟨*width*⟩ in variables for all the specified places. If the `*` form was given we use the calculated value, otherwise the bare argument.

```
            {%
              \IfBooleanTF{#1}{%
                \expandafter\edef\csname
                  f@nch@width@\f@nch@eo\f@nch@lcr\f@nch@hf\endcsname{\the\@tempdima}%
              }%
              {%
                \expandafter\def\csname
                  f@nch@width@\f@nch@eo\f@nch@lcr\f@nch@hf\endcsname{#6}%
              }%
```

Then we apply the defaults to the ⟨*alignment*⟩ argument. The defaults are:

- for the vertical alignment (v): `b` in a header, and `t` in a footer. This is done by executing a corresponding macro.
- for the horizontal alignment (h): l, c, r: the same as the field. Just copy the field letter.

If the optional parameter is empty, we leave it at these defaults, otherwise we process the arguments with `\f@nchdrwdt@align`. Finally we store the result in variables for the specified places.

```
              \csname f@nchdrwdt@align@v@\f@nch@hf\endcsname
              \edef\f@nch@align@@h{\f@nch@lcr}%
              \def\temp@a{#5}%
              \ifx\temp@a\@empty \else \f@nchdrwdt@align#5\@nil{#2}\fi
              \expandafter\edef\csname
                f@nch@align@\f@nch@eo\f@nch@lcr\f@nch@hf\endcsname
                   {\f@nch@align@@v\f@nch@align@@h}}}}}
```

(*End of definition for* `\f@nch@fancyhfwidth`*.*)

`\f@nch@width@elh`
`\f@nch@width@ech`
`\f@nch@width@erh`
`\f@nch@width@olh`
`\f@nch@width@och`
`\f@nch@width@orh`
`\f@nch@width@elf`
`\f@nch@width@ecf`
`\f@nch@width@erf`
`\f@nch@width@olf`
`\f@nch@width@ocf`
`\f@nch@width@orf`

Length parameters for the widths. These are stored as macros. They are calculated as lengths when the header/footer is built.

```
\def\f@nch@width@elh{\headwidth}
\def\f@nch@width@ech{\headwidth}
\def\f@nch@width@erh{\headwidth}
\def\f@nch@width@olh{\headwidth}
\def\f@nch@width@och{\headwidth}
\def\f@nch@width@orh{\headwidth}
\def\f@nch@width@elf{\headwidth}
\def\f@nch@width@ecf{\headwidth}
\def\f@nch@width@erf{\headwidth}
\def\f@nch@width@olf{\headwidth}
\def\f@nch@width@ocf{\headwidth}
\def\f@nch@width@orf{\headwidth}
```

(*End of definition for* `\f@nch@width@elh` *and others.*)

`\f@nch@align@elh`
`\f@nch@align@ech`
`\f@nch@align@erh`
`\f@nch@align@olh`
`\f@nch@align@och`
`\f@nch@align@orh`
`\f@nch@align@elf`
`\f@nch@align@ecf`
`\f@nch@align@erf`
`\f@nch@align@olf`
`\f@nch@align@ocf`
`\f@nch@align@orf`

Alignment parameters for the widths.

```
\def\f@nch@align@elh{bl}
\def\f@nch@align@ech{bc}
\def\f@nch@align@erh{br}
\def\f@nch@align@olh{bl}
\def\f@nch@align@och{bc}
\def\f@nch@align@orh{br}
\def\f@nch@align@elf{tl}
\def\f@nch@align@ecf{tc}
\def\f@nch@align@erf{tr}
\def\f@nch@align@olf{tl}
\def\f@nch@align@ocf{tc}
\def\f@nch@align@orf{tr}
```

(*End of definition for* `\f@nch@align@elh` *and others.*)

\f@nchdrwdt@align@v@h
\f@nchdrwdt@align@v@f
\f@nchdrwdt@align

`\f@nchdrwdt@align@v@h`: set v to `b`

`\f@nchdrwdt@align@v@f`: set v to `t`

`\f@nchdrwdt@align{⟨vert⟩}{⟨hor⟩}\@nil{⟨originating command⟩}`

The internal processing for ⟨`alignment`⟩ parameter in `\fancyhfwidth`, etc.

```
ALGORITHM \f@nchdrwdt@align:
(v = vertical position; h = horizontal position)
These have been set to their defaults before calling us.
IF #1 in {T,t,c,b,B,-}
THEN if #1 ≠ '-' then v := #1 fi
    if #2 is not empty then h := #2 fi
ELSE (#1 not in {T,t,c,b,B,-} – it must be a horizontal alignment)
    h := #1
FI
if h not in {l,c,r,j} then ERROR fi
```

The result can be found in the variables `\f@nch@align@@v` and `\f@nch@align@@h`.

```
\def\f@nchdrwdt@align@v@h{\def\f@nch@align@@v{b}}%
\def\f@nchdrwdt@align@v@f{\def\f@nch@align@@v{t}}%
\long\def\f@nchdrwdt@align#1#2\@nil#3{%
  \f@nch@ifin{#1}{TtcbB-}{%
    \f@nch@ifin{#1}{-}{}{\def\f@nch@align@@v{#1}}%
    \def\@tempa{#2}%
    \ifx\@tempa\@empty \else \def\f@nch@align@@h{#2}\fi
  }%
  {\def\f@nch@align@@h{#1}}%
  \expandafter\f@nch@ifin\expandafter{\f@nch@align@@h}{lcrj}{}%
    {\PackageError{fancyhdr}
                  {\string#3: Illegal char `\f@nch@align@@h'\MessageBreak
                              in alignment argument}{}}%
}
```

(*End of definition for* `\f@nchdrwdt@align@v@h`*,* `\f@nchdrwdt@align@v@f`*, and* `\f@nchdrwdt@align`*.*)

\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}}
```

(*End of definition for* `\lhead` *and others. These functions are documented on page 71.*)

`\f@nch@headwidth`

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

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

(*End of definition for* `\f@nch@headwidth`*.*)

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

Length parameters for the offsets.

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

(*End of definition for* `\f@nch@offset@elh` *and others.*)

`\headrulewidth`
`\footrulewidth`

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

(*End of definition for* `\headrulewidth` *and* `\footrulewidth`*. These functions are documented on page 4.*)

`\headruleskip`

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

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

(*End of definition for* `\headruleskip`*. This function is documented on page 4.*)

`\footruleskip`

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

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

(*End of definition for* `\footruleskip`*. This function is documented on page 4.*)

`\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}
```

(*End of definition for* `\plainheadrulewidth` *and* `\plainfootrulewidth`*. These functions are documented on page 71.*)

`\if@fancyplain`

Boolean for the implementation of `\fancyplain`

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

(*End of definition for* `\if@fancyplain`*.*)

`\fancyplain`

Deprecated macro

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

(*End of definition for* `\fancyplain`*. This function is documented on page 71.*)

`\headwidth`

Initialise `\headwidth` with a magic constant.

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

(*End of definition for* `\headwidth`*. This function is documented on page 4.*)

`\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
\ifdefined\ExplSyntaxOn
  \ExplSyntaxOn
  \providecommand\IfFormatAtLeastTF{\@ifl@t@r\fmtversion}
  \IfFormatAtLeastTF{2021-06-01}{
```

We disable paragraph hooks, so that no paragraph hooks will intrude in fancyhdr code. **NOTE: This is a hack, and should be replaced by cleaner code as soon as the LaTeX kernel provides the necessary commands.** The way we do this now: Every hook consists of 4 global 'variables':

`\f@nch@saveclr@parhook`
`\f@nch@restore@parhook`

- `\__hook` ⟨*name*⟩
- `\__hook_toplevel` ⟨*name*⟩
- `\__hook_next` ⟨*name*⟩
- `\g__hook_`⟨*name*⟩`_code_prop`

and there are 4 hooks (`para/before`, `para/begin`, `para/end`, and `para/after`). At the beginning of a header/footer, i.e., before any init code and hooks, all these variables are locally saved into variables with the same name, prefixed with '`f@nch@`', and then clear the hook. At the end of the header/footer they are globally restored to the saved value. So we do only global assignments to them to avoid problems. Save a (paragraph) hook locally and and clear it globally. Restore it globally at the end of header/footer processing.

```
    \def\f@nch@saveclr@parhook #1{
      \expandafter\let\csname f@nch@__hook~#1\expandafter\endcsname
                      \csname __hook~#1\endcsname
      \expandafter\let\csname f@nch@__hook_toplevel~#1\expandafter\endcsname
                      \csname __hook_toplevel~#1\endcsname
      \expandafter\let\csname f@nch@__hook_next~#1\expandafter\endcsname
                      \csname __hook_next~#1\endcsname
      \expandafter\let\csname f@nch@g__hook_#1_code_prop\expandafter\endcsname
                      \csname g__hook_#1_code_prop\endcsname
      \RemoveFromHook{#1}[*]
      \ClearHookNext{#1}
    }
    \def\f@nch@restore@parhook #1{
      \global\expandafter\let\csname __hook~#1\expandafter\endcsname
                             \csname f@nch@__hook~#1\endcsname
      \global\expandafter\let\csname __hook_toplevel~#1\expandafter\endcsname
                             \csname f@nch@__hook_toplevel~#1\endcsname
      \global\expandafter\let\csname __hook_next~#1\expandafter\endcsname
                             \csname f@nch@__hook_next~#1\endcsname
      \global\expandafter\let\csname g__hook_#1_code_prop\expandafter\endcsname
                             \csname f@nch@g__hook_#1_code_prop\endcsname
    }
    \def\f@nch@resetpar{
      \f@nch@everypar{}
      \f@nch@saveclr@parhook{para/before}
      \f@nch@saveclr@parhook{para/begin}
      \f@nch@saveclr@parhook{para/end}
      \f@nch@saveclr@parhook{para/after}
    }
    \def\f@nch@restorepar{
      \f@nch@restore@parhook{para/before}
      \f@nch@restore@parhook{para/begin}
      \f@nch@restore@parhook{para/end}
      \f@nch@restore@parhook{para/after}
    }
  }{
    \def\f@nch@resetpar{
      \f@nch@everypar{}
    }
    \def\f@nch@restorepar{}
  }
  \ExplSyntaxOff
\else
  \def\f@nch@resetpar{%
    \f@nch@everypar{}%
  }
  \def\f@nch@restorepar{}
\fi
```

(*End of definition for* `\f@nch@raggedleft` *and others.*)

`\f@nch@noUppercase`

We want `\nouppercase` to work with the various evolutionary stages of `\MakeUppercase`. The current version (2022/11/09) accepts an optional argument with a language specification. Therefore we define a dummy macro `\f@nch@noUppercase` which copies its mandatory argument, as a replacement for `\MakeUppercase` while `\nouppercase` is active.

```
\newcommand\f@nch@noUppercase[2][]{#2}
```

`\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). Also reset the catcodes that are changed in verbatim environments, `\makeatother` and `\ExplSyntaxOn`. 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`. Actually this is done in the LaTeX output routine, so we don't have to do it here.

```
\def\f@nch@reset{\f@nch@resetpar\restorecr\endlinechar=13
  \catcode`\\=0\catcode`\{=1\catcode`\}=2\catcode`\$=3\catcode`\&=4
  \catcode`\#=6\catcode`\^=7\catcode`\_=8\catcode`\ =10\catcode`\@=11
  \catcode`\:=11\catcode`\~=13\catcode`\%=14
  \catcode0=15 %NULL
  \catcode9=10 %TAB
  \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\f@nch@noUppercase
      \expandafter\let\csname MakeUppercase \endcsname\relax
      \expandafter\def\csname MakeUppercase\space\space\space\endcsname
                                                   [####1]####2{####2}%
      ##1}}%
  \@ifundefined{@normalsize} {\normalsize} % for ucthesis.cls
   {\@normalsize}%
  }
```

(*End of definition for* `\f@nch@noUppercase` *and* `\f@nch@reset`*.*)

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

```
\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-field`*⟩ is empty[29]:

```
  \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-field`*⟩. We will calculate them in the `\@tempskipa` and `\@tempskipb` registers. 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-field`*⟩ is empty:

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

---

[29] This code is reused from the nccfancyhdr package by Alexander I. Rozhenko

Do the same things with the `\@tempskipb` register if the ⟨*right-field*⟩ 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-field*⟩ and ⟨*right-field*⟩. 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
}
```

(*End of definition for* `\fancycenter`. *This function is documented on page 5.*)

`\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.

Storage for the header initialisation code.

`\f@nch@headinit`

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

(*End of definition for* `\fancyheadinit` *and* `\f@nch@headinit`. *These functions are documented on page 4.*)

`\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.

Storage for the footer initialisation code.

`\f@nch@footinit`

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

(*End of definition for* `\fancyfootinit` *and* `\f@nch@footinit`. *These functions are documented on page 4.*)

`\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}%
}
```

(*End of definition for* `\fancyhfinit`*. This function is documented on page 4.*)

`fancyhdr/before` (*hook*)
`fancyhdr/after` (*hook*)
`fancyhdr/head/begin` (*hook*)
`fancyhdr/head/end` (*hook*)
`fancyhdr/foot/begin` (*hook*)
`fancyhdr/foot/end` (*hook*)

Here we define the fancyhdr hooks. It will be conditional on the presence of hook support in the LaTeX kernel.

```
\ifdefined\NewMirroredHookPair
  \NewMirroredHookPair{fancyhdr/before}{fancyhdr/after}
  \NewMirroredHookPair{fancyhdr/head/begin}{fancyhdr/head/end}
  \NewMirroredHookPair{fancyhdr/foot/begin}{fancyhdr/foot/end}
\fi
```

`\f@nch@height`

Length variable to store height of header/footer for use in `\fancyhdrsettoheight`

```
\newlength\f@nch@height
```

(*End of definition for* `\f@nch@height`*.*)

`\f@nch@footalignment`

Length variable to store alignment length of `\fancyfootalign`

```
\newlength\f@nch@footalignment
```

(*End of definition for* `\f@nch@footalignment`*.*)

`\iff@nch@footalign`

Boolean variable to store if a ⟨`length`⟩ parameter was given to `\fancyfootalign`

```
\newif\iff@nch@footalign\f@nch@footalignfalse
```

(*End of definition for* `\iff@nch@footalign`*.*)

`\fancyfootalign`

This macro sets the distance between the bottom of the footer and the bottom margin. The argument can be empty, or a ⟨`length`⟩.

```
\newcommand{\fancyfootalign}[1]{%
  \def\temp@a{#1}%
  \ifx\temp@a\@empty
    \f@nch@footalignfalse
  \else
    \f@nch@footaligntrue
    \setlength\f@nch@footalignment{#1}%
  \fi
}
```

(*End of definition for* `\fancyfootalign`*. This function is documented on page 4.*)

`\fancyhdrsettoheight`

Macro to store the height of a header/footer in a length variable.
`\fancyhdrsettoheight{`⟨*lengthvar*⟩`}{`⟨*header/footer*⟩`}`
The second parameter can be `oddhead`, `evenhead`, `oddfoot`, or `evenfoot`.

```
\newcommand\fancyhdrsettoheight[2]{%
  \expandafter\ifx\csname f@nch@#2\endcsname\fancyhdrsettoheight
    \else\PackageError{fancyhdr}{Unknown parameter #2 in \string\fancyhdrsettoheight}{}\fi
  \setbox\@tempboxa\hbox{{\f@nch@checkfalse\csname @#2\endcsname}}%
  \setlength{#1}\f@nch@height
  \setbox\@tempboxa\box\voidb@x
}
```

Define commands that specify the valid arguments for the second parameter.

```
\let\f@nch@oddhead\fancyhdrsettoheight
\let\f@nch@evenhead\fancyhdrsettoheight
\let\f@nch@oddfoot\fancyhdrsettoheight
\let\f@nch@evenfoot\fancyhdrsettoheight
```

(*End of definition for* `\fancyhdrsettoheight`*. This function is documented on page 5.*)

`\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}%
  \global\f@nch@height=\ht0
  \ifdim\ht0>#1\relax
```

This is the part where the the header/footer is too tall for the vertical space. If the `[nocheck]` package option is not given, we give a warning message.

```
    \iff@nch@check
      \dimen0=#1\advance\dimen0-\ht0
      \PackageWarning{fancyhdr}{%
        \string#1 is too small (\the#1): \MessageBreak
        Make it at least \the\ht0, for example:\MessageBreak
        \string\setlength{\string#1}{\the\ht0}%
```

If the `[compatV3]` option was given (and not `[nocheck]`), we will also change the `\headheight`/`\footskip` globally below, and announce this in the warning message.

```
        \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:\MessageBreak
          \string\addtolength{\string\topmargin}{\the\dimen0}%
        \fi
        \@gobble
      }%
```

Here we do the actual global changing of the `\headheight`/`\footskip`.

```
      \iff@nch@compatViii
        \dimen0=#1\relax
        \global#1=\ht0\relax
        \ht0=\dimen0 %
      \else
        \ht0=#1\relax
      \fi
```

However, if the `[nocheck]` options is given, we just make the height of the header/footer equal to the reserved space, so that no warning about “Overfull vbox” will be given. So we pretend that it fits, and it is the user’s responsibility to make sure no unwanted effects take place.

```
    \else
      \ht0=#1\relax
    \fi
  \fi
  \box0}
```

(*End of definition for* `\f@nch@vbox`.)

`\f@nch@head` Put together a header (`\f@nch@head`) or footer (`\f@nch@foot`) given the left, center and right text and their widths, 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 8 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).
6. Even (`e`) or odd (`o`).

Before constructing the header or footer, the environment is reset to a known state, the appropriate hooks (`fancyhdr/before` and `fancyhdr/head/begin` or `fancyhdr/foot/begin`) are run, and then the corresponding initialisation code as given in `\fancyheadinit` or `\fancyfootinit`, respectively, is run.
After constructing the header or footer, the hooks for the end (`fancyhdr/head/end` or `fancyhdr/foot/end` and `fancyhdr/after`) are run.

```
\newcommand\f@nch@head[6]{%
  \f@nch@reset
  \ifdefined\UseHook\UseHook{fancyhdr/before}\UseHook{fancyhdr/head/begin}\fi
  \f@nch@headinit\relax
  #1%
  \hbox to\headwidth{%
    \f@nch@vbox\headheight{%
      \f@nch@hfbox{#2}{#3}{#4}{#6}{h}%
      \vskip\headruleskip\relax
      \headrule
    }%
  }%
  #5%
  \ifdefined\UseHook\UseHook{fancyhdr/head/end}\UseHook{fancyhdr/after}\fi
  \f@nch@restorepar
}
```

`\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[6]{%
  \f@nch@reset
  \ifdefined\UseHook\UseHook{fancyhdr/before}\UseHook{fancyhdr/foot/begin}\fi
  \f@nch@footinit\relax
  #1%
  \hbox to\headwidth{%
    \f@nch@vbox\footskip{%
      \setbox0=\vbox{\footrule}\unvbox0
      \vskip\footruleskip
      \f@nch@hfbox{#2}{#3}{#4}{#6}{f}%
```

Add vertical space if `\fancyfootalign{`⟨*length*⟩`}` has been given.

```
      \iff@nch@footalign \vskip\f@nch@footalignment \fi
    }%
  }%
  #5%
  \ifdefined\UseHook\UseHook{fancyhdr/foot/end}\UseHook{fancyhdr/after}\fi
  \f@nch@restorepar
}
```

(*End of definition for* `\f@nch@head` *and* `\f@nch@foot`*.*)

`\f@nch@widthL`
`\f@nch@widthC`
`\f@nch@widthR`

Length variables to store the field widths during construction of the header/footer.

```
\newlength\f@nch@widthL
\newlength\f@nch@widthC
\newlength\f@nch@widthR
```

(*End of definition for* `\f@nch@widthL`*,* `\f@nch@widthC`*, and* `\f@nch@widthR`*.*)

`\f@nch@hfbox`

This macro constructs the box with the header or footer. It has 5 parameters:

1. Left field
2. Center field
3. Right field
4. Even (`e`) or odd (`o`).
5. Header (`h`) or footer (`f`).

Algorithm:

First we store the field widths in length variables.

If the sum of the field widths > `\headwidth`: the center field is centered in the header/footer, and the left and right fields are put in an `\`⟨*x*⟩`lap` to prevent error messages about overlapping.

Otherwise, if there is no overlap between the fields, also put the center field centered in the header/footer. This is done by the macro `\f@nch@hfbox@center`

Otherwise (there is enough space, but centering would cause overlap):

put the center field centered between the left and right field, i.e., with equal gaps on both sides. This is done by the macro `\f@nch@hfbox@fit`.

```
\newcommand\f@nch@hfbox[5]{%
  \setlength\f@nch@widthL{\csname f@nch@width@#4l#5\endcsname}%
  \setlength\f@nch@widthC{\csname f@nch@width@#4c#5\endcsname}%
  \setlength\f@nch@widthR{\csname f@nch@width@#4r#5\endcsname}%
  \let\@tempa\f@nch@hfbox@center
  \ifdim \dimexpr \f@nch@widthL+\f@nch@widthC+\f@nch@widthR>\headwidth
  \else
    \ifdim \dimexpr \f@nch@widthL+0.5\f@nch@widthC>0.5\headwidth
      \let \@tempa\f@nch@hfbox@fit
    \fi
    \ifdim \dimexpr \f@nch@widthR+0.5\f@nch@widthC>0.5\headwidth
      \let \@tempa\f@nch@hfbox@fit
    \fi
  \fi
  \@tempa{#1}{#2}{#3}#4#5%
}
```

(*End of definition for* `\f@nch@hfbox`*.*)

`\f@nch@hfbox@center`

This macro constructs the box with the header or footer. This is the version that centers the center field in the total header/footer. It has 4 parameters:

1. Left field
2. Center field
3. Right field
4. Even (`e`) or odd (`o`).
5. Header (`h`) or footer (`f`).

```
\newcommand\f@nch@hfbox@center[5]{%
  \hbox to \headwidth{%
    \rlap{\f@nch@parbox{#1}\f@nch@widthL{#4}l{#5}}%
    \hfill
    \f@nch@parbox{#2}\f@nch@widthC{#4}c{#5}%
    \hfill
    \llap{\f@nch@parbox{#3}\f@nch@widthR{#4}r{#5}}%
  }%
}
```

(*End of definition for* `\f@nch@hfbox@center`.)

`\f@nch@hfbox@fit` This macro constructs the box with the header or footer. This is the version that centers the center field between the left and right fields. It has 4 parameters:

1. Left field
2. Center field
3. Right field
4. Even (`e`) or odd (`o`).
5. Header (`h`) or footer (`f`).

```
\newcommand\f@nch@hfbox@fit[5]{%
  \hbox to \headwidth{%
    \f@nch@parbox{#1}\f@nch@widthL{#4}l{#5}%
    \hfill
    \f@nch@parbox{#2}\f@nch@widthC{#4}c{#5}%
    \hfill
    \f@nch@parbox{#3}\f@nch@widthR{#4}r{#5}%
  }%
}%
```

(*End of definition for* `\f@nch@hfbox@fit`.)

`\f@nch@parbox` This macro constructs one `\parbox` in the header or footer. It has 4 parameters:

1. The contents
2. The width for the `\parbox`.
3. Even (`e`) or odd (`o`).
4. Left (`l`), center (`c`) or right (`r`).
5. Header (`h`) or footer (`f`).

Result: the proper `\parbox`

First we get (with `\csname`) the proper alignment parameter. Then we expand this so that `\f@nch@parbox@align` gets the two alignment letters as separate arguments. Then `\f@nch@parbox@align` is called to set up the alignment variables. After that we construct the `\parbox` with the calculated variables.

```
\newcommand\f@nch@parbox[5]{%
  \expandafter\expandafter\expandafter\f@nch@parbox@align
                    \csname f@nch@align@#3#4#5\endcsname
  \parbox[\f@nch@align@@v]{#2}%
    {%
      \f@nch@align@@pre
      \f@nch@align@@h\leavevmode\ignorespaces#1%
      \f@nch@align@@post
    }%
}
```

(*End of definition for* `\f@nch@parbox`.)

`\f@nch@parbox@align` The macro `\f@nch@parbox@align` sets the alignment variables for `\f@nch@parbox`. It has 2 parameters, the two letters for the vertical and horizontal alignment, with the defaults applied.

1. Vertical (`T, t, c, b, B`).
2. Horizontal (`l, c, r, j`).

Result variables:

`\f@nch@align@@v` The vertical alignment for the `\parbox`: `t`, `c` or `b`.

`\f@nch@align@@h` The horizontal alignment for the `\parbox`: `\raggedright`, `\centering`, `\raggedleft` or empty for `l, c, r, j`, respectively.

`\f@nch@align@@pre` code before the contents of the `\parbox`: `\vspace{0pt}` for `T` alignment, otherwise empty.

`\f@nch@align@@post` code after the contents of the `\parbox`: `\vspace{0pt}` for `B` alignment, otherwise empty.

First we set the defaults for `\f@nch@align@@pre` and `\f@nch@align@@post` (empty). Then we call the proper macros for the alignment parameters.

```
\newcommand\f@nch@parbox@align[2]{%
  \def\f@nch@align@@pre{}%
  \def\f@nch@align@@post{}%
  \csname f@nch@parbox@align@v#1\endcsname
  \csname f@nch@parbox@align@h#2\endcsname
}
```

(*End of definition for* `\f@nch@parbox@align`.)

`\f@nch@parbox@align@vT`
`\f@nch@parbox@align@vt`
`\f@nch@parbox@align@vc`
`\f@nch@parbox@align@vb`
`\f@nch@parbox@align@vB`
`\f@nch@parbox@align@hl`
`\f@nch@parbox@align@hc`
`\f@nch@parbox@align@hr`
`\f@nch@parbox@align@hj`

The macro `\f@nch@parbox@align@`⟨*v*|*h*⟩⟨*x*⟩ sets the variables for the vertical or horizontal alignment option ⟨*x*⟩.

```
\def\f@nch@parbox@align@vT{\def\f@nch@align@@v{t}\def\f@nch@align@@pre{\vspace{0pt}}}
\def\f@nch@parbox@align@vt{\def\f@nch@align@@v{t}}
\def\f@nch@parbox@align@vc{\def\f@nch@align@@v{c}}
\def\f@nch@parbox@align@vb{\def\f@nch@align@@v{b}}
\def\f@nch@parbox@align@vB{\def\f@nch@align@@v{b}\def\f@nch@align@@post{\vspace{0pt}}}
\def\f@nch@parbox@align@hl{\def\f@nch@align@@h{\raggedright}}
\def\f@nch@parbox@align@hc{\def\f@nch@align@@h{\centering}}
\def\f@nch@parbox@align@hr{\def\f@nch@align@@h{\raggedleft}}
\def\f@nch@parbox@align@hj{\def\f@nch@align@@h{}}
```

(*End of definition for* `\f@nch@parbox@align@vT` *and others.*)

`\@chapapp`

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

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

(*End of definition for* `\@chapapp`.)

`\f@nch@initialise`

This macro initialises the headers and footers and `\chaptermark` and/or `\[sub]sectionmark` for page style `fancy`

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

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

`\chaptermark`
`\sectionmark`
`\subsectionmark`

```
  \@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: left head empty on “plain” pages, `\rightmark` on even, `\leftmark` on odd pages; right head 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
```

(*End of definition for* `\f@nch@initialise` *and others.*)

\ps@f@nch@fancyproto

`\ps@f@nch@fancyproto` is the initial value for page style `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 page style.

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

(*End of definition for* `\ps@f@nch@fancyproto`.)

`\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}{}
```

(*End of definition for* `\ps@fancy`.)

`\ps@fancyplain`

The page style `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}
```

(*End of definition for* `\ps@fancyplain`.)

`\f@nch@ps@empty`

Save the definition of `\ps@empty` (page style `empty`).

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

(*End of definition for* `\f@nch@ps@empty`.)

`\ps@f@nch@fancycore`

The actual implementation of page style `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\f@nch@oddhead{\f@nch@head\f@nch@Oolh\f@nch@olh\f@nch@och\f@nch@orh\f@nch@Oorh{o}}%
  \def\@oddhead{%
    \iff@nch@twoside
      \ifodd\c@page
        \f@nch@oddhead
      \else
        \@evenhead
      \fi
    \else
      \f@nch@oddhead
    \fi
  }
  \def\f@nch@oddfoot{\f@nch@foot\f@nch@Oolf\f@nch@olf\f@nch@ocf\f@nch@orf\f@nch@Oorf{o}}%
  \def\@oddfoot{%
    \iff@nch@twoside
      \ifodd\c@page
        \f@nch@oddfoot
      \else
        \@evenfoot
      \fi
    \else
      \f@nch@oddfoot
    \fi
  }
  \def\@evenhead{\f@nch@head\f@nch@Oelh\f@nch@elh\f@nch@ech\f@nch@erh\f@nch@Oerh{e}}%
  \def\@evenfoot{\f@nch@foot\f@nch@Oelf\f@nch@elf\f@nch@ecf\f@nch@erf\f@nch@Oerf{e}}%
}
```

(*End of definition for* `\ps@f@nch@fancycore`*.*)

`\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
```

(*End of definition for* `\f@nch@Oolh` *and others.*)

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

(*End of definition for* `\f@nch@offsolh` *and* `\f@nch@offselh`*.*)

`\f@nch@offsolf`
`\f@nch@offself`

The same for the footer.

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

(*End of definition for* `\f@nch@offsolf` *and* `\f@nch@offself`*.*)

`\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 page style `fancy`) in `compatV3` mode) or locally (for `\fancypagestyle`). The macros `\f@nch@Oxyz` tell what should be done at the various ends of the headers/footers. They are done with `\def` rather than `\let` so they are easier to pick up 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\def\f@nch@Oolh{\f@nch@offsolh}%
  \f@nch@gbl\def\f@nch@Oelh{\f@nch@offselh}%
  \f@nch@gbl\def\f@nch@Oorh{\hss}%
  \f@nch@gbl\def\f@nch@Oerh{\hss}%
  \f@nch@gbl\def\f@nch@Oolf{\f@nch@offsolf}%
  \f@nch@gbl\def\f@nch@Oelf{\f@nch@offself}%
  \f@nch@gbl\def\f@nch@Oorf{\hss}%
  \f@nch@gbl\def\f@nch@Oerf{\hss}%
}
```

(*End of definition for* `\f@nch@setoffs`.)

`\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\f@nch@topfloat\@toplist\let\f@nch@botfloat\@botlist\latex@makecol}%
}
```

(*End of definition for* `\iff@nch@footnote` *and* `\@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\f@nch@topfloat\@empty #2\else #1\fi}%
\newcommand\ifbotfloat[2]{\ifx\f@nch@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}%
```

(*End of definition for* `\iftopfloat` *and others. These functions are documented on page 5.*)

`\@temptokenb`

A token register to collect information for `\fancypagestyle*`. The definition is conditional on the non-existence of it.

```
\ifx\@temptokenb\undefined \csname newtoks\endcsname\@temptokenb\fi
```

(*End of definition for* `\@temptokenb`.)

`\iff@nch@pagestyle@star`

A conditional to record if `\fancypagestyle*` is used.

```
\newif\iff@nch@pagestyle@star
```

(*End of definition for* `\iff@nch@pagestyle@star`.)

`\fancypagestyle`

Define a new page style. With `*` define a “closed” page style, otherwise an “open” one.

```
\newcommand\fancypagestyle{%
  \@ifstar{\f@nch@pagestyle@startrue\f@nch@pagestyle}%
          {\f@nch@pagestyle@starfalse\f@nch@pagestyle}%
}
```

(*End of definition for* `\fancypagestyle`. *This function is documented on page 5.*)

`\f@nch@pagestyle`

Internal macro for `\fancypagestyle`. The optional second argument is the base page style. It defaults to `\ps@f@nch@fancyproto`.

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

(*End of definition for* `\f@nch@pagestyle`.)

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

```
\long\def\f@nch@@pagestyle#1[#2]#3{%
  \@ifundefined{ps@#2}{%
    \PackageError{fancyhdr}{\string\fancypagestyle: Unknown base page style ‘#2’}{}%
  }{%
    \@ifundefined{f@nch@ps@#2-is-fancyhdr}{%
      \PackageError{fancyhdr}{\string\fancypagestyle: Base page style ‘#2’ is not fancyhdr-ba
    }%
    {%
```

First put necessary definitions in `\@temptokenb`, if required (`\fancypagestyle*`) by calling `\f@nch@pagestyle@setup`. Then define the page style by expanding `\the\@temptokenb` and adding the base style and our definitions.

```
      \f@nch@pagestyle@setup
      \def\temp@b{\@namedef{ps@#1}}%
      \expandafter\temp@b\expandafter{\the\@temptokenb
          \let\f@nch@gbl\relax\@nameuse{ps@#2}#3\relax}%
      \@namedef{f@nch@ps@#1-is-fancyhdr}{}%
    }%
  }%
}
```

(*End of definition for* `\f@nch@@pagestyle`.)

`\f@nch@pagestyle@setup` Internal macro for `\fancypagestyle`. Setup `\@temptokenb`:
For `\fancypagestyle*` collect relevant macro definitions in `\@temptokenb`.
For `\fancypagestyle` make `\@temptokenb` empty.

```
\newcommand\f@nch@pagestyle@setup{%
  \iff@nch@pagestyle@star
```

For `\fancypagestyle*`, first save value of `\iff@nch@check` (the `[nocheck]` option).

```
    \iff@nch@check\@temptokenb={\f@nch@checktrue}\else\@temptokenb={\f@nch@checkfalse}\fi
```

Save values of all relevant macros (50 in total):
headers and footers (12), header and footer widths (12), header and footer alignments (12), header and footer offsets (8), header and footer inits (2), `\headrule` and `\footrule` and `...width` (4)

```
    \@tfor\temp@a:=
      \f@nch@olh\f@nch@och\f@nch@orh\f@nch@elh\f@nch@ech\f@nch@erh
      \f@nch@olf\f@nch@ocf\f@nch@orf\f@nch@elf\f@nch@ecf\f@nch@erf
      \f@nch@width@elh\f@nch@width@ech\f@nch@width@erh\f@nch@width@olh
      \f@nch@width@och\f@nch@width@orh\f@nch@width@elf\f@nch@width@ecf
      \f@nch@width@erf\f@nch@width@olf\f@nch@width@ocf\f@nch@width@orf
      \f@nch@align@elh\f@nch@align@ech\f@nch@align@erh\f@nch@align@olh
      \f@nch@align@och\f@nch@align@orh\f@nch@align@elf\f@nch@align@ecf
      \f@nch@align@erf\f@nch@align@olf\f@nch@align@ocf\f@nch@align@orf
      \f@nch@Oolh\f@nch@Oorh\f@nch@Oelh\f@nch@Oerh
      \f@nch@Oolf\f@nch@Oorf\f@nch@Oelf\f@nch@Oerf
      \f@nch@headinit\f@nch@footinit
      \headrule\headrulewidth\footrule\footrulewidth
    \do {%
```

First get the body of the macro. Next put it in a `\def`⟨`\macro`⟩`{`⟨*body of* `\macro`⟩`}`.

```
      \toks@=\expandafter\expandafter\expandafter{\temp@a}%
      \toks@=\expandafter\expandafter\expandafter{%
        \expandafter\expandafter\expandafter\def
        \expandafter\expandafter\temp@a\expandafter{\the\toks@}}%
```

Set up a macro to append `\toks@` to `\@temptokenb` and then execute it.

```
      \edef\temp@b{\@temptokenb={\the\@temptokenb\the\toks@}}%
      \temp@b
    }%
```

Now pick up the offset length variables in a similar way, but with `\setlength` rather than `\def`.

```
    \@tfor\temp@a:=
      \f@nch@offset@olh\f@nch@offset@orh\f@nch@offset@elh\f@nch@offset@erh
      \f@nch@offset@olf\f@nch@offset@orf\f@nch@offset@elf\f@nch@offset@erf
    \do {%
      \toks@=\expandafter\expandafter\expandafter{\expandafter\the\temp@a}%
      \toks@=\expandafter\expandafter\expandafter{%
        \expandafter\expandafter\expandafter\setlength
        \expandafter\expandafter\temp@a\expandafter{\the\toks@}}%
```

Set up a macro to append `\toks@` to `\@temptokenb` and then execute it.

```
      \edef\temp@b{\@temptokenb={\the\@temptokenb\the\toks@}}%
      \temp@b
    }%
  \else
```

For `\fancypagestyle` without `*`, set `\@temptokenb` empty.

```
    \@temptokenb={}%
  \fi
}
```

(*End of definition for* `\f@nch@pagestyle@setup`.)

`\fancypagestyleassign`

`\fancypagestyleassign{`⟨*ps1*⟩`}{`⟨*ps2*⟩`}`

Assigns page style ⟨*ps2*⟩ to ⟨*ps1*⟩. This causes ⟨*ps1*⟩ to be an exact copy of ⟨*ps2*⟩, but completely independent of ⟨*ps2*⟩. We do the equivalent of a `\let` command, like `\let\ps@`⟨*ps1*⟩`\ps@`⟨*ps2*⟩.

```
\newcommand\fancypagestyleassign[2]{%
  \@ifundefined{ps@#2}{%
    \PackageError{fancyhdr}{\string\fancypagestyleassign: Unknown page style ‘#2’}{}%
    }{%
     \expandafter\let
       \csname ps@#1\expandafter\endcsname
       \csname ps@#2\endcsname
```

If ⟨*ps2*⟩ is fancyhdr-based, ⟨*ps1*⟩ will also be fancyhdr-based, otherwise it is not.

```
     \@ifundefined{f@nch@ps@#2-is-fancyhdr}{%
       \expandafter\let\csname f@nch@ps@#1-is-fancyhdr\endcsname\@undefined
     }{%
       \@namedef{f@nch@ps@#1-is-fancyhdr}{}%
     }%
   }%
}
```

(*End of definition for* `\fancypagestyleassign`. *This function is documented on page 5.*)

`\ps@fancydefault`

This is page style `fancydefault`. It is in fact page style `fancy` with all the defaults embedded, including the relevant definitions of `\chaptermark` and `\[sub]sectionmark`. This is in contrast with page style `fancy` that gets all its settings from the environment. It is defined with `\fancypagestyle*`.

```
\fancypagestyle*{fancydefault}{\f@nch@initialise}
```

(*End of definition for* `\ps@fancydefault`*.*)

`\fancyhdrbox`

`\fancyhdrbox[`⟨*alignment*⟩`][`⟨*width*⟩`]{`⟨*lines separated by* `\\`⟩`}`

This command creates a `\halign` inside a vertical box (`\vbox` or `\vtop`).

We need some variables, but these don't have to be declared. They are characterised by `@@` in their name.

`\f@nchdrbox@@v` – vertical alignment (`T`, `t`, `c`, `b`, `B`)

`\f@nchdrbox@@h` – horizontal alignment (`l`, `c`, `r`)

`\f@nchdrbox@@pre` – code to be inserted before the first row, a 'topstrut' or `\vspace{0pt}`

`\f@nchdrbox@@postx` – code to be executed at the end of the last row, possibly a 'botstrut'

`\f@nchdrbox@@posty` – code to be executed after the `\halign`, possibly a `\vspace{0pt}`

`\f@nchdrbox@@crstrut` – a 'strut' to be inserted at each `\\` in the `\halign`

`\f@nchdrbox@@halignto` – This is either empty, if no ⟨*width*⟩ argument is given, or '`to` ⟨*width*⟩' if it is given.

A 'strut' is a TEX construct to keep the baselines of the lines in a text on a fixed distance. It normally is an invisible rule of `width 0pt`, `height 0.7\baselineskip` and `depth 0.3\baselineskip`. Therefore the struts are dependent of the font of the text. But for the `\fancyhdrbox` alignments `T` and `B` we need special 'topstrut' (which only has the `height`) part, or a 'botstrut' (which only has the `depth`) part. For example with the `T` alignment, there should be no strut on the first line, because then we don't want the extra space above this line. We use instead a `vspace{0pt}`. But we need the `depth` part because we want the extra space below the line. Similar for the `B` alignment, but then in the opposite direction.

`\f@nchdrbox@topstrut`
`\f@nchdrbox@botstrut`

```
\def\f@nchdrbox@topstrut{\vrule height\ht\strutbox width\z@}
\def\f@nchdrbox@botstrut{\vrule depth\dp\strutbox width\z@}
```

`\f@nchdrbox@nostrut`

At each `\\` `\f@nchdrbox@@crstrut` will be inserted. It will be a normal `\strut`, except in the first row when the alignment is `T`; then it will be a 'botstrut', and moreover, we will insert a `\vspace{0pt}` at the top of the box. The macro `\f@nchdrbox@nostrut` will set this up. The assignment to `\f@nchdrbox@@crstrut` will be local to the `\halign` cell, so after the `\\` it will be reset to the default.

```
\def\f@nchdrbox@nostrut{\noalign{\vspace{0pt}}\let\f@nchdrbox@@crstrut\f@nchdrbox@botstrut}
```

Now we start the only user command in the part: `\fancyhdrbox`. The code is run in a group so that changes to variables are local. This is necessary in case we use nested `\fancyhdrbox`es.

First we set the variables `\f@nchdrbox@@pre`, `\f@nchdrbox@topstrut`, `\f@nchdrbox@@posty`, and `\f@nchdrbox@@crstrut` to their default values. Then we test if the second optional argument (⟨*width*⟩) was given, and if so, record this in `\f@nchdrbox@@halignto`. We put the ⟨*width*⟩ value in a length variable with `\setlength` so that we can support calc-style values.

And then we check if the first optional argument ⟨*alignment*⟩ is empty. In that case we use `cl` instead.

```
\NewDocumentCommand{\fancyhdrbox}{ O{cl} o m }{%
\begingroup
  \let\f@nchdrbox@@pre\f@nchdrbox@topstrut
  \let\f@nchdrbox@@postx\f@nchdrbox@botstrut
  \let\f@nchdrbox@@posty\relax
  \let\f@nchdrbox@@crstrut\strut
  \IfNoValueTF{#2}%
    {\let\f@nchdrbox@@halignto\@empty}%
    {\setlength\@tempdima{#2}%
      \def\f@nchdrbox@@halignto{to\@tempdima}}%
  \def\@tempa{#1}%
  \ifx\@tempa\@empty
    \f@nchdrbox@align cl\@nil{#3}%
  \else
    \f@nchdrbox@align #1\@nil{#3}%
  \fi
\endgroup
}
```

`\f@nchdrbox@cr`
`\@f@nchdrbox@xcr`
`\@f@nchdrbox@argc`
`\@f@nchdrbox@xargc`
`\@f@nchdrbox@yargc`

This is the definition for `\\` inside `\fancyhdrbox`, `\\*` does nothing special here, but we accept it anyway. The code is mostly copied from the `tabular` code from the LaTeX kernel, but simplified, and the names of the macros are changed so that we don't rely on internals in the LaTeX kernel that may change. The trick with the ``\ifnum0=` `` allows to get unbalanced braces in a macro.

```
\protected\def\f@nchdrbox@cr{%
  {\ifnum0=`}\fi\@ifstar\@f@nchdrbox@xcr\@f@nchdrbox@xcr}

\def\@f@nchdrbox@xcr{%
  \unskip\f@nchdrbox@@crstrut
  \@ifnextchar[\@f@nchdrbox@argc{\ifnum0=`{\fi}\cr}%
}

\def\@f@nchdrbox@argc[#1]{%
  \ifnum0=`{\fi}%
    \ifdim #1>\z@
      \unskip\@f@nchdrbox@xargc{#1}%
    \else
      \@f@nchdrbox@yargc{#1}%
    \fi}

\def\@f@nchdrbox@xargc#1{\@tempdima #1\advance\@tempdima \dp \strutbox
   \vrule \@height\z@ \@depth\@tempdima \@width\z@ \cr}

\def\@f@nchdrbox@yargc#1{\cr\noalign{\setlength\@tempdima{#1}\vskip\@tempdima}}
```

Processing for the vertical alignment options `T`, `t`, `c`, `b`, `B`.

`\f@nchdrbox@T`
`\f@nchdrbox@t`
`\f@nchdrbox@c`
`\f@nchdrbox@b`
`\f@nchdrbox@B`

`T` set `\f@nchdrbox@@pre` to 'nostrut' and execute the `t` code

`t` set vertical alignment to `t` and horizontal to `l`

`c` set both vertical and horizontal alignment to `c`

- `b` set vertical alignment to `b` and horizontal to `l`
- `B` set `\f@nchdrbox@@postx` to do nothing and `\f@nchdrbox@@posty` to `\vspace{0pt}` and execute the `b` code

The horizontal alignments are defaults, they may be changed by processing the horizontal argument, if present.

```
\def\f@nchdrbox@T{\let\f@nchdrbox@@pre\f@nchdrbox@nostrut
                  \f@nchdrbox@t}
\def\f@nchdrbox@t{\def\f@nchdrbox@@v{t}\def\f@nchdrbox@@h{l}}
\def\f@nchdrbox@c{\def\f@nchdrbox@@v{c}\def\f@nchdrbox@@h{c}}
\def\f@nchdrbox@b{\def\f@nchdrbox@@v{b}\def\f@nchdrbox@@h{l}}
\def\f@nchdrbox@B{\let\f@nchdrbox@@postx\relax
                  \def\f@nchdrbox@@posty{\vspace{0pt}}%
                  \f@nchdrbox@b}
```

`\f@nchdrbox@align{`⟨*vert*⟩`}{`⟨*hor*⟩`}\@nil{`⟨*lines*⟩`}`

`\f@nchdrbox@align`

The internal processing for the `\halign` in a vertical box.

```
ALGORITHM \f@nchdrbox@align:
(v = vertical position; h = horizontal position)
IF #1 in {T,t,c,b,B}
THEN v := #1; h := (if #1 == c then c else l fi) (coded in \f@nchdrbox@⟨#1⟩)
   (The h value is the default in case #2 is empty)
   if #2 is not empty then h := #2 fi
ELSE (#1 not in {T,t,c,b,B} – it must be a horizontal alignment)
   v := c
   h := #1
FI
if h not in {l,c,r} then ERROR fi
```

Set the `\halign` in a `\vtop` (for `T/t` alignment) or `\vbox` (for others). This box is put into `\box0` because we have to change it for the vertical alignment `c`. For the others it isn't necessary, but it just makes the code easier to do it anyway. We also insert the `\f@nchdrbox@@pre`, `\f@nchdrbox@@postx` and `\f@nchdrbox@@posty` variables in the proper places. The rest of the code is roughly based on the `tabular` code in the LaTeX kernel.

```
\long\def\f@nchdrbox@align#1#2\@nil#3{%
  \f@nch@ifin{#1}{TtcbB}{%
    \@nameuse{f@nchdrbox@#1}%
    \def\@tempa{#2}%
    \ifx\@tempa\@empty\else \def\f@nchdrbox@@h{#2}\fi
  }%
  {\def\f@nchdrbox@@v{c}\def\f@nchdrbox@@h{#1}}%
  \expandafter\f@nch@ifin\expandafter{\f@nchdrbox@@h}{lcr}{}%
  {\PackageError{fancyhdr}{\string\fancyhdrbox: Illegal char `\f@nchdrbox@@h'\MessageBreak
                           in alignment argument}{}}%
  \let\\\f@nchdrbox@cr
  \setbox0=\if \f@nchdrbox@@v t\vtop
  \else \vbox
  \fi
  {%
     \ialign \f@nchdrbox@@halignto
     \bgroup \relax
     {\if \f@nchdrbox@@h l\hskip 1sp\else \hfil \fi
       \ignorespaces ##\unskip
       \if\f@nchdrbox@@h r\else \hfil \fi
     }%
     \tabskip\z@skip \cr
     \f@nchdrbox@@pre
     #3\unskip \f@nchdrbox@@postx
     \crcr
     \egroup
     \f@nchdrbox@@posty
  }%
```

If the vertical alignment is `c`, we calculate the total height + depth of the resulting `\box0` and then set the depth and height of this box each to half of this value. This way the box will be vertically centered. We don't use `\vcenter` for this, because it centers the box on the *math axis*, which doesn't make sense here, and sometimes gives a different vertical positioning (not exactly centered).

```
  \if\f@nchdrbox@@v c\@tempdima=\ht0\advance\@tempdima\dp0%
    \ht0=0.5\@tempdima\dp0=0.5\@tempdima\fi
```

Finally we put the box in horizontal mode in the running text.

```
  \leavevmode \box0
}
```

(*End of definition for* `\fancyhdrbox` *and others. These functions are documented on page* 5.)

The (really outdated) document class newlfm uses some internal fancyhdr commands that have gotten new names. So here we check if that class is loaded and then we redefine the affected newlfm macros. We have to do some of the redefinitions in `\AtBeginDocument`, as fancyhdr is loaded before the affected macros are defined. Also the macro `\@zfancyhead` is only called once, with wrong (outdated) parameters, so instead of changing the call of the macro, we substitute the right parameters inline.

```
\@ifclassloaded{newlfm}
{
  \let\ps@@empty\f@nch@ps@empty
  \AtBeginDocument{%
    \renewcommand{\@zfancyhead}[5]{\relax\hbox to\headwidth{\f@nch@reset
      \@zfancyvbox\headheight{\hbox
        {\rlap{\parbox[b]{\headwidth}{\raggedright\f@nch@olh}}\hfill
          \parbox[b]{\headwidth}{\centering\f@nch@olh}\hfill
          \llap{\parbox[b]{\headwidth}{\raggedleft\f@nch@orh}}}%
        \zheadrule}}\relax}%
  }
}
{}
```

</fancyhdr>

45 extramarks.sty

<*extramarks>

This package gives you extra marks, that you can define, set and use in your page headers and footers. It is based on the new LaTeX marks mechanism as introduced in the 2022/06/01 LaTeX release. If your LaTeX implementation is older it will fallback to extramarks version 4.

Provide a rollback to earlier version.

```
\DeclareRelease{v4}{2024/11/30}{extramarks-v4.sty}
\DeclareCurrentRelease{v5}{2025/02/07}
```

```
\@ifundefined{NewMarkClass}
              {\PackageWarningNoLine{extramarks}{%
**************************************************\MessageBreak
Your LaTeX installation is too old for extramarks version 5.\MessageBreak
We will fallback to extramarks version 4 now.\MessageBreak
Please note that some commands will not be available,\MessageBreak
and that the functionality may be slightly different.\MessageBreak
You are advised to update your LaTeX installation.\MessageBreak
**************************************************}
              \RequirePackage{extramarks-v4}\endinput}{}
```

We also do a sanity check for the package multicol. If it is too old it will not work correctly with the new extramarks. In that case extramarks-v4 should be used instead. So in that case we give a warning and then load package extramarks-v4. First `\extramarks` must be made undefined, otherwise loading extramarks-v4 will give an error.

```
\AtBeginDocument{%
  \IfPackageLoadedT{multicol}%
    {\IfPackageAtLeastF{multicol}{2024-11-21}{%
      \PackageWarningNoLine{extramarks}{%
        You are using package ‘extramarks’ with a version\MessageBreak
        of ‘multicol’ that is too old. The new version\MessageBreak
        of ‘multicol’ will be released on June 1, 2025.\MessageBreak
        We will fallback to extramarks version 4 now.}%
      \let\extramarks\undefined
      \RequirePackage{extramarks-v4}
      }%
    }%
}
```

`\@mkboth`

Initialization of `\@mkboth`, so that it will pick up changes to `\markboth`

```
\ifx\@mkboth\@gobbletwo\else\def\@mkboth{\protect\markboth}\fi
```

(*End of definition for* `\@mkboth`.)

`extramarks-left`
`extramarks-right`

`extramarks-left`
`extramarks-right`

We need two mark classes. We call them `extramarks-left` and `extramarks-right`.

```
\NewMarkClass{extramarks-left}
\NewMarkClass{extramarks-right}
```

(*End of definition for* `extramarks-left` *and* `extramarks-right`*. These variables are documented on page 7.*)

`\extramarks`

This command is used to define the extra marks.

```
\newcommand\extramarks[2]{%
  \InsertMark{extramarks-left}{#1}%
  \InsertMark{extramarks-right}{#2}}
```

(*End of definition for* `\extramarks`*. This function is documented on page 7.*)

`\extramarksleft`
`\extramarksright`

These commands can be used to set the two marks independently. These are only available in extramarks version 4.5 or later.

```
\newcommand\extramarksleft[1]{%
  \InsertMark{extramarks-left}{#1}}
\newcommand\extramarksright[1]{%
  \InsertMark{extramarks-right}{#1}}
```

(*End of definition for* `\extramarksleft` *and* `\extramarksright`. *These functions are documented on page 7.*)

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

The new marks to be used in the headers and/or footers (based on the standard marks info).

```
\newcommand\firstleftmark{\FirstMark{2e-left}}
\newcommand\lastrightmark{\LastMark{2e-right}}
```

We first define the following commands with `\newcommand` to detect possible name clashes; then we redefine them with `\let`.

```
\newcommand\firstrightmark{\rightmark}
\let\firstrightmark \rightmark
\newcommand\lastleftmark{\leftmark}
\let\lastleftmark \leftmark
```

(*End of definition for* `\firstleftmark` *and others. These functions are documented on page 7.*)

`\firstleftxmark`
`\firstrightxmark`
`\topleftxmark`
`\toprightxmark`
`\lastleftxmark`
`\lastrightxmark`
`\firstxmark`
`\lastxmark`
`\topxmark`

The new extra marks.

```
\newcommand\firstleftxmark{\FirstMark{extramarks-left}}
\newcommand\lastrightxmark{\LastMark{extramarks-right}}
\newcommand\firstrightxmark{\FirstMark{extramarks-right}}
\newcommand\topleftxmark{\TopMark{extramarks-left}}
\newcommand\toprightxmark{\TopMark{extramarks-right}}
\newcommand\lastleftxmark{\LastMark{extramarks-left}}
```

We first define the following commands with `\newcommand` to detect possible name clashes; then we redefine them with `\let`.

```
\newcommand\firstxmark{\firstleftxmark}
\let\firstxmark\firstleftxmark
\newcommand\lastxmark{\lastrightxmark}
\let\lastxmark\lastrightxmark
\newcommand\topxmark{\topleftxmark}
\let\topxmark\topleftxmark
```

(*End of definition for* `\firstleftxmark` *and others. These functions are documented on page 7.*)
</extramarks>

46 extramarks-v4.sty

<*extramarks-v4>

`\@temptokenb`

A token register to store some marks information

```
\ifx\@temptokenb\undefined \csname newtoks\endcsname\@temptokenb\fi
```

(*End of definition for* `\@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}
```

(*End of definition for* `\unrestored@protected@xdef`.)

`\markboth`

Our own definition of `\markboth`, mainly because `\@markboth` gets more parameters. First the definition for modern LaTeX distributions.

```
\ifdefined\ExplSyntaxOn
\ExplSyntaxOn
\DeclareRobustCommand*\markboth[2]{%
  \begingroup
    \let\label\relax \let\index\relax \let\glossary\relax
    \expandafter\@markboth\@themark{#1}{#2}%
    \@temptokena \expandafter{\@themark}%
    \ifdefined\mark_insert:nn
    % 3 new lines to set the new marks
      \mark_insert:nn{2e-left}{#1}
      \mark_insert:nn{2e-right}{#2}
      \tl_if_empty:nF{#2}{ \mark_insert:nn{2e-right-nonempty}{#2} }
    \fi
    \mark{\the\@temptokena}%
  \endgroup
  \if@nobreak\ifvmode\nobreak\fi\fi}
\ExplSyntaxOff
```

If we are with a pre-LaTeX3 kernel, we use the definition from an older version of extramarks.

```
\else
\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}
\fi
```

(*End of definition for* `\markboth`.)

`\@mkboth`

Initialization of `\@mkboth`, so that it will pick up changes to `\markboth`

```
\ifx\@mkboth\@gobbletwo\else\def\@mkboth{\protect\markboth}\fi
```

(*End of definition for* `\@mkboth`.)

`\markright`

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

(*End of definition for* `\markright`.)

`\@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}}
```

(*End of definition for* `\@markboth`.)

`\@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}}
```

(*End of definition for* `\@markright`.)

`\@leftmark`
`\@rightmark`

Internal macros to get the standard marks.

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

(*End of definition for* `\@leftmark` *and* `\@rightmark`*.*)

`\leftmark`
`\rightmark`
`\firstleftmark`
`\lastrightmark`
`\firstrightmark`
`\lastleftmark`

The standard marks + the new ones (based on the standard marks info). We provide `\IfFormatAtLeastTF` in case we have a rather old LaTeX format (in which case the test will always be false). If the LaTeX format is 2025-06-01 or later, `\leftmark` and `\rightmark` have definitions based upon the new marks, so we should not redefine these even in the extramarks-v4 mode.

```
\providecommand\IfFormatAtLeastTF{\@ifl@t@r\fmtversion}
\IfFormatAtLeastTF{2025-06-01}{}{%
  \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
```

(*End of definition for* `\leftmark` *and others.*)

`\@themark`

This is where the marks information is stored.

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

(*End of definition for* `\@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}
```

(*End of definition for* `\extramarks`*. This function is documented on page 7.*)

`\@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}}}
```

(*End of definition for* `\@markextra`*.*)

`\extramarksleft`

This command is used to define the left extra mark. As this is not independent from the other marks, it is not perfect.

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

(*End of definition for* `\extramarksleft`*. This function is documented on page* 7*.*)

`\@extramarksleft` Internal macro to store the left mark in the marks storage.
Note: Put `#1#2` and `#4`in toks registers.

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

(*End of definition for* `\@extramarksleft`*.*)

`\extramarksright` This command is used to define the right extra mark. As this is not independent from the other marks, it is not perfect.

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

(*End of definition for* `\extramarksright`*. This function is documented on page* 7*.*)

`\@extramarksright` Internal macro to store the right mark in the marks storage.
Note: Put `#1#2#3` in toks register.

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

(*End of definition for* `\@extramarksright`*.*)

`\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
```

(*End of definition for* `\firstleftxmark` *and others. These functions are documented on page* 7*.*)

`\@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}
```

(*End of definition for* `\@tleftxmark` *and* `\@rightxmark`.)
</extramarks-v4>

47 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. 119, 120
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`. 119, 120
fancyhdr v 2.0
  General: version 2.0 Release. . . 119, 120
extramarks v2.1
  General: Added a `\ProvidesPackage` line.
    Updated contact information. 119, 120
extramarks v3.9
  General: Unify version number with fancyhdr.sty. 119, 120
extramarks v3.9a
  General: Restore

`\newtoks\@temptokenb` 120

extramarks v4.0.3

`\@mkboth`: Initialize definition of `\@mkboth` to `\def\@mkboth{\protect\markboth}` if it wasn't equal to `\@gobbletwo` so that it will pick up changes to `\markboth` 119, 121

extramarks v4.4

`\markboth`: Add setting the new style marks for `\leftmark` (`2e-left`) and `\rightmark` (`2e-right` and `2e-right-nonempty`). We do this only if the new marks are defined in the LaTeX kernel. 121

extramarks v5.0

General: Check if extramarks version 5 is not used with a too old version of multicol. 119

Make `\newtoks\@temptokenb` conditional 120

New implementation with independent marks, and fallback option to earlier version 4. 118

extramarks v5.1

General: Bug fix: use `\IfPackageLoadedT`. 119

extramarks v5.2

General: Assume `\DeclareRelease` is present (LaTeX version at lease 2018-04-01). 118

extramarks-v4 v4.5

General: Add commands `\extramarksleft` and `\extramarksright` for compatibility with extramarks version 5. 122

Don't redefine `\leftmark` and `\rightmark` in LaTeX kernel 2025-06-01 and later. 122

fancyhdr v1.4

General: Correction for use with `\reversemarginpar` 90

fancyhdr v1.5

General: Added the `\iftopfloat`, `\ifbotfloat` and `\iffloatpage` commands 90

fancyhdr v1.6

General: Reset single spacing in headers/footers for use with setspace.sty or doublespace.sty 90

fancyhdr v1.7

General: Changed `\let\@mkboth\markboth` to `\def\@mkboth{\protect\markboth}` to make it more robust. 90

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). 90

fancyhdr v1.9

General: The proposed `\renewcommand{\headrulewidth}{\iffloatpage...` construction in the doc did not work properly with the `fancyplain` style. 90

fancyhdr v1.91

General: The definition of `\@mkboth` wasn't restored on subsequent `\pagestyle{fancy}`'s. 90

fancyhdr v1.92

General: The sequence `\pagestyle{fancyplain}` `\pagestyle{plain}` `\pagestyle{fancy}` would erroneously select the plain version. 90

fancyhdr v1.93

General: `\fancypagestyle` command added. 90

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). 90

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. 90

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. 90

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. 90

fancyhdr v1.98

General: Added % after the line `\def\nouppercase` 90

fancyhdr v1.99

General: This is the alpha version of fancyhdr 2.0
Introduced the new commands `\fancyhead`, `\fancyfoot`, and `\fancyhf`. Changed `\headrulewidth`, `\footrulewidth`, `\footruleskip` to macros rather than length parameters, In this way they can be conditionalized and they don't consume length registers. There is no need to have them as length registers unless you want to do calculations with them, which is unlikely. Note that this may make some uses of them incompatible (i.e., if you have a file that uses `\setlength` or `\xxxx=`) 90

fancyhdr v1.99a

General: Added a few more % signs. . 90

fancyhdr v1.99b

General: Changed the syntax of `\f@nch@for` to be resistant to catcode changes of `:=`.
Removed the `[1]` from the defs of `\lhead` etc. because the parameter is consumed by the `\@[xy]lhead` etc. macros. 90

fancyhdr v1.99c

General: Corrected `\nouppercase` to also include the protected form of `\MakeUppercase`.
`\global` added to manipulation of `\headwidth`.
`\iffootnote` command added.
Some comments added about `\f@nch@head` and `\f@nch@foot`. . . 90

fancyhdr v1.99d

General: Changed the default `\ps@empty` to `\ps@@empty` in order to allow `\fancypagestyle{empty}` redefinition. 90

fancyhdr v2.0

General: Added LPPL license clause.
A check for `\headheight` is added.
An errormessage is given (once) if the header is too large. Empty headers don't generate the error even if `\headheight` is very small or even 0pt.
Warning added for the use of 'E' option when `twoside` option is not used. In this case the 'E' fields will never be used. 90

fancyhdr v2.1beta

General: New command: `\fancyhfoffset[place]{length}` defines offsets to be applied to the header/footer to let it stick into the margins (if length > 0). `place` is like in `\fancyhead`, except that only `E,O,L,R` can be used. This replaces the old calculation based on `\headwidth` and the marginpar area. `\headwidth` will be dynamically calculated in the headers/footers when this is used. 90

fancyhdr v2.1beta2

General: `\fancyhfoffset` now also takes `H,F` as possible letters in the argument to allow the header and footer widths to be different.
New commands `\fancyheadoffset` and `\fancyfootoffset` added comparable to `\fancyhead` and `\fancyfoot`.
Errormessages and warnings have been made more informative. 90

- fancyhdr v2.1
  - General: The defaults for `\footrulewidth`, `\plainheadrulewidth` and `\plainfootrulewidth` are changed from `\z@skip` to 0pt. In this way when someone inadvertently uses `\setlength` to change any of these, the value of `\z@skip` will not be changed, rather an errormessage will be given. 90
- fancyhdr v3.0
  - General: Release of version 3.0. 90
- fancyhdr v3.1
  - General: Added '`\endlinechar=13`' to `\f@nch@reset` to prevent problems with `\includegraphics` in header/footer when `verbatiminput` is active. 90
- fancyhdr v3.10
  - `\f@nch@foot`: Move `\footruleskip` outside of the `\footrule` definition. 105
    - Put `\footrule` in a `\vbox` to accommodate for flexible footrules. 105
    - Use `\unvbox` on the footrule `\vbox` to preserve vertical spacing. 105
  - `\f@nch@forc`: Use `\newcommand` instead of `\def`. 92
  - `\f@nch@vbox`: Don't use `\global\setlength`. 104
    - Use `\newcommand` instead of `\def`. 104
  - `\footrule`: Move `\footruleskip` outside of the `\footrule` definition and remove useless `\vskip` at the top. 109
- fancyhdr v3.2
  - General: Reset `\everypar` (the real one) in `\f@nch@reset` because spanish.ldf does strange things with `\everypar` between « and ». . 90
- 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. 90
- fancyhdr v3.4
  - General: Replace `\rm` by `\normalfont\rmfamily` and `\sl` by `\normalfont\slshape`. 90
- fancyhdr v3.5
  - General: Don't define `\footruleskip` if it is already defined. 90
- fancyhdr v3.6
  - General: Added a `\ProvidesPackage` line.
    Updated contact information. . . . 90
- fancyhdr v3.7
  - General: Removed `\normalfont` from default values, as every field is already initialised with `\normalfont`.
    Set `\hsize` to `\headwidth` in header/footer. 90
- 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. 90
- fancyhdr v3.9
  - General: Put everything in a `.dtx` file. 90
    - Rename some macros to have 'f@nch@' in their names, to get a more uniform naming scheme for internal macros. 90
  - `\cfoot`: Let `\newcommand` do the handling of the optional parameter. 97
  - `\chead`: Let `\newcommand` do the handling of the optional parameter. 97
  - `\fancyfoot`: Let `\newcommand` do the handling of the optional parameter. 93
  - `\fancyfootoffset`: Let `\newcommand` do the handling of the optional parameter. 93
  - `\fancyhead`: Let `\newcommand` do the handling of the optional parameter. 93
  - `\fancyheadoffset`: Let `\newcommand` do the handling of the optional parameter. 93
  - `\fancyhf`: Let `\newcommand` do the handling of the optional parameter. 93
  - `\fancyhfoffset`: Let `\newcommand` do the handling of the optional parameter. 93
  - `\lfoot`: Let `\newcommand` do the handling of the optional parameter. 97
  - `\lhead`: Let `\newcommand` do the handling of the optional parameter. 97
  - `\rfoot`: Let `\newcommand` do the handling of the optional parameter. 97
  - `\rhead`: Let `\newcommand` do the handling of the optional parameter. 97

- fancyhdr v4.0
  - General: Added `headings` and `myheadings` options. 90
    - Process package options. 92
  - `\f@nch@@pagestyle`: Make the definition of `\f@nch@pagestyle` `\long`. 113
  - `\f@nch@foot`: `\fancyfootinit` initialisation code added and `\f@nch@reset` moved up. 105
  - `\f@nch@gbl`: Remove the `\global` in definitions 90
  - `\f@nch@head`: `\fancyheadinit` initialisation code added and `\f@nch@reset` moved up. 104
    - Parameter `\headruleskip`. 104
  - `\f@nch@initialise`: Put all the initialisation code in `\f@nch@initialise` 108
  - `\f@nch@pagestyle`: Added optional base style argument. 112
  - `\f@nch@ps@empty`: Rename `\ps@@empty` to `\f@nch@ps@empty` 110
  - `\f@nch@vbox`: Don't adjust the `\headheight`/`\footskip`, except when the `compatV3` option is given. 104
    - Don't check when the `nocheck` option is given. 104
  - `\fancycenter`: Macro `\fancycenter` added 101
  - `\headruleskip`: Parameter `\headruleskip`. 98
  - `\iff@nch@check`: Implement the `nocheck` option 90
  - `\iff@nch@compatViii`: Implement the `compatV3` option 90
  - `\ps@f@nch@fancycore`: Rename `\ps@@fancy` to `\ps@f@nch@fancycore` 110
  - `\ps@f@nch@fancyproto`: Reorganise `\ps@fancy` 109
  - `\ps@fancydefault`: Added `\ps@fancydefault` 115
- fancyhdr v4.0.2
  - `\f@nch@fancyhf`: Make `\f@nch@fancyhf` `\long`. 94
  - `\f@nch@fancyhfoffs`: Change the offset length variables `\f@nch@O@xyz` to `\f@nch@offset@xyz` 94
  - `\f@nch@foot`: Added `\leavevmode\ignorespaces` to each header/footer field. The `\leavevmode` prevents a bug when a field starts with a `\color` command. The `\ignorespaces` skips initial spaces in the parameter, as is usual in a `\parbox`, for backwards compatibility. 105
  - `\f@nch@forc`: Make `\f@nch@rc` `\long`. 92
  - `\f@nch@gbl`: Untangle the code for`\f@nch@gbl` and the `compatV3` option 90
  - `\f@nch@head`: Added `\leavevmode\ignorespaces` to each header/footer field. The `\leavevmode` prevents a bug when a field starts with a `\color` command. The `\ignorespaces` skips initial spaces in the parameter, as is usual in a `\parbox`, for backwards compatibility. 104
  - `\f@nch@headwidth`: Change the offset length variables `\f@nch@O@xyz` to `\f@nch@offset@xyz` 98
  - `\f@nch@vbox`: If the option `[nocheck]` is given, just keep quiet and don't change the `\headheight`/`\footskip` even if the `[compatV3]` option is given. . 104
    - Shorten one sentence in the warning message. 104
- fancyhdr v4.0.3
  - `\ps@headings`: Changed definition of `\@mkboth` from `\let\@mkboth\markboth` to `\def\@mkboth{\protect\markboth}` so that it will pick up changes to `\markboth` 91
- fancyhdr v4.1
  - General: Support for class newlfm. . 118
  - `\f@nch@fancyhf`: Implement `twoside` option. 94
  - `\f@nch@fancyhf@Echeck`: Implement `twoside` option. 93
  - `\f@nch@fancyhfoffs`: Implement `twoside` option. 94
  - `\f@nch@reset`: Change `\nouppercase` to work with new definition of `\MakeUppercase`. 101
  - `\iff@nch@twoside`: Implement `twoside` option. 90
  - `\ps@f@nch@fancycore`: Implement `twoside` option. 110
- fancyhdr v4.2
  - `\f@nch@reset`: Reset catcodes to their default values in order to facilitate `\input` in headers/footers when `verbatim` is active. (Issue # 8

https://github.com/pietvo/fancyhdr/issues/8.) 101

fancyhdr v4.3
- \f@nch@everypar: Changed \f@nch@everypar. If the LaTeX kernel has expl3, use \tex_ everypar:D, and reset \par, \@@par and \endgraf to their original TEX definitions, so that no paragraph hooks will intrude in fancyhdr code. 99
- \f@nch@reset: Remove pre-NFSS stuff 101
  - Replace \f@nch@everypar by \f@nch@resetpar. 101

fancyhdr v4.3.1
- \f@nch@everypar: Also reset \everypar to its original TEX value \tex_ everypar:D in \f@nch@resetpar, otherwise environments based on \trivlist will not work properly in fancyhdr headers and footers. 99

fancyhdr v4.5
- \f@nch@foot: Hooks added 105
- \f@nch@head: Hooks added 104
- \f@nch@restore@parhook: We use a less fragile way to disable paragraph hooks, thereby partially reverting the solution in version v4.3 and v4.3.1. This is less intruding in the hook system, and especially it doesn't affect kernel hooks, only user provided ones. We check the kernel version to see if it support paragraph hooks. . . . 99
- fancyhdr/foot/end: Hooks added . 103

fancyhdr v5.0
- General: Define \iff@nch@pagestyle@star to record if \fancypagestyle* is used. 112
  - Define \newtoks\@temptokenb . . 112
  - Removed \f@nch@errmsg and \f@nch@warning and used \PackageError and \PackageWarning directly. 92
- \@makecol: Change the internal variables \topfloat and \botfloat to localised ones \f@nch@topfloat and \f@nch@botfloat. 112
- \f@nch@@pagestyle: Add \fancypagestyle* 113
- \f@nch@foot: \vskip⟨*length*⟩ added if \fancyfootalign{⟨*length*⟩} was given. 105
  - Support for \fancyheadwidth, \fancyfootwidth and \fancyhfwidth. 105
- \f@nch@head: Support for \fancyheadwidth, \fancyfootwidth and \fancyhfwidth. 104
- \f@nch@height: Length variable \f@nch@height added 103
- \f@nch@offselh: Change the offset length variables \f@nch@O@xyz to \f@nch@offset@xyz 111
- \f@nch@pagestyle@setup: Add \f@nch@pagestyle@setup macro for \fancypagestyle* 113
  - Also save the offset length variables \f@nch@offset@xyz with \setlength 113
  - Implement \fancyhfwidth etc., store ⟨*width*⟩. 113
  - Replace \myname with \temp@a . . 113
- \f@nch@setoffs: Use \def instead of \let to make it easier to pick up these macros for \fancypagestyle* 111
- \f@nch@vbox: Set \f@nch@height to height of box 104
- \fancyfootalign: Macro \fancyfootalign, \iff@nch@footalign and length variable \f@nch@footalignment added 103
- \fancyhdrbox: Macro \fancyhdrbox added 115
- \fancyhdrsettoheight: Macro \fancyhdrsettoheight added . . 103
- \fancyhfwidth: New commands \fancyheadwidth, \fancyfootwidth and \fancyhfwidth. 95
- \fancypagestyle: Add \fancypagestyle* 112
- \fancypagestyleassign: Macro \fancypagestyleassign added . 114
- \iff@nch@compatViii: Add warning that the compatV3 option is deprecated 90
- \iffootnote: Replace \empty with \@empty 112
- \ps@f@nch@fancycore: Support for \fancyheadwidth, \fancyfootwidth and \fancyhfwidth. 110
- \ps@fancydefault: Defined \ps@fancydefault with

\fancypagestyle* 115

fancyhdr v5.1
- \f@nch@foot: Add paragraph hook restore code (\f@nch@restorepar). 105
- \f@nch@head: Add paragraph hook restore code (\f@nch@restorepar). 104
- \f@nch@restore@parhook: A new way to disable the paragraph hooks in headers and footers. 99

fancyhdr v5.1.1
- \f@nch@restore@parhook: Use low-level commands to reset paragraph hooks to prevent LaTeX error when running with latex3 debugging. 99

fancyhdr v5.2
- General: Conditionally require package xparse. 90
  - Remove fallback definition of \MakeUppercase. 108
  - Use \@ifundefined instead of \f@nch@ifundefined. 90
- \@chapapp: Use \@ifundefined instead of \f@nch@ifundefined. . 108
- \f@nch@@pagestyle: Use \@ifundefined instead of \f@nch@ifundefined. 113
- \f@nch@def: Inline \if@nch@mpty. . . 90
  - Remove \leavevmode, because the initialisation code also inserts a \leavevmode. 90
- \f@nch@foot: Implement \fancyhfwidth etc. ⟨*alignment*⟩ option. 105
- \f@nch@head: Implement \fancyhfwidth etc. ⟨*alignment*⟩ option. 104
- \f@nch@hfbox: Implement \fancyhfwidth etc. ⟨*alignment*⟩ option. 106
- \f@nch@hfbox@center: Implement \fancyhfwidth etc. ⟨*alignment*⟩ option. 106
- \f@nch@hfbox@fit: Implement \fancyhfwidth etc. ⟨*alignment*⟩ option. 107
- \f@nch@pagestyle@setup: Implement \fancyhfwidth etc., store ⟨*alignment*⟩. 113
- \f@nch@reset: Use \@ifundefined instead of \f@nch@ifundefined. . 101
- \f@nch@restore@parhook: Use official interface to reset the paragraph hooks. 99
- \f@nchdrbox@align: Check for legal alignment argument added. 117
- \fancyhfwidth: Implement * version of these commands. 95
  - Second optional parameter ⟨*alignment*⟩. 95
- \fancypagestyleassign: Remove \PackageNote. 114
  - Use \@ifundefined instead of \f@nch@ifundefined. 114
- \footruleskip: Use \@ifundefined instead of \f@nch@ifundefined. . . 98
- \headruleskip: Use \@ifundefined instead of \f@nch@ifundefined. . . 98
- \ps@f@nch@fancycore: Implement \fancyhfwidth etc. ⟨*alignment*⟩ option. 110
- \ps@headings: Use \@ifundefined instead of \f@nch@ifundefined. . . 91
- \subsectionmark: Use \@ifundefined instead of \f@nch@ifundefined. . 108

Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.