Addendum

ADDENDUM

The Memoir Class

for

Configurable Typesetting

User Guide

Peter Wilson

THP
The Herries Press

The Herries Press, Normandy Park, WA.

Printed in the World

The paper used in this publication may meet the minimum requirements of the American National Standard for Information Sciences — Permanence of Paper for Printed Library Materials, ANSI Z39.48–1984.

10 09 08 07 06 05 04 03 02 15 14 13 12 11 10

First edition:	November 2002
Second impression, with minor additions	December 2002
Third impression, with additions	January 2003
Fourth impression, with additions	January 2003
Fifth impression, with minor additions	February 2003
Sixth impression, with minor additions	February 2003
Seventh impression, with minor additions	April 2003
Eighth impression, with minor additions	June 2003
Ninth impression, with minor additions	July 2003
Tenth impression, with minor additions	September 2003

addendum, *n.* [L., gerundive of *addere:* see ADD] **1.** a thing added or to be added **2.** an appendix or supplement to a book, etc. **3.** the part of a gear tooth that projects beyond the pitch circle, or the distance that it projects

Webster's New World Dictionary, Second College Edition.

memoir, *n.* [Fr. *mémoire,* masc., a memorandum, memoir, fem., memory < L. *memoria,* MEMORY] **1.** a biography or biographical notice, usually written by a relative or personal friend of the subject **2.** [*pl.*] an autobiography, usually a full or highly personal account **3.** [*pl.*] a report or record of important events based on the writer's personal observation, special knowledge, etc. **4.** a report or record of a scholarly investigation, scientific study, etc. **5.** [*pl.*] the record of the proceedings of a learned society

Webster's New World Dictionary, Second College Edition.

Short contents

Contents

Introduction

At the request of users I keep extending the memoir class. The *User Manual* has some 250 or so pages and it is a burden to the author to keep changing it and also for the readers to keep getting new copies, especially when a change can be as small as a sentence or paragraph. Hence I trust that this addendum will suffice until there is enough material to warrant a new edition of the manual.

This addendum applies to the fifth edition of the *User Manual* which describes version 1.2 of the memoir class. The class is currently at version 1.3a with patch version 1.8 or later.

The main extensions and changes to the class and manual include:

- There is more flexibility in typesetting the titles of unnumbered chapters;
- Major extensions for typesetting footnotes;
- Major extensions for indexing, including one column and multiple indexes;
- Major extensions to cropmarks;
- Ability to use `\tableofcontents` and friends multiple times;
- Sheet numbers in addition to page numbers, plus access to the numbers of the last sheet and last page;
- Various methods for formatting numbers;
- Better cooperation with the chapterbib and natbib packages when they use their sectionbib option;
- Sectioning commands can take a second optional argument for header text;
- Section titles, as well as numbers, may be referenced;
- Extra 'need space' macros;
- New macros for 'slashed' fractions (fractions like $^{6}/_{29}$);
- Sequential footnote markers;
- Extensions to framed boxes;
- Odd page checking extended to apply to non-arabic numbered pages;
- Means of setting 'optimum' textwidth;
- More intuitive effects of `\mainmatter` and `\backmatter` when the article option is used;
- Control of the spacing of items in the bibliography;
- A 'fixed' version of `\marginpar`;
- As usual, minor glitches have been removed from the code.

One

Corrections

Ezequiel Martín Cámara[1] has corrected me on a historical note. In an email dated 20 January 2003 he said:

> On page 19 you assert that 'Magallanes set sail from Portugal', which is not correct. Magallanes was Portugese, but like Columbus he set sail from Spain, from Sanlúcar de Barrameda, near Cádiz.

Changing to another topic, in the chapter 'Laying out the page' all references to the length `\edgemargin` should be changed to `\foremargin`.[2] These occur on page 59 and in Table 6.7.

Any other occurrences of `\edgemargin` should also be changed to `\foremargin`.

Bastiaan Veelo has kindly provided an updated version of his `\draftnote` example of the use of the `\foremargin` macro. On 7 April 2003 he said in part:

> ...I discovered that `\draftnote` in its present definition does not behave well on a verso page when `\trimedge > 0`. Instead of running into the trimmed area, it runs into the main text, which means that `\marginparsep` is violated. I am not sure whether this should be considered a bug in memoir or LaTeX, but the following hacks around this anyway.

On 19 June 2003 he added '...the implementation has become a little uglier due to a workaround for strange behaviour on verso pages.'

```
%% A new command that allows you to note down ideas or annotations in
%% the margin of the draft. If you are printing on a stock that is wider
%% than the final page width, we will go to some length to utilise the
%% paper that would otherwise be trimmed away, assuming you will not be
%% trimming the draft. These notes will not be printed when we are not
%% in draft mode.
%% Improved spacing by Peter Wilson.
\makeatletter
   \ifdraftdoc
     \newlength{\draftnotewidth}
     \newlength{\draftnotesignwidth}
```

[1] ezequielmartin@yahoo.com

[2] Discovered by Bastiaan Veelo (Bastiaan.N.Veelo@immtek.ntnu.no).

Table 6.1: Length of CMR lowercase alphabets (corrected)

Font size	Length (points)
5 pt	87
6 pt	94
7 pt	102
8 pt	108
9 pt	118
10 pt	128
11 pt	139
12 pt	150
14 pt	175
17 pt	207
20 pt	245
25 pt	290

```
    \newcommand{\draftnote}[1]{\@bsphack%
      {%% do not interfere with settings for other marginal notes
        \strictpagechecktrue%
        \checkoddpage%
        \setlength{\draftnotewidth}{\foremargin}%
        \addtolength{\draftnotewidth}{\trimedge}%
        \addtolength{\draftnotewidth}{-3\marginparsep}%
        \ifoddpage
          \setlength{\marginparwidth}{\draftnotewidth}%
          \marginpar{\flushleft\textbf{\textit{\HUGE !\ }}\small #1}%
        \else
          \settowidth{\draftnotesignwidth}{\textbf{\textit{\HUGE\ !}}}%
          \addtolength{\draftnotewidth}{-\draftnotesignwidth}%
          \marginpar{\raggedleft\makebox[0pt][r]{%% hack around
              \parbox[t]{\draftnotewidth}{%%%%%%%%% funny behaviour
                \raggedleft\small\hspace{0pt}#1%
              }}\textbf{\textit{\HUGE\ !}}%
          }%
        \fi
      }\@esphack}
  \else
    \newcommand{\draftnote}[1]{\@bsphack\@esphack}
  \fi
\makeatother
```

Morten Høgholm[3] has pointed out some serious errors in Table 6.1 on page 54 of the manual. I don't quite know what I was thinking of but I may have done some poor conversions between picas and points. The corrected version of the table is given above.

[3](moho01ab@student.cbs.dk) Private communication, 18 May 2003.

`\abovelegendskip` `\belowlegendskip`
`\abovecaptionskip` `\belowcaptionskip`

The manual erroneously[4] states that the lengths `\abovelegendskip` and `\belowlegendskip` control the spaces above and below a `\legend`. There are no such lengths; the `\abovecaptionskip` and `\belowcaptionskip` control the spaces around both legends and captions.

[4] Reported by Stefan Kahrs on 203/09/01.

Two

Laying out the page

2.1 pdfLaTeX

When pdfLaTeX is used to generate a PDF version of a memoir document some special setup must be done. This is done automatically by the hyperref package but problems have arisen when the package is not used. These have now been resolved.

```
\fixpdflayout
```

The macro `\fixpdflayout`, which is automatically called after the preamble, fixes the layout when pdfLaTeX is used to generate PDF. The default definition is effectively:

```
\newcommand{\fixpdflayout}{\ifpdf\ifnum\pdfoutput>0\relax
  \pdfpageheight=\the\stockheight
  \pdfpagewidth=\the\stockwidth
  \ifdim\pdfvorigin=0pt\pdfvorigin=1in\fi
  \ifdim\pdfhorigin=0pt\pdfhorigin=1in\fi
  \fi\fi}
```

The first settings (`\pdfpage...`) ensure that pdfLaTeX knows the size of the physical sheet for printing. The `\...origin` settings set the pdf origin per the TeX origin, provided that their values are 0pt. If you have set the origin values yourself, either in a pdfLaTeX configuration file or earlier in the preamble, the `\fixpdflayout` macro will not alter them (if you need an origin value to be 0, then set it to 0sp, which is visually indistinguishable from 0pt).

2.2 Font dependency

As well as correcting the original Table 6.1, Morten Høgholm did some curve fitting to the data in Table 2.2 on page 25, which shows the average number of characters per line. He found that the expressions

$$L_{65} = 2.042\alpha + 33.41\text{pt}$$

and

$$L_{45} = 1.415\alpha + 23.03\text{pt}$$

fitted aspects of the data, where α is the length of the alphabet and L_i is the desired linewidth when the line should contain i characters. He suggested the following macros.

```
\setlxvchars[⟨fontspec⟩]
\setxlvchars[⟨fontspec⟩]
```

The macros `\setlxvchars` and `\setxlvchars` set the lengths `\lxvchars` and `\xlvchars` respectively for the font ⟨*fontspec*⟩. The default for ⟨*fontspec*⟩ is `\normalfont`.

For example, the values of `\lxvchars` and `\xlvchars` after calling

```
\setlxvchars \setxlvchars[\small\sffamily]
```

are: `\lxvchars` = 305.42249pt, and `\xlvchars` = 193.65605pt.

Continuing on this theme, Morten also wrote:

> . . . I was defining some environments that had to have `\parindent` as their indentation. For some reason I just wrote `1.5em` instead of `\parindent` because I knew that was the value. What I had overlooked was that I had loaded the mathpazo package, thus altering various `\fontdimen`s. Conclusion: the environment would insert 1.5 em = 18.0 pt, whereas the `\parindent` was only 17.6207 pt.
>
> This, and other related situations can be avoided if one places
>
> `\RequirePackage{`⟨*font-package*⟩`}\normalfont`
>
> before `\documentclass`, but I have to this day never seen this suggested. I would believe that most document classes have settings that depend on *current* font settings, which they should do for such things as `\parindent`. However the decision to let Computer Modern be the default font in LaTeX causes these dimensions to be set to erroneous values. . .

Three

Document divisions

3.1 Introduction

This chapter describes the changes and extensions to the commands for producing section heads.

3.2 Chapter headings

There is one new parameter for controlling the layout of `\chapter` and `\chapter*` titles. In the standard classes the title of an unnumbered chapter is typeset at the same position on the page as the word 'Chapter' for numbered chapters.

`\printchapternonum`

The macro `\printchapternonum` is called just before an unnumbered chapter title text is typeset. By default this does nothing but you can use `\renewcommand` to change this. For example, if you wished the title text for both numbered and unnumbered chapters to be at the same height on the page then you could redefine `\printchapternonum` to insert the amount of vertical space taken by any 'Chapter N' line.

`\chapterprecishere{`⟨*text*⟩`}`
`\prechapterprecis`
`\postchapterprecis`

The `\chapterprecishere` macro is intended for use immediately after a `\chapter`. The ⟨*text*⟩ argument is typeset in italics in a `quote` environment. The macro is not new (it is part of the `\chapterprecis` macro) but its definition has been changed to:

```
\newcommand{\chapterprecishere}[1]{%
  \prechapterprecis #1\postchapterprecis}
```

where `\prechapterprecis` and `\postchapterprecis` are defined as:

```
\newcommand{\prechapterprecis}{%
  \vspace*{-2\baselineskip}%
```

```
    \begin{quote}\normalfont\itshape}
  \newcommand{\postchapterprecis}{\end{quote}}
```

The `\prechapterprecis` and `\postchapterprecis` macros can be changed if another style of typesetting is required.

3.3 Lower level headings

Like the `\chapter` command the lower level headings — `\section` and below — can take two optional arguments. This extension was introduced as one result of the CTT thread *Long headers* which started on 15 January 2003.

```
\section[⟨toc-title⟩][⟨head-title⟩]{⟨title⟩}
\section*{⟨title⟩}
```

The lower level division headings use the ⟨*title*⟩ argument as the division title. Division titles are also available for the ToC and page headings, as follows:

- No optional argument: ⟨*title*⟩ is used for the division title, the ToC title and a page header title.
- One optional argument: ⟨*title*⟩ is used for the division title; ⟨*toc-title*⟩ is used for the ToC title and a page header title.
- Two optional arguments: ⟨*title*⟩ is used for the division title; ⟨*toc-title*⟩ is used for the ToC title; ⟨*head-title*⟩ is used for a page header title.

NOTE: Because of the second optional argument, if you use the hyperref or the nameref packages you will also have to use the memhfixc package (see Chapter 10)

3.4 Moved headings

In the standard classes a `\section` or other divisional heading that is too close to the bottom of a page is moved to the top of the following page. If this happens and `\flushbottom` is in effect, the contents of the short page are stretched to make the last line flush with the bottom of the typeblock.

```
\raggedbottomsectiontrue
\raggedbottomsectionfalse
\bottomsectionskip
```

The `\raggedbottomsectiontrue` declaration will typeset any pages that are short because of a moved divisional header as though `\raggedbottom` was in effect for the short page; other pages are not affected. The length `\bottomsectionskip` controls the amount of stretch on the short page. Setting it to zero allows the last line to be flush with the bottom of the typeblock. The default setting of 10mm appears to remove any stretch.

The declaration `\raggedbottomsectionfalse`, which is the default, cancels any previous `\raggedbottomsectiontrue` declaration.

* * *

The `\plainfancybreak` macro inserts a plain break in the middle of a page or if the break would come at the bottom or top of a page it inserts a fancy break instead.

```
\pfbreak \pfbreak*
\pfbreakskip
\pfbreakdisplay{⟨text⟩}
```

The \pfbreak macro is an alternate for \plainfancybreak that may be more convenient to use. The gap for the plain break is given by the length \pfbreakskip which is initialised to produce two blank lines. The fancy break, which takes the same vertical space, is given by the ⟨*text*⟩ argument of \pfbreakdisplay. The default definition typesets three asterisks, as shown a few lines before this.

♣ ♢ ♣

You can change the definition of \pfbreakdisplay for a different style if you wish. The fancy break just before this was produced via:

```
\renewcommand{\pfbreakdisplay}{%
  $\clubsuit$\quad\$\diamondsuit$\quad$\clubsuit$}
\fancybreak{\pfbreakdisplay}
```

I used \fancybreak as I'm not sure where the break will come on the page and the simple \pfbreak macro might just have produced a couple of blank lines instead of the fancy display.

The paragraph following \pfbreak is not indented. If you want it indented use the \pfbreak* starred version.

3.5 The article option

On 15 July 2003, Romano Giannetti[1] posted the following to CTT under the heading *Memoir class: a little comment about "article" option*. I have edited his remarks slightly.

> For the document I needed the article (treat chapter as article) option, which worked almost well. Could I suggest to rephrase the part of the manual about chapter sectional division where it is said that 'chapters *always* start on a new page…'? I mean, cite that it's like that *unless* article option is used.
>
> On the same idea, when in article mode I think that:
>
> - \tableofcontents should not issue a \pagestyle{chapter}
> - \mainmatter and co should respect the article flag (I mean, \mainmatter emits an unconditional \counterwithin for figure and tables, which it is not — at least for me — the expected behaviour).

Taking these words to heart, a chapter *always* starts a new page unless the article option is used.

When the article option is in effect the \mainmatter command just turns on sectional numbering and starts arabic page numbering; the \backmatter command just turns off sectional numbering.

The \tableofcontents command and friends, as well as any other commands created via \newlistof, *always*[2] call \thispagestyle{chapter}. If you are using the article option you will probably want to ensure that the *chapter* pagestyle is the same as you normally use for the document.

[1] romano@dea.icai.upco.es

[2] This is a consequence of the internal timing of macro calls.

On 2003/07/21 Emanuele Vicentini[3] noted that when using the article option the spacing in the `\chapter` and `\maketitle` commands did not match the spacing when using the article class.

The spacing of `\chapter` when used under the article option now matches that of `\section` in the article class.

In the standard classes the spacing for `\maketitle` depends on whether or not the titlepage option is used. There is no titlepage option in the memoir class and the `\maketitle` spacing is invariant (but of course you can specify what spacings you wish). The default spacing for `\maketitle` in the class was a compromise between the two spacings provided by the standard classes. Now the spacing when the article option is used matches that for the non-titlepage version for the standard article class. This has been accomplished by calling the following article option code:

```
\ifartopt
  \renewcommand{\maketitlehookb}{\vskip -1.5\topsep\vskip -1.5\partopsep}
  \renewcommand{\maketitlehookc}{\vskip -1.5\topsep\vskip -1.5\partopsep}
\fi
```

When the article option is not used these two macros do nothing.

In my view, if you are doing some kind of title page, perhaps using the `titlingpage` environment, it is better to forget about `\maketitle` and design your own layout. Although you can modify `\maketitle` to do what you want, I find it easier to just lay out the title. However, when producing a series of documents with the same title layout it could be more efficient to use a style file that included a modified `\maketitle`.

[3] emanuelevicentini@yahoo.it

Four

Tops and tails

4.1 Introduction

Some small conveniences have been added for the ToC and friends. There are major enhancements for indexing and indexes.

4.2 Table of contents

It is now possible to use `\tableofcontents`, `\listoffigures`, etc., multiple times in a document.

`\settocdepth{`⟨*sec*⟩`}`
`\changetocdepth{`⟨*num*⟩`}`

The `\settocdepth` macro puts the `\changetocdepth` macro into the `toc` file, where ⟨*sec*⟩ is converted into the ⟨*num*⟩ argument. The default definition of `\changetocdepth` is:

```
\DeclareRobustCommand{\changetocdepth}[1]{\setcounter{tocdepth}{#1}}
```

so the `tocdepth` is reset within the ToC itself.

`\chapterprecistoc{`⟨*text*⟩`}`
`\precistoctext{`⟨*text*⟩`}`
`\precistocfont`

The `\chapterprecistoc` macro puts the macro `\precistoctext` into the `toc` file. The default definition is

```
\DeclareRobustCommand{\precistoctext}[1]{%
  {\leftskip \cftchapterindent\relax
   \advance\leftskip \cftchapternumwidth\relax
   \rightskip \@tocrmarg\relax
   \precistocfont #1\par}}
```

Effectively, in the ToC `\precistoctext` typesets its argument like a chapter title using the `\precistocfont` (default `\itshape`).

If you are setting both a short and a long ToC, for the short ToC you may wish to temporarily make `\changetocdepth` and `\precistoctext` swallow their arguments without doing anything else.

`\partnumberline{`⟨*num*⟩`}`
`\chapternumberline{`⟨*num*⟩`}`

In the ToC, the macros `\partnumberline` and `\chapternumberline` are responsible respectively for typesetting the `\part` and `\chapter` numbers. If you do not want, say, the `\chapter` number to appear you can do:

```
\renewcommand{\chapternumberline}[1]{}
```

NOTE: Because the hyperref package does not understand the `\partnumberline` and `\chapternumberline` commands, if you use the hyperref package you will also have to use the memhfixc package (see Chapter 10).

`\cftchapterbreak`

When `\l@chapter` starts to typeset a `\chapter` entry in the ToC the first thing it does is to call the macro `\cftchapterbreak`. This is defined as:

```
\newcommand{\cftchapterbreak}{\addpenalty{-\@highpenalty}}
```

which encourages a page break before rather than after the entry. As usual, you can change `\cftchapterbreak` to do other things that you feel might be useful.

4.3 Indexing

The indexing commands have been significantly enhanced and include the functionality provided by the makeidx, showidx and index packages; these packages should not be used.

In the standard classes the index is set in two columns.

`\onecolindextrue`
`\onecolindexfalse`

The declaration `\onecolindexfalse`, which is the default, causes any indexes to be set in two columns. The declaration `\onecolindextrue` causes any following indexes to be set in one column. This can be useful if, for example, you need an index of the first lines of poems.

`\makeindex[`⟨*file*⟩`]`
`\printindex[`⟨*file*⟩`]`

The macro `\makeindex`, which must be put in the preamble if it is used, opens an `idx` file, which by default is called `jobname.idx`, where `jobname` is the name of the main LaTeX source file. If the optional ⟨*file*⟩ argument is given then a file called `file.idx` will be opened instead. The macro `\printindex` reads an `ind` file called `jobname.ind`, which

should contain an `theindex` environment and the indexed items. If the optional ⟨*file*⟩ argument is given then the `file.ind` file will be read. The MAKEINDEX program is often used to convert an `idx` file to an `ind` file.

```
\index[⟨file⟩]{⟨item⟩}
\specialindex{⟨file⟩}{⟨counter⟩}{⟨item⟩}
```

The macro `\index` writes its ⟨*item*⟩ argument to an `idx` file. If the optional ⟨*file*⟩ argument is given then it will write to `file.idx` otherwise it writes to `jobname.idx`. The page for the ⟨*item*⟩ is also written to the `idx` file. The `\specialindex` macro writes its ⟨*item*⟩ argument to the `file.idx` and also writes the page number (in parentheses) and the value of the ⟨*counter*⟩. This means that indexing can be with respect to something other than page numbers. However, if the hyperref package is used the special index links will be to pages even though they appear to be with respect to the ⟨*counter*⟩; for example, if figure numbers are used as the index reference the hyperref link will be to the page where the figure appears and not the figure itself.

```
\see{⟨item⟩} \seename
\seealso{⟨items⟩} \alsoname
```

The macro `\see` can be used in an `\index` command to tell the reader to 'see ⟨*item*⟩' instead of printing a page number. Likewise the `\seealso` macro directs the reader to 'see also ⟨*items*⟩'. For example:

```
\index{Alf|see{Alfred}}
\index{Frederick|seealso{Fred, Rick}}
```

The actual values for 'see' and 'see also' are given by the `\seename` and `\alsoname` macros whose default definitions are:

```
\newcommand{\seename}{see}
\newcommand{\alsoname}{see also}
```

```
\reportnoidxfilefalse
\reportnoidxfiletrue
```

Following the declaration `\reportnoidxfilefalse`, which is the default, LaTeX will silently pass over attempts to use an `idx` file which has not been declared via `\makeindex`. After the declaration `\reportnoidxfiletrue` LaTeX will whinge about any attempts to write to an unopened file.

```
\showindexmarktrue
\showindexmarkfalse
```

After the declaration `\showindexmarktrue` (practically) all `\index` and `\specialindex` ⟨*item*⟩ arguments are listed in the margin of the page on which the index command is issued. The default is `\showindexmarkfalse`.

INDEXING AND THE NATBIB PACKAGE

The natbib package will make an index of citations if `\citeindextrue` is put in the preamble after the natbib package is called for.

`\citeindexfile`

The name of the file for the citation index is stored in the macro `\citeindexfile`. This is initially defined as:

```
\newcommand{\citeindexfile}{\jobname}
```

That is, the citation entries will be written to the default `idx` file. This may be not what you want so you can change this, for example to:

```
\renewcommand{\citeindexfile}{names}
```

If you do change `\citeindexfile` then you have to put

```
\makeindex[\citeindex]
```

before

```
\usepackage[...]{natbib}
```

So, there are effectively two choices, either along the lines of

```
\renewcommand{\citeindexfile}{authors} % write to authors.idx
\makeindex[\citeindexfile]
\usepackage{natbib}
\citeindextrue
...
\renewcommand{\indexname}{Index of citations}
\printindex[\citeindexfile]
```

or along the lines of

```
\usepackage{natbib}
\citeindextrue
\makeindex
...
\printindex
```

POPULATING THE IDX FILE

In the standard classes, indexed items are written directly to an `idx` file. With the class, however, the indexed items are written to the `aux` file and then on the next LaTeX run the indexed items in the `aux` file are written to the designated `idx` file.

The disadvantage of this two stage process is that after any change to the indexed items LaTeX has to be run twice to ensure that the change is propagated to the `idx` file. Then, of course, a new `ind` will have to be created and LaTeX run one more time. However, this process is very similar to what you have to do if you are using BibTeX to create a bibliography.

The advantage of the approach is that indexed items from `\include` files that are not processed on a particular run are not lost. The standard direct write to an `idx` file loses any 'non-inluded' indexed items.

4.4 The bibliography

As far as an author is concerned there is no change to the `thebibliography` environment, but internally the definition of the environment has been changed as a result of the discussion in the CTT thread *memoir, natbib, and chapterbib* which started on 4 January 2003. The original definition was:

```
\newenvironment{thebibliography}[1]{%
  \chapter*{\bibname}
  \bibmark
  \ifnobibintoc\else
    \phantomsection
    \addcontentsline{toc}{chapter}{\bibname}
  \fi
  \prebibhook
  \begin{bibitemlist}{#1}}{\end{bibitemlist}\postbibhook}
```

The new definition is:

```
\newenvironment{thebibliography}[1]{%
  \bibsection
  \begin{bibitemlist}{#1}}{\end{bibitemlist}\postbibhook}
```

```
\bibsection
```

The macro `\bibsection` defines the heading for the `thebibliography` environment; that is, everything before the actual list of bibliographic items starts. Its default definition is:

```
\newcommand{\bibsection}{%
  \chapter*{\bibname}
  \bibmark
  \ifnobibintoc\else
    \phantomsection
    \addcontentsline{toc}{chapter}{\bibname}
  \fi
  \prebibhook}
```

To change the style of the heading for the bibliography, redefine `\bibsection`. For example, to have the bibliography typeset as a numbered section instead of a chapter, redefine `\bibsection` as:

```
\renewcommand{\bibsection}{%
  \section{\bibname}
  \prebibhook}
```

If you use the natbib and/or the chapterbib packages with the sectionbib option, then `\bibsection` is changed appropriately to typeset as a numbered section.

The natbib package provides a length, `\bibsep` which can be used to alter the vertical spacing between the entries in the bibliography. Setting `\bibsep` to 0pt removes any extra space between the entries.

The equivalent length provided by the class for changing the space between bibliography entries is `\bibitemsep`, which by default is set to the default value of `\itemsep`.

The bibliography is set as a list, and the spacing between the items is (\bibitemsep + \parsep). To eliminate any extra vertical space do

```
\setlength{\bibitemsep}{-\parsep}
```

A hook, called \biblistextra, is provided that is called at the end of the bibliography list setup. By default it does nothing but it can be used, for example, to set all bibliography entries flushleft by modify the list parameters as shown below.

```
\renewcommand{\biblistextra}{%
  \setlength{\leftmargin}{0pt}%
  \setlength{\itemindent}{\labelwidth}%
  \setlength{\itemindent}{\labelsep}%
}
```

See the manual for explanations of the list parameters.

The jurabib package redefines the thebibliography environment[1], providing its own methods for listing the items. However, the redefinition also eliminates the opportunity to add the Bibliography to the Table of Contents and to have some introductory text. To restore these to the class specification, put the following in your preamble after loading jurabib:

```
\usepackage{jurabib}
\makeatletter
\renewcommand{\bib@heading}{\bibsection}
\makeatother
```

However, thanks to the kindness of Jens Berger, if your version of jurabib is 0.6 or later then the fix is not required.

[1] Reported by Robert (w.m.l@gmx.net) on 2003/08/31 in the CTT thread *jurabib + memoir*, where he also suggested the fix.

Five

Typesetting verse

Verse lines are sometimes indented according to the space taken by the text on the previous line.

`\vinphantom{`⟨*text*⟩`}`

The macro `\vinphantom` can be used at the start of a line of verse to give an indentation as though the line started with ⟨*text*⟩. For example:

> …
> Come away with me.
>
> Impossible!
> …

The above fragment from a poem was typeset by:

```
\begin{verse}
\ldots \\
Come away with me.
\end{verse}
\begin{verse}
\vinphantom{Come away with me.} Impossible! \\
\ldots
\end{verse}
```

`\vinphantom` may also be used in the middle of any line to leave some blank space. For example, compare the two lines below, which are produced by this code:

```
\noindent Come away with me and be my love --- Impossible. \\
          Come away with me \vinphantom{and be my love} --- Impossible.
```

Come away with me and be my love — Impossible.
Come away with me — Impossible.

Six

Trim marks

6.1 Introduction

When the memoir class showtrims option is used, trim marks can be placed on each page, usually to indicate where the stock should be trimmed to obtain the planned page size.

Peter Heslin (p.j.heslin@durham.ac.uk) asked me to extend the simple trim mark provided by the memoir class as it appeared unlikely that the author of the crop package would take account of the class (see the thread titled *Incompatibility of memoir.cls and crop.sty*, October 2002 on CTT for details).

6.2 Marks

Trim marks can be placed at each corner of the (trimmed) page, and also at the middle of each side of the page.

```
\trimXmarks
\trimLmarks
\trimFrame
\trimNone
```

Some predefined trimming styles are provided. After the declaration `\trimXmarks` marks in the shape of a cross are placed at the four corners of the page. The declaration `\trimLmarks` calls for corner marks in the shape of an 'L', in various orientations depending on the particular corner. After `\trimFrame` a frame will be drawn around each page, coinciding with the page boundaries. The declaration `\trimNone` disables all kinds of trim marking.

```
\trimmarks
\tmarktl \tmarktr \tmarkbr \tmarkbl
\tmarktm \tmarkmr \tmarkbm \tmarkml
```

The `\trimmarks` macro is responsible for displaying up to 8 marks. The marks are defined as zero-sized pictures which are placed strategically around the borders of the page.

The command \trimmarks places the pictures \tmarktl, \tmarktr, \tmarkbl, and \tmarkbr at the top left, top right, bottom right and bottom left corners of the page. The pictures \tmarktm, \tmarkmr, \tmarkbm, and \tmarkml are placed at the top middle, middle right, bottom middle and middle left of the edges of the page. All these \tmark.. macros should expand to zero-sized pictures.

For example, to draw short lines marking the half-height of the page, try this:

```
\providecommand{\tmarkml}{}
\renewcommand{\tmarkml}{%
  \begin{picture}(0,0){%
    \unitlength 1mm
    \thinlines
    \put(-2,0){\line(-1,0){10}}
  \end{picture}}}
\providecommand{\tmarkmr}{}
\renewcommand{\tmarkmr}{%
  \begin{picture}(0,0){%
    \unitlength 1mm
    \thinlines
    \put(2,0){\line(1,0){10}}
  \end{picture}}}
```

Thin horizontal lines of length 10mm will be drawn at the middle left and middle right of the page, starting 2mm outside the page boundary.

6.3 Sheet numbering

One purpose of trim marks is to show a printer where the stock should be trimmed. In this application it can be useful to also note the sheet number on each page, where the sheet number is 1 for the first page and increases by 1 for each page thereafter. The sheet number is independent of the page number.

```
\thesheetsequence
```

The macro \thesheetsequence typesets the current sheet sequence number and is analogous to the \thepage macro.

```
lastsheet
lastpage
```

The counter lastsheet holds the number of sheets processed during the *previous* run of LaTeX. Similarly, the counter lastpage holds the number of the last page processed during the previous run. Note that the last page number is not necessarily the same as the last sheet number. For example:

In this document this is sheet 32 of 58 sheets, and page 20 of 46.

The previous sentence was the result of processing the following code

```
\textit{In this document this is
        sheet \thesheetsequence\ of \thelastsheet\ sheets,
```

```
        and page \thepage\ of \thelastpage.}
```

You may wish to use the sheet and/or page numbers as part of some trim marks. The following will note the sheet numbers above the page.

```
\newcommand{\trimseqpage}{%
  \begin{picture}(0,0)
    \unitlength 1mm
    \put(0,2){\makebox(0,0)[b]{Sheet: \thesheetsequence\ of \thelastsheet}}
  \end{picture}}
\let\tmarktm\trimseqpage
```

Seven

Margin and foot notes

The standard classes provide the `\marginpar` command for putting things into the margin. The class supports two extra kinds of side notes. It also provides extended footnoting capabilities.

7.1 Sidebars

It appears that the `\sidebar` command, which was originally noted as being experimental, has been used successfully. The command has been revised slightly so that `itemize` and similar environments do not overflow the sidebar width.

`\sidebarform`

Sidebars are normally narrow so text is set ragged right. More accurately, the text is set according to `\sidebarform` which is defined as:

```
\newcommand{\sidebarform}{\rightskip=\z@ \@plus 2em}
```

which is ragged right but with less raggedness than `\raggedright` would allow. As usual, you can change `\sidebarform`, for example:

```
\renewcommand{\sidebarform}{}            % justified text
```

7.2 Side notes

The vertical position of side notes specified via `\marginpar` is flexible so that adjacent notes are prevented from overlapping.

`\sidepar[`⟨*left*⟩`]{`⟨*right*⟩`}`
`\sideparvshift`

The `\sidepar` macro is similar to `\marginpar` except that it produces side notes that do not float — they may overlap. The same spacing is used for both `\marginpar` and `\sidepar`, namely the lengths `\marginparsep` and `\marginparwidth`. The length `\sideparvshift` can be used to make vertical adjustments to the position of `\sidepar`

notes. By default this is set to a value of -2.08ex which seems to be the magical length that aligns the top of the note with the text line.

By default the ⟨*right*⟩ argument is put in the right margin. When the twoside option is used the ⟨*right*⟩ argument is put into the left margin on the verso (even numbered) pages; however, for these pages the optional ⟨*left*⟩ argument is used instead if it is present. For two column text the relevent argument is put into the 'outer' margin with respect to the column.

```
\sideparswitchtrue \sideparswitchfalse
\reversesidepartrue \reversesideparfalse
\parnopar
```

The default placement can be changed by judicious use of the `\reversidepar*` and `\sideparswitch*` declarations. For two sided documents the default is `\sideparswitchtrue`, which specifies that notes should be put into the outer margins; in one sided documents the default is `\sideparswitchfalse` which specifies that notes should always be put into the right hand margin. Following the declaration `\reversesidepartrue` notes are put into opposite margin than that described above (the default declaration is `\reversesideparfalse`).

When LaTeX is deciding where to place the side notes it checks whether it is on an odd or even page and sometimes TeX doesn't realise that it has just moved onto the next page. Effectively TeX typesets paragraph by paragraph (including any side notes) and at the end of each paragraph sees if there should have been a page break in the middle of the paragraph. If there was it outputs the first part of the paragraph, inserts the page break, and retains the second part of the paragraph, without retypesetting it, for eventual output at the top of the new page. This means that side notes for any given paragraph are in the same margin, either left or right. A side note at the end of a paragraph may then end up in the wrong margin. The macro `\parnopar` forces a new paragraph but without appearing to (the first line in the following paragraph follows immediately after the last element in the prior paragraph with no line break). You can use `\parnopar` to make TeX to do its page break calculation when you want it to, by splitting what appears to be one paragraph into two paragraphs.

7.3 Footnotes

The manual described ways of configuring the appearance of `\thanks` notes but did not cover footnotes. This section remedies that omission.

The `\footnote` macro essentially does three things:

- Typesets a marker at the point where `\footnote` is called;
- Typesets a marker at the bottom of the page on which `\footnote` is called;
- Following the marker at the bottom of the page, typesets the text of the footnote.

```
\@makefnmark
```

The `\footnote` macro calls the kernel command `\@makefnmark` to typeset the footnote marker at the point where `\footnote` is called (the value of the marker is kept in the macro `\@thefnmark`). The default definition typesets the mark as a superscript and is essentially

```
\def\@makefnmark{\hbox{\textsuperscript{\@thefnmark}}}
```
To get, say, the marker to be typeset on the baseline in the normal font and enclosed in brackets —
```
\renewcommand*{\@makefnmark}{ [\@thefnmark]}
```

\footfootmark \footmarkwidth \footmarkstyle{⟨*arg*⟩}

The class macro for typesetting the marker at the foot of the page is \footfootmark. The mark is set in a box of width \footmarkwidth. If this is negative, the mark is outdented into the margin, if zero the mark is flush left, and when positive the mark is indented. The appearance of the mark is contolled by \footmarkstyle. The default specification is
```
\footmarkstyle{\textsuperscript{#1}}
```
where the #1 indicates the position of \@thefnmark in the style. The default results in the mark being set as a superscript. For example, to have the marker set on the baseline and followed by a right parenthesis, do
```
\footmarkstyle{#1) }
```

\footmarksep \footparindent

The mark is typeset in a box of width \footmarkwidth and this is then followed by the text of the footnote. Second and later lines of the text are offset by the length \footmarksep from the end of the box. The first line of a paragraph within a footnote is indented by \footparindent.

\foottextfont

The text in the footnote is typeset using the \foottextfont font. The default is \footnotesize.

Altogether, the class specifies
```
\footmarkstyle{\textsuperscript{#1}}
\setlength{\footmarkwidth}{1.8em}
\setlength{\footmarksep}{-1.8em}
\setlength{\footparindent}{1em}
\newcommand{\foottextfont}{\footnotesize}
```
to replicate the standard footnote layout.

You might like to try the following combinations of \footmarkwidth and \footmarksep to see if you prefer the layout produced by one of them to the standard layout (which is produced by the first pairing in the list below):
```
\setlength{\footmarkwidth}{1.8em}  \setlength{\footmarksep}{-1.8em}
\setlength{\footmarkwidth}{1.8em}  \setlength{\footmarksep}{0em}
\setlength{\footmarkwidth}{0em}    \setlength{\footmarksep}{0em}
\setlength{\footmarkwidth}{-1.8em} \setlength{\footmarksep}{1.8em}
\setlength{\footmarkwidth}{0em} \setlength{\footmarksep}{1.8em} \footmarkstyle{#1)\hfill}
```
Any footnotes after this point will be set according to:
```
\setlength{\footmarkwidth}{-1.0em}
\setlength{\footmarksep}{-\footmarkwidth}
```

```
\footmarkstyle{#1}
```

\multfootsep

In the standard classes if two or more footnotes are applied sequentially[1,2] then the markers in the text are just run together. The class, like the footmisc and ledmac packages, inserts a separator between the marks. In the class the macro \multfootsep is used as the separator. Its default definition is:

```
\newcommand*{\multfootsep}{\textsuperscript{\normalfont,}}
```

\feetabovefloat \feetbelowfloat

In the standard classes, footnotes on a page that has a float at the bottom are typeset before the float. I think that this looks peculiar. Following the \feetbelowfloat declaration footnotes will be typeset at the bottom of the page below any bottom floats; they will also be typeset at the bottom of \raggedbottom pages as opposed to being put just after the bottom line of text. The standard positioning is used following the \feetabovefloat declaration, which is the default.

\plainfootnotes \twocolumnfootnotes \threecolumnfootnotes \paragraphfootnotes

Normally, each footnote starts a new paragraph. The class provides three other styles, making four in all. Following the \twocolumnfootnotes declaration footnotes will be typeset in two columns, and similarly they are typeset in three columns after the \threecolumnfootnotes declaration. Footnotes are run together as a single paragraph after the \paragraphfootnotes declaration. The default style is used after the \plainfootnotes declaration.

The style can be changed at any time but there may be odd effects if the change is made in the middle of a page when there are footnotes before and after the declaration. You may find it interesting to try changing styles in an article type document that uses \maketitle and \thanks, and some footnotes on the page with the title:

```
\title{...\thanks{...}}
\author{...\thanks{...}...}
...
\begin{document}
\paragraphfootnotes
\maketitle
\plainfootnotes
...
```

1 One footnote
2 Immediately followed by another

```
\newfootnoteseries{⟨series⟩}
\plainfootstyle{⟨series⟩}
\twocolumnfootstyle{⟨series⟩}
\threecolumnfootstyle{⟨series⟩}
\paragraphfootstyle{⟨series⟩}
```

The class provides for additional series of footnotes; perhaps the normal footnotes are required, flagged with arabic numerals, and another kind of footnote flagged with roman numerals. A new footnote series is created by the `\newfootseries` macro, where ⟨*series*⟩ is an alphabetic identifier for the series. This is most conveniently a single (upper case) letter, for example `P`.

Calling, say, `\newfootnoteseries{Q}` creates a set of macros equivalent to those for the normal `\footnote` but with the ⟨*series*⟩ appended. These include `\footnoteQ`, `\footnotemarkQ`, `\footnotetextQ` and so on. These are used just like the normal `\footnote` and companions.

By default, a series is set to typeset using the normal style of a paragraph per note. The series' style can be changed by using one of the `\...footstyle` commands.

For example, to have a 'P' (for paragraph) series using roman numerals as markers which, in the main text are superscript with a closing parenthesis and at the foot are on the baseline followed by an endash, and the text is set in italics at the normal footnote size:

```
\newfootnoteseries{P}
\paragraphfootstyle{P}
\renewcommand{\thefootnoteP}{\roman{footnoteP}}
\footmarkstyleP{#1--}
\renewcommand{\@makefnmarkS}{\hbox{\textsuperscript{\@thefnmarkP)}}}
\renewcommand{\foottextfont}{\itshape\footnotesize}
```

This can then be used like:

```
.... this sentence\footnoteP{A 'p' footnote\label{fnp}}
includes footnote~\footrefP{fnp}.
```

The `\newfootnoteseries` macro does not create series versions of the footnote-related length commands, such as `\footmarkwidth` and `\footmarksep`, nor does it create versions of `\footnoterule`.

At the foot of the page footnotes are grouped according to their series; all ordinary footnotes are typeset, then all the first series footnotes (if any), then the second series, and so on. The ordering corresponds to the order of `\newfootnoteseries` commands.

```
\fnsymbol{⟨counter⟩}
\@fnsymbol{⟨num⟩}
```

The `\fnsymbol` macro typesets the representation of the counter ⟨*counter*⟩ like a footnote symbol. Internally it uses the kernel `\@fnsymbol` macro which converts a positive integer ⟨*num*⟩ to a symbol. If you are not fond of the standard ordering of the footnote symbols, this is the macro to change. Its original definition is:

```
\def\@fnsymbol#1{\ensuremath{\ifcase#1\or *\or \dagger\or \ddagger\or
  \mathsection\or \mathparagraph\or \|\or **\or \dagger\dagger
  \or \ddagger\ddagger \else\@ctrerr\fi}}
```

This, as shown by `\@fnsymbol{1}, ...\@fnsymbol{9}` produces the series: ∗, †, ‡, §, ¶, ‖, ∗∗, ††, and ‡‡.

Robert Bringhurst quotes the following as the traditional ordering (at least up to ¶): ∗, †, ‡, §, ‖, ¶, ∗∗, ††, and ‡‡. You can obtain this sequence by redefining `\@fnsymbol` as:

```
\renewcommand{\@fnsymbol}[1]{\ensuremath{%
  \ifcase#1\or *\or \dagger\or \ddagger\or
  \mathsection\or \|\or \mathparagraph\or **\or \dagger\dagger
  \or \ddagger\ddagger \else\@ctrerr\fi}}
```

not forgetting judicious use of `\makeatletter` and `\makeatother` if you do this in the preamble. Other authorities or publishers may prefer other sequences and symbols.

Eight

Number formatting

Several methods are provided for formatting numbers. Two classes of number representations are catered for. A 'numeric number' is typeset using arabic digits and a 'named number' is typeset using words.

The argument to the number formatting macros is a 'number', essentially something that resolves to a series of arabic digits. Typical arguments might be:

- Some digits, e.g., `\ordinal{123}` -> 123rd
- A macro expanding to digits, e.g., `\def\temp{3}\ordinal{\temp}` -> 3rd
- The value of a counter, e.g., `\ordinal{\value{page}}` -> 29th
- The arabic representation of a counter, e.g., `\ordinal{\thepage}` -> 29th
  However, if the representation of a counter is not completely in arabic digits, such as `\thesection` which here prints as 8.0, it will produce odd errors or peculiar results if it is used as the argument. For instance:
  `\ordinal{\thesection}` -> .08th

8.1 Numeric numbers

`\cardinal{`⟨*number*⟩`}`
`\fcardinal{`⟨*number*⟩`}`
`\fnumbersep`

The macro `\fcardinal` prints its ⟨*number*⟩ argument formatted using `\fnumbersep` between each triple of digits. The default definition of `\fnumbersep` is:

```
\newcommand{\fnumbersep}{,}
```

Here are some examples:

`\fcardinal{12}` -> 12
`\fcardinal{1234}` -> 1,234
`\fcardinal{1234567}` -> 1,234,567
`\renewcommand{\fnumbersep}{ }\fcardinal{12345678}` -> 12 345 678

The `\cardinal` macro is like `\fcardinal` except that there is no separation between any of the digits.

```
\ordinal{⟨number⟩}
\fordinal{⟨number⟩}
\ordscript{⟨chars⟩}
```

The \fordinal macro typesets its ⟨*number*⟩ argument as a formatted ordinal, using \fnumbersep as the separator. The macro \ordinal is similar except that there is no separation between any of the digits.

Some examples are:

\fordinal{3} -> 3rd
\fordinal{12341} -> 12,341st
\renewcommand{\ordscript}[1]{^{#1}}\fordinal{2} -> 2nd
\ordinal{1234567} -> 1234567th
This is the \ordinal{\value{chapter}} chapter. -> This is the 8th chapter.

The characters denoting the ordinal (ordination?) are typeset as the argument of \ordscript, whose default definition is:

```
\newcommand{\ordscript}[1]{#1}
```

As the above examples show, this can be changed to give a different appearance.

```
\nthstring \iststring \iindstring \iiirdstring
```

The ordinal characters are the values of the macros \nthstring (default: th) for most ordinals, \iststring (default: st) for ordinals ending in 1 like 21st, \iindstring (default: nd) for ordinals like 22nd, and \iiirdstring (default: rd) for ordinals like 23rd.

8.2 Named numbers

The original memoir class provided limited facilities for typesetting named numbers; these have since been extended.

```
\numtoname{⟨number⟩}
\numtoName{⟨number⟩}
\NumToName{⟨number⟩}
```

The macro \numtoname typesets its ⟨*number*⟩ argument using lowercase words. The other two macros are similar, except that \numtoName uses uppercase for the initial letter of the first word and \NumToName uses uppercase for the initial letters of all the words.

As examples:

\numtoname{12345} -> twelve thousand, three hundred and forty-five
\numtoName{12345} -> Twelve thousand, three hundred and forty-five
\NumToName{12345} -> Twelve Thousand, Three Hundred and Forty-Five
The minimum number in TeX is \numtoname{-2147483647}
(i.e., \fcardinal{-2147483647}) ->
The minimum number in TeX is minus two billion, one hundred and forty-seven million, four hundred and eighty-three thousand, six hundred and forty-seven (i.e., -2,147,483,647)

```
\ordinaltoname{⟨number⟩}
\ordinaltoName{⟨number⟩}
\OrdinalToName{⟨number⟩}
```

These three macros are similar to \numtoname, etc., except that they typeset the argument as a wordy ordinal.

For example:

```
This is the \ordinaltoname{\value{chapter}} chapter. ->
```
This is the eighth chapter.

\namenumberand \namenumbercomma \tensunitsep

By default some punctuation and conjunctions are used in the representation of named numbers. According to both American and English practice, a hyphen should be inserted between a 'tens' name (e.g., fifty) and a following unit name (e.g., two). This is represented by the value of \tensunitsep. English practice, but not American, is to include commas (the value of \namenumbercomma) and conjunctions (the value of \namenumberand) in strategic positions in the typesetting. These macros are initially defined as:

```
\newcommand*{\namenumberand}{ and }
\newcommand*{\namenumbercomma}{, }
\newcommand*{\tensunitsep}{-}
```

The next example shows how to achieve American typesetting.

```
\renewcommand*{\namenumberand}{ }
\renewcommand*{\namenumbercomma}{ }
The maximum number in TeX is \numtoname{2147483647} (i.e., \cardinal{2147483647}). ->
```
The maximum number in TeX is two billion one hundred forty-seven million four hundred eighty-three thousand six hundred forty-seven (i.e., 2147483647).

\minusname \lcminusname \ucminusname

When a named number is negative, \minusname is put before the spelled out number. The definitions of the above three comands are:

```
\newcommand*{\lcminusname}{minus }
\newcommand*{\ucminusname}{Minus }
\let\minusname\lcminusname
```

which means that 'minus' is normally all lowercase. To get 'minus' typeset with an initial uppercase letter simply:

```
\let\minusname\ucminusname
```

Only one version of \namenumberand is supplied as I consider that it is unlikely that 'and' would ever be typeset as 'And'. If the initial uppercase is required, renew the macro when appropriate.

There is a group of macros that hold the names for the numbers. To typeset named numbers in a language other than English these will have to be changed as appropriate. You can find them in the class and patch code. The actual picking and ordering of the names is done by an internal macro called \n@me@number. If the ordering is not appropriate for a particular language, that is the macro to peruse and modify. Be prepared, though, for the default definitions to be changed in a later release in case there is a more efficient way of implementing their functions.

CHAPTER 9

Miscellaneous

9.1 Chapter style

Bastiaan Veelo[1] posted the code for a new chapter style to CTT on 2003/07/22 under the title *[memoir] [contrib] New chapter style*. His code, which I have slightly modified and changed the name to *veelo*, is below. I have also exercised editorial privilege on his comments.

> I thought I'd share a new chapter style to be used with the memoir class The style is tailored for documents that are to be trimmed to a smaller width. When the bound document is bent, black tabs will appear on the fore side at the places where new chapters start as a navigational aid. We are scaling the chapter number, which most DVI viewers will not display accurately.
>
> Bastiaan.

```
\makeatletter
\newlength{\numberheight}
\newlength{\barlength}
\makechapterstyle{veelo}{%
   \setlength{\beforechapskip}{40pt}
   \setlength{\midchapskip}{25pt}
   \setlength{\afterchapskip}{40pt}
   \renewcommand{\chapnamefont}{\normalfont\LARGE\flushright}
   \renewcommand{\chapnumfont}{\normalfont\HUGE}
   \renewcommand{\chaptitlefont}{\normalfont\HUGE\bfseries\flushright}
   \renewcommand{\printchaptername}{%
     \chapnamefont\MakeUppercase{\@chapapp}}
   \renewcommand{\chapternamenum}{}
%   \newlength{\numberheight}
```

1 Bastiaan.N.Veelo@immtek.ntnu.no

```
%   \newlength{\barlength}
    \setlength{\numberheight}{18mm}
    \setlength{\barlength}{\paperwidth}
    \addtolength{\barlength}{-\textwidth}
    \addtolength{\barlength}{-\spinemargin}
    \renewcommand{\printchapternum}{%
      \makebox[0pt][l]{%
        \hspace{.8em}%
        \resizebox{!}{\numberheight}{\chapnumfont \thechapter}%
        \hspace{.8em}%
        \rule{\barlength}{\numberheight}
      }
    }
    \makeoddfoot{plain}{}{}{\thepage}
}
\makeatother
```

The style requires the graphicx package because of the `\resizebox` macro. I have removed the two `\newlength` macros to outside the `\makechapterstyle` code just in case the style is called more than once in a document (otherwise there will be 'command already defined' complaints).

As a demonstration, this chapter uses the *veelo* chapterstyle. The style works best for chapters that start on recto pages.

9.2 Cross referencing

The class provides the normal `\label` and `\ref` macros for numeric cross-referencing. For example, the following code and typeset result

```
Chapter~\ref{chap:mempack} starts on page~\pageref{chap:mempack}.
```

Chapter 10 starts on page 41.

It can be useful to refer to parts of a document by name rather than number, as in

```
The chapter \textit{\titleref{chap:mempack}} describes \ldots
```

The chapter *Memoir and packages* describes ...

There are two packages, nameref and titleref, that let you refer to things by name instead of number.

Name references were added to the class as a consequence of adding a second optional argument to the sectioning commands. I found that this broke the nameref package, and hence the hyperref package as well, so they had to be fixed. The change also broke Donald Arseneau's titleref package, and it turned out that nameref also clobbered titleref. The class also provides titles, like `\poemtitle`, that are not recognised by either of the packages. From my viewpoint the most efficient thing to do was to enable the class itself to provide name referencing.

`\label{`⟨*key*⟩`}` `\ref{`⟨*key*⟩`}` `\pageref{`⟨*key*⟩`}`
`\titleref{`⟨*key*⟩`}`
`\headnamereftrue` `\headnamereffalse`

The macro \titleref is an addition to the usual set of cross referencing commands. Instead of typesetting a number it typesets the title associated with the labelled number. This is, of course, only useful if there is an associated title, such as from a \caption or \section command. As a bad example:

```
Labelling for \verb?\titleref? may be applied to:
\begin{enumerate}
\item Chapters, sections, etc.       \label{sec:xref:item1}
...
\item Items in numbered lists, etc. \ldots \label{sec:xref:item3}
\end{enumerate}
Item \ref{sec:xref:item2} in section~\ref{sec:xref} mentions captions
while item \titleref{sec:xref:item3} in the same section
\textit{\titleref{sec:xref}} lists other things.
```

Labelling for \titleref may be applied to:

1. Chapters, sections, etc.
2. Captions
3. Legends
4. Poem titles
5. Items in numbered lists, etc.

Item 2 in section 9.2 mentions captions while item Cross referencing in the same section *Cross referencing* lists other things.

As the above example shows, you have to be a little careful in using \titleref. Generally speaking, \titleref{⟨*key*⟩} produces the last named thing before the \label that defines the ⟨*key*⟩.

Chapters, and the lower level sectional divisions, may have three different title texts — the main title, the title in the ToC, and a third in the page header. By default (\headnamereffalse) the ToC title is produced by \titleref. Following the declaration \headnamereftrue the text intended for page headers will be produced.

NOTE: Specifically with the memoir class, do not put a \label command inside an argument to a \chapter or \section or \caption, etc., command. Most likely it will either be ignored or referencing it will produce incorrect values. This restriction does not apply to the standard classes, but in any case I think it is good practice not to embed any \label commands.

\currenttitle

If you just want to refer to the current title you can do so with \currenttitle. This acts as though there had been a label associated with the title and then \titleref had been used to refer to that label. For example:

```
This sentence in the section titled ‘\currenttitle’ is an example of the
use of the command \verb?\currenttitle?.
```

This sentence in the section titled ‘Cross referencing’ is an example of the use of the command \currenttitle.

\theTitleReference{⟨*num*⟩}{⟨*text*⟩}

Both \titleref and \currenttitle use the \theTitleReference to typeset the title. This is called with two arguments — the number, ⟨*num*⟩, and the text, ⟨*text*⟩, of the title. The default definition is:

```
\newcommand{\theTitleReference}[2]{#2}
```

so that only the ⟨*text*⟩ argument is printed. You could, for example, change the definition to

```
\renewcommand{\theTitleReference}[2]{#1\space \textit{#2}}
```

to print the number followed by the title in italics. If you do this, only use \titleref for numbered titles, as a printed number for an unnumbered title (a) makes no sense, and (b) will in any case be incorrect.

The commands \titleref, \theTitleReference and \currenttitle are direct equivalents of those in the titleref package.

\namerefon \namerefoff

Implementing name referencing has had an unfortunate side effect of turning some arguments into moving ones; the argument to the \legend command is one example. If you don't need name referencing you can turn it off by the \namerefoff declaration; the \namerefon declaration enables name referencing.

9.3 Needing space

There are two new macros in addition to the original \needspace for reserving space at the bottom of a page. The \needspace macro depends on penalties for deciding what to do which means that the reserved space is an approximation. However, except for the odd occasion, the macro gives adequate results.

\Needspace{⟨*length*⟩}
\Needspace*{⟨*length*⟩}

Like \needspace, the \Needspace macro checks if there is ⟨*length*⟩ space at the bottom of the current page and if there is not it starts a new page. The command should only be used between paragraphs; indeed, the first thing it does is to call \par. The \Needspace command checks for the actual space left on the page and is more exacting than \needspace.

If either \needspace or \Needspace produce a short page it will be ragged bottom even if \flushbottom is in effect. With the starred \Needspace* version, short pages will be flush bottom if \flushbottom is in effect and will be ragged bottom when \raggedbottom is in effect.

Generally speaking, use \needspace in preference to \Needspace unless it gives a bad break or the pages must be flush bottom.

9.4 Minor space adjustment

The kernel provides the \, macro for inserting a thin space in both text and math mode. There are other space adjustment commands, such as \! for negative thin space, and \: and \; for medium and thick spaces, which can only be used in math mode.

`\thinspace` `\medspace` `\:` `\!`

On occasions I have found it useful to be able to tweak spaces in text by some fixed amount, just as in math mode. The kernel macro `\thinspace` specifies a thin space, which is 3/18 em. The class `\medspace` specifies a medium space of 4/18 em. As mentioned, the kernel macro `\:` inserts a medium space in math mode. The class version can be used in both math and text mode to insert a medium space. Similarly, the class version of `\!` can be used to insert a negative thin space in both text and math mode.

The math thick space is 5/18 em. To specify this amount of space in text mode you can combine spacing commands as:

```
\:\:\!
```

which will result in an overall space of 5/18 em (from $(4 + 4 - 3)/18$).

9.5 Fractions

When typesetting a simple fraction in text there is usually a choice of two styles, like 3/4 or $\frac{3}{4}$, which do not necessarily look as though they fit in with their surroundings. These fractions were typeset via:

```
... like 3/4 or $\frac{3}{4}$ ...
```

`\slashfrac{`⟨*top*⟩`}{`⟨*bottom*⟩`}`
`\slashfracstyle{`⟨*num*⟩`}`

The class provides the `\slashfrac` command which typesets its arguments like 3/4. Unlike the `\frac` command which can only be used in math mode, the `\slashfrac` command can be used in text and math modes.

The `\slashfrac` macro calls the `\slashfracstyle` command to reduce the size of the numbers in the fraction. You can also use `\slashfracstyle` by itself.

```
In summary, fractions can be typeset like 3/4 or $\frac{3}{4}$,
or \slashfrac{3}{4} or \slashfracstyle{3/4} because several choices
are provided.
```

In summary, fractions can be typeset like 3/4 or $\frac{3}{4}$ or 3/4 or 3/4 because several choices are provided.

`\textsuperscript{`⟨*super*⟩`}`
`\textsubscript{`⟨*sub*⟩`}`

While on the subject of moving numbers up and down, the kernel provides the `\textsuperscript` macro for typesetting its argument as though it is a superscript. The class also provides the `\textsubscript` macro for typesetting its argument like a subscript.

```
You can typeset superscripts like \textsuperscript{3}/4 and
subscripts like 3/\textsubscript{4},
or both like \textsuperscript{3}/\textsubscript{4}.
```

You can typeset superscripts like $^{3}/4$ and subscripts like $3/_{4}$, or both like $^{3}/_{4}$.

9.6 Framed boxes

Donald Arseneau's framed package is currently at or beyond v0.7 while the original copy used in the class is from an earlier version. The class version of the framed functions has been updated to v0.7.

\begin{framed} text \end{framed} \begin{shaded} text \end{shaded} \begin{leftbar} text \end{leftbar}

The framed environment puts the text into an \fbox-like framed box, the shaded environment puts the text into a coloured box, and the leftbar environment draws a vertical line at the left of the text. In all cases the text and boxes can include page breaks.

\FrameRule \FrameSep \FrameHeightAdjust shadecolor

The thickness of the rules is the length \FrameRule and the separation between the text and the box is given by the length \FrameSep. The height of the frame above the baseline at the top of a page is specified by the macro \FrameHeightAdjust. The default definitions being:

```
\setlength{\FrameRule}{\fboxrule}
\setlength{\FrameSep}{3\fboxsep}
\newcommand{\FrameHeightAdjust}{0.6em}
```

The background color in the shaded environment is specified by shadecolor which you have to specify using the color package. For example:

```
\usepackage{color}
\definecolor{shadecolor}{gray}{0.75}
```

\frameasnormaltrue \frameasnormalfalse

Following the declaration \frameasnormaltrue paragraphing within the environments will be as specified for the main text. After the declaration \frameasnormalfalse paragraphing will be as though the environments were like a minipage. The initial declaration is \frameasnormaltrue.

There is one known problem with framing: when framing is used on a page where the page header is in a smaller type than the body, the header may be moved slightly up or down. This can be avoided by putting the font size change in a group, but it seems that a larger font does not need to be grouped. For example:

```
\makeoddhead{myheadings}{{\tiny Tiny header}}{}{\LARGE Large header}
```

You can use the framed package with the memoir class, in which case you will get whatever functionality is provided by the package as it will override the class' code.

9.7 Captions

The \captionstyle macro has been extended so that it is now possible to separately specify the style for short and long captions.

`\captionstyle[`⟨*short*⟩`]{`⟨*normal*⟩`}`
`\raggedleft` `\centering` `\raggedright` `centerlastline`

Caption styles are set according to the `\captionstyle` declaration. Unless the optional ⟨*short*⟩ argument is given all captions are typeset according to ⟨*normal*⟩. If the optional ⟨*short*⟩ argument is specififed, captions that are less than one line in length are typeset according to ⟨*short*⟩.

Permissable values for the arguments include, but are not limited to, `\raggedleft`, `\centering`, `\raggedright`, and `centerlastline`. The class initially specifies

```
\captionstyle{}
```

which gives the normal block paragraph style.

Ten

Memoir and packages

The memoir class does some things differently from the standard classes. Some packages that might be used with memoir rely on the standard methods, and change them to suit their own purposes. Some such changes may not work with memoir and the package may not recognize that it is being used with memoir and not with a standard class.

From my viewpoint, the ideal solution is for the packages to be changed so that they cooperate with memoir. However, until that happy day arrives I have provided the memhfixc package that attempts to make these packages cooperate with the class.

Currently, if you use either the hyperref or the nameref package you will also need to use the memhfixc package. The ordering of the memhfixc and other packages can be important:

- memhfixc must be used *after* the hyperref package.
- The ordering of memhfixc and nameref is immaterial.

There is a basic incompatability between the hyperref package and sequential footnotes — the `\multfootsep` macro is essentially ignored. If that is important to you and you don't mind not having hyperreferences to footnotes, call hyperref like:

```
\usepackage[hyperfootnotes=false, ...]{hyperref}
```

A similar hyperref incompatibily also occurs with at least the footmisc package.

Thinking of the footmisc package, for some of its options it changes the kernel output routine. The class itself changes the output routine in order to add sidebars. Packages, like the version of footmisc current at the date of writing (2003/04/28), which change the output routine without specifically catering for the memoir class are likely to cause problems. As an alternative to the footmisc package, the ledmac package understands memoir and provides further multiple classes of footnotes.

Index

The first page number is usually, but not always, the primary reference to the indexed topic.