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 18 17 16 15 14

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
Eleventh impression, with minor additions	November 2003
Second edition:	September 2005
Twelth impression, with minor additions	October 2005
Thirteenth impression, with minor additions	November 2005
Fourteenth impression, with minor additions	January 2006

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

List of Tables

Introduction to Edition 1

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 1.9 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 $^{6}/_{29}$);
- 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`;
- Extensions for typesetting arrays and tabulars, including continuous tabulars and automatic tabulation;
- As usual, minor glitches have been removed from the code.

Introduction to Edition 2

January 2004 saw a major extension of the memoir class to version 1.6; all the then current patches were folded into the class itself. Edition 6 of the User Manual was released which included the contents of the November 2003 Addendum. Thus, at that time there was no need for either a patch file or the Addendum.

Since then new patches have been made for memoir, fixing problems but adding no new commands. However, in September 2005 a new version, v1.618, of memoir was released which did include some new functions, but not significant enough to warrant a new edition of the User Manual. Edition 2 of the Addendum applies to Edition 6 of the User Manual and covers memoir v1.618 extensions. Further impressions may be released if there are later extensions.

The more major changes or extensions noted in this edition include:

- New part-like pages
- Improved control over higher level ToC entries
- New macros for typesetting the titles of poems
- New macros for making and typsetting glossaries
- Minor extensions for sidebars, boxed verbatims and verses
- The `\em` command is no longer deprecated and minor extension to `\emph`
- Side captions
- New 'book' document division
- Minor extension to numbering verse lines

The following extensions are more for package writers than general authors:

- New macros for specifying emulated packages
- New macro for extending an existing macro
- Hooks into sectioning, captioning, etc., commands

Acknowledgements

The Acknowledgements list in the User Manual is somewhat out of date. I am very grateful to the following who have improved memoir by noting or fixing errors, or providing ideas, suggestions or code: Paul Abrahams, William Adams, Donald Arseneau, Stephan von Bechtolsheim, Jens Berger, Karl Berry, Javier Bezos, Sven Bovin, Ignasi Furió Caldenty, Ezequiel Martín Camara, David Carlisle, Gustavo Cevolani, Jean-Côme Charpentier, Michael A. Cleverly, Steven Douglas Cochran, Frederic Connes, Žarko F. Čučej, Michael W. Daniels, Michael Downes, Thomas Dye, Victor Eijkhout, Danie Els, Robin Fairbairns, Simon Fear, Kai von Fintel, Matthew Ford, Musa Furber, Daniel Richard G., Ignacio Fernández Galván, Romano Giannetti, Donald Goodman, Gabriel Guernik, Kathryn Hargreaves, Sven Hartrumpf, hazydirk, Carsten Heinz, Florence Henry, Peter Heslin, Lars Hoemke, Urs Hofer, Morton Høgholm, Hendrik Holm, Vladimir Ivanovic, Martin Jørgensen, Stefan Kahrs, Marcus Kohm, Jørgen Larsen, Kevin Lin, Matthew Lovell, Daniel Luecking, Lars Madsen, Vittorio De Martino, Frank Mittelbach, Vilar Camara Neto, Rolf Niepraschk, Patrik Nyman, Heiko Oberdiek, Scott Pakin, Adriano Pascoletti, Paul, Steve Peter, Erik Quaeghebeur, Aaron Rendahl, Chris Rowley, Bernd Raichle, Robert Schlicht, Dirk Schlimm, Arnaud Schmittbuhl, Rainer Schöpf, Paul Stanley, James Szinger, Jens Taprogge, Reuben Thomas, Bastiaan Niels Veelo, Emanuele Vicentini, Jürgen Vollmer, and possibly others. If I have inadvertently left anyone out please let me know[1] and I will make the correction.

Along those lines, if you have any questions please direct them to the `comp.text.tex` newsgroup instead of directly to me as you are more likely to get a satisfactory and timely response.

[1] I am currently reachable via email at `herries dot press at earthlink dot net`.

One

Corrections and omissions

1.1 Pagination

In section 1.2 of the manual I say that pagination usually starts with the Table of Contents page. This is wrong! *All* pages take part in the pagination, including the Half-title and Title pages. It is the folios (i.e., the printed page numbers) that usually start on the Table of Contents page.

1.2 Writing out verbatim

The class provides two methods for writing out verbatim material to a file. Unfortunately only one was described in the manual. Here are the two methods.

```
\begin{verbatimoutput}{⟨file⟩} material \end{verbatimoutput}
\begin{writeverbatim}{⟨stream⟩} material \end{writeverbatim}
```

The material in these environments is written out verbatim. Material from the `verbatimoutput` environment is written to the ⟨*file*⟩ file. On the other hand, material from the `writeverbatim` environment is written to whatever file is currently associated with the ⟨*stream*⟩ output stream.

Two

Document divisions

2.1 Book headings

Frederic Connes has told me that in French typography there is often a document division above the `\part` level. This is also sometimes the case with English typography — the *Chicago Manual of Style* [CMS93, p21] shows an example. Based on code that he kindly sent me, a new document division, Book, is provided that is above Part in the division hierarchy. There are now six levels of document division and these are given in Table 2.1.

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

The `\book` command is like the `\part` command and produces a similar looking title page in the document.

```
\bookmark{⟨title⟩}
```

The `\book` code includes `\bookmark{⟨title⟩}` for storing the title of the book if it is going to be used, for example, in page headers. Its default definition is simply:

Table 2.1: Division levels

Division	Level
`\book`	-2
`\part`	-1
`\chapter`	0
`\section`	1
`\subsection`	2
`\subsubsection`	3
`\paragraph`	4
`\subparagraph`	5

```
\newcommand*{\bookmark}[1]{}
```

\bookblankpage \nobookblankpage

A page with a \book title is normally followed by a blank page. If you do not want this then use the \nobookblankpage declaration. The \bookblankpage declaration reverts the behaviour to the normal blank page.

The pagestyle *book* is applied to a book title page and the pagestyle *afterbook*, which defaults to *empty*, is applied to the blank page, if any, following a \book page.

BOOK HEADING LAYOUT

The layout for a \book page is very similar to that for a \part page, although there are separate controls for each.

The code for typesetting the page is in essence like this.

```
\newcommand{\book}[1]{%
  \cleardoublepage
  \thispagestyle{book}%
  \beforebookskip
  \printbookname\booknamenum{\booknumfont\thebook}%
  \midbookskip
  \printparttitle{#1}%
  \bookmark{#1}%
  \afterpartskip}
```

\beforebookskip \midbookskip \afterbookskip

These commands are called before any part of the title is set, between typesetting the number (if any) and the title, and after the title has been set.

\bookname \printbookname \booknamefont \booknamenum

\bookname, defined as \newcommmand*{\bookname}{Book} is the name for a book. \printbookname, prints the \bookname using the \booknamefont, which is defined as:
\newcommand*{\booknamefont}{\normalfont\huge\bfseries}
This is followed by \booknamenum, which is defined to be \space.

\booknumfont \printbooknum

\printbooknum prints the book number (i.e., \thebook) using the \booknumfont, whose default definition is the same as \booknamefont.

\booktitlefont \printbooktitle{⟨*title*⟩}

`\printbooktitle{`⟨*title*⟩`}` prints the book title using the `\booktitlefont`, which is defined as:
`\newcommand*{\booktitlefont}{\normalfont\Huge\bfseries}`

`\bookblankpage` `\nobookblankpage`

Following the default `\bookblankpage` declaration a blank page is output after a book title page. No blank page is produced after the `\nobookblankpage` declaration.

2.2 Part headings

`\partmark{`⟨*title*⟩`}`

The `\part` code now includes `\partmark{`⟨*title*⟩`}` for storing the title of the part if it is going to be used, for example, in page headers. Its default definition is simply:
`\newcommand*{\partmark}[1]{}`

`\partblankpage` `\nopartblankpage`

A page with a `\part` title is normally followed by a blank page. If you do not want this then use the `\nopartblankpage` declaration. The `\partblankpage` declaration reverts the behaviour to the normal blank page.

A new pagestyle *afterpart*, which defaults to *empty*, is applied to the blank page, if any, following a `\part` page.

`\newleadpage[`⟨*page-style*⟩`]{`⟨*cmdname*⟩`}{`⟨*title*⟩`}`
`\newleadpage*[`⟨*page-style*⟩`]{`⟨*cmdname*⟩`}{`⟨*title*⟩`}`
`\renewleadpage[`⟨*page-style*⟩`]{`⟨*cmdname*⟩`}{`⟨*title*⟩`}`
`\renewleadpage*[`⟨*page-style*⟩`]{`⟨*cmdname*⟩`}{`⟨*title*⟩`}`

`\newleadpage` and associates are variants of the `\newcommand` and companions; the original suggestion and code was from Danie Els. The `\newleadpage` macro defines a macro `\cmdname` that when called typeset an Appendixpage-like page with a title ⟨*title*⟩ using the ⟨*page-style*⟩ for the page. The default ⟨*page-style*⟩ is `empty`. The macro `\renewleadpage` redefines an existing leadpage command.

For example,

```
\newleadpage{plates}{Picture Gallery}
```

creates the new command `\plates` which when called generates an unnumbered part-like page with the title **Picture Gallery**.

`\leadpagetoclevel`

When `\(re)newleadpage` is used the resulting command adds ⟨*title*⟩ to the ToC as though it was an unnumbered chapter. More precisely it will be added as though it were an unnumbered `\leadpagetoclevel` entry, whose default definition is:

```
\newcommand*{\leadpagetoclevel}{chapter}
```

When the starred versions `\(re)newleadpage*` are used the resulting command will not add ⟨*title*⟩ to the ToC.

Internally the resulting commands use `\partmark` for storing the ⟨*title*⟩ for possible later use by you.

2.3 Chapter styles

REPARTICLE

When the article option is used the default chapter and section styles are close to, but not identical, the corresponding division headings in the article class.

The *reparticle* chapterstyle makes a `\chapter` replicate the appearance of a `\section` in the article class.

```
\reparticle
```

The `\reparticle` command makes chapter and lower level division heads replicate those of the article class. You can use it, for example, like:

```
\ifartopt
  \reparticle
\fi
```

to call it if the article option has been requested.

SOUTHALL

On 2006/01/08 Thomas Dye posted his *southall* chapterstyle on `comp.text.tex` and kindly gave me permission to include it here. It is based on the headings in a Cambridge Press book[1] by Aidan Southall. It produces a simple numbered heading with the title set as a block paragraph, and with a horizontal rule underneath. His original code called for lining figures for the number but I have commented out that bit.

```
%% Thomas Dye's southall chapter style
\newlength{\headindent}
\newlength{\rightblock}
\makechapterstyle{southall}{%
  \setlength{\headindent}{36pt}
  \setlength{\rightblock}{\textwidth}
  \addtolength{\rightblock}{-\headindent}
  \setlength{\beforechapskip}{2\baselineskip}
  \setlength{\afterchapskip}{5\baselineskip}
  \setlength{\midchapskip}{0pt}
  \renewcommand{\chaptitlefont}{\huge\rmfamily\raggedright}
  \renewcommand{\chapnumfont}{\chaptitlefont}
  \renewcommand{\printchaptername}{}
  \renewcommand{\chapternamenum}{}
```

[1] Which I haven't seen

```
\renewcommand{\afterchapternum}{}
\renewcommand{\printchapternum}{%
  \begin{minipage}[t][\baselineskip][b]{\headindent}
    {\vspace{0pt}\chapnumfont%%%\figureversion{lining}
                 \thechapter}
  \end{minipage}}
\renewcommand{\printchaptertitle}[1]{%
  \hfill\begin{minipage}[t]{\rightblock}
    {\vspace{0pt}\chaptitlefont ##1\par}\end{minipage}}
\renewcommand{\afterchaptertitle}{%
  \par\vspace{\baselineskip}%
  \hrulefill \par\nobreak\noindent \vskip\afterchapskip}}
```

CHAPPELL

Another style that includes rules is one I have called *chappell*, which is based on the chapter heads in [CB99]. The style can easily form the basis for general heads in non-technical books.

```
\makechapterstyle{chappell}{%
  \setlength{\beforechapskip}{0pt}
  \renewcommand*{\chapnamefont}{\large\centering}
  \renewcommand*{\chapnumfont}{\large}
  \renewcommand*{\printchapternonum}{%
    \vphantom{\printchaptername}%
    \vphantom{\chapnumfont 1}%
    \afterchapternum
    \vskip -\onelineskip}
  \renewcommand*{\chaptitlefont}{\Large\itshape}
  \renewcommand*{\printchaptertitle}[1]{%
    \hrule\vskip\onelineskip \centering\chaptitlefont ##1}}
```

This style centers the chapter number, draws a rule acroos the page under it, and below that comes the title, again centered. All the fiddling in the `\printchapternonum` macro is to try and ensure that the rule above the title is at the same height whether or not the chapter is numbered (the ToC being an example of an unnumbered headinggg).

2.4 Poem Titles

The new command `\PoemTitle` provides more flexibility in typesetting titles of poems that the original `poemtitle` does.

```
\PoemTitle[⟨fortoc⟩][⟨forhead⟩]{⟨title⟩}
\NumberPoemTitle
\PlainPoemTitle
\thepoem
\poemtitlemark{⟨forhead⟩}
\poemtitlepstyle
```

The \PoemTitle command takes the same arguments as the \chapter command; it typesets the title for a poem and adds it to the ToC. Following the declaration \NumberPoemTitle the title is numbered but there is no numbering after the \PlainPoemTitle declaration.

The macro \poemtitlemark is called with the argument ⟨*forhead*⟩ so that it may be used to set marks for use in a page header via the normal mark process. The \poemtitlepstyle macro, which by default does nothing, is provided as a hook so that, for example, it can be redefined to specify a particular pagestyle that should be used. For example:

```
\renewcommand*{\poemtitlemark}[1]{\markboth{#1}{#1}}
\renewcommmand*{\poemtitlepstyle}{%
  \pagestyle{headings}%
  \thispagestyle{empty}}
```

\PoemTitle*[⟨*forhead*⟩]{⟨*title*⟩}
\poemtitlestarmark{⟨*forhead*⟩}
\poemtitlestarpstyle

The \PoemTitle* command produces an unnumbered title that is not added to the ToC. Apart from that it operates in the same manner as the unstarred version. The \poemtitlestarmark and \poemtitlestarpstyle can be redefined to set marks and pagestyles.

MAIN POEM TITLE LAYOUT PARAMETERS

\PoemTitleheadstart
\printPoemTitlenonum
\printPoemTitlenum
\afterPoemTitlenum
\printPoemTitletitle{⟨*title*⟩}
\afterPoemTitle

The essential of the code used to typeset a numbered ⟨*title*⟩ from a PoemTitle is:

```
\PoemTitleheadstart
\printPoemTitlenum
\afterPoemTitlenum
\printPoemTitletitle{title}
\afterPoemTitle
```

If the title is unnumbered then \printPoemTitlenonum is used instead of the \printPoemTitlenum and \afterPoemTitlenum pair of macros.

The various elements of this can be modified to change the layout. By default the number is centered above the title, which is also typeset centered, and all in a \large font.

The elements are detailed in the next section.

DETAILED POEM TITLE LAYOUT PARAMETERS

```
\beforePoemTitleskip
\PoemTitlenumfont
\midPoemTitleskip
\PoemTitlefont
\afterPoemTitleskip
```

As defined, \PoemTitleheadstart inserts vertical space before a poem title. The default definition is:

```
\def\PoemTitleheadstart{\vspace{\beforePoemTitleskip}}
\newlength{\beforePoemTitleskip}
  \setlength{\beforePoemTitleskip}{1\onelineskip}
```

\printPoemTitlenum typesets the number for a poem title. The default definition, below, prints the number centered and in a large font.

```
\def\printPoemTitlenum{\PoemTitlenumfont \thepoem}
\newcommand*{\PoemTitlenumfont}{\normalfont\large\centering}
```

The definition of \printPoemTitlenonum, which is used when there is no number, is simply

```
\def\printPoemTitlenonum{}
```

\afterPoemTitlenum is called between setting the number and the title. It ends a paragraph (thus making sure any previous \centering is used) and then may add some vertical space. The default definition is:

```
\def\afterPoemTitlenum{\par\nobreak\vskip \midPoemTitleskip}
\newlength{\midPoemTitleskip}
  \setlength{\midPoemTitleskip}{0pt}
```

The default definition of \printPoemTitletitle is below. It typesets the title centered and in a large font.

```
\def\printPoemTitletitle#1{\PoemTitlefont #1}
\newcommand*{\PoemTitlefont}{\normalfont\large\centering}
```

The macro \afterPoemTitle finishes off the title typesetting. The default definition is:

```
\def\afterPoemTitle{\par\nobreak\vskip \afterPoemTitleskip}
\newlength{\afterPoemTitleskip}
  \setlength{\afterPoemTitleskip}{1\onelineskip}
```

Three

Front and rear

3.1 The ToC

There have been requests for a ToC layout along the following lines:

```
PART I Title of part ....
Chapter 1. Title of chapter ...
  1.1 A section ...
...
Appendix A. Title of appendix
  A.1 Another section
...
```

Danie Els suggested that this could be accomplished with a few new commands.[1]

```
\cftpartname
\cftchaptername
\cftappendixname
```

These commands form part of the code for `\part` and `\chapter` entries in the ToC, coming before the relevant number. Their default definitions are empty. To get the above ToC you can do the following:

```
\renewcommand*{\cftpartname}{PART~}
\renewcommand*{\cftchaptername}{\chaptername~}
\renewcommand*{\cftappendixname}{\appendixname~}
\renewcommand*{\cftchapteraftersnum}{.} % dot after number
\setlength{\cftchapternumwidth}{2em} % allow more space
```

Another query has been how to get the titles in the ToC to be set raggedright instead of the usual flushright. Assuming that there are more than 100 pages in the document:

```
\setrmarg{3.55em plus 1fil}
```

[1] Plus the recoding of some internal macros.

where the last four characters before the closing brace are: digit 1, lowercase F, lowercase I, and lowercase L.

You may have noticed that there are two ToCs for this document — a short one and a long one. This is how they were produced.

The code below was used to produce the short ToC.

```
%% Short contents and Different ToC style
\renewcommand{\contentsname}{Short contents}
\let\oldchangetocdepth\changetocdepth
\let\oldcftchapterfillnum\cftchapterfillnum
\renewcommand{\changetocdepth}[1]{}
\setcounter{tocdepth}{0} % chapters
\renewcommand{\cftchapterfont}{\hfill\sffamily}
\renewcommand{\cftchapterleader}{ \textperiodcentered\space}
\renewcommand{\cftchapterafterpnum}{\cftparfillskip}
\setpnumwidth{0em}
\setrmarg{0.3\textwidth}
\tableofcontents
\clearpage
```

The above sets the page numbers as though they were left adjusted in the page number box but with LaTeX reporting overfull hboxes. Changing the width of the box for the page number

```
\setpnumwidth{1.5em}
```

results in the page numbers being right adjusted in the 1.5em wide box, which I think does not look as good. To get the left adjusted effect with no complaints try:

```
\renewcommand*{\cftchapterfillnum}[1]{%
  {\cftchapterleader}\nobreak
  \hbox to 1.5em{\cftchapterpagefont #1\hfil}\cftchapterafterpnum\par}
```

The next piece of code will typeset a group of subsections in the ToC as a paragraph.

```
%%% have subsections as a paragraph in the ToC
\makeatletter
\let\oldnumberline\numberline
\renewcommand{\cftsubsectionfont}{\itshape}
\renewcommand{\cftsubsectionpagefont}{\itshape}
\renewcommand{\l@subsection}[2]{\relax
  \def\numberline##1{\textit{##1}~}%
  \leftskip=\cftsubsectionindent
  \rightskip=\@tocrmarg
%  \advance\rightskip \z@ plus \hsize % uncomment this for raggedright
%  \advance\rightskip \z@ plus 2em % uncomment this for semi-raggedright
  \parfillskip=\fill
  \ifhmode ,\ \else\noindent\fi
  \ignorespaces {\cftsubsectionfont #1}~{\cftsubsectionpagefont #2}%
  \let\numberline\oldnumberline\ignorespaces
}
\AtEndDocument{\addtocontents{toc}{\par}}
\makeatother
```

The final piece of code below sets up the rest of the second ToC.

```
%% Default contents
\renewcommand{\contentsname}{Contents}
\let\changetocdepth\oldchangetocdepth
\let\cftchapterfillnum\oldcftchapterfillnum
\renewcommand{\cftchapterfont}{\normalfont\sffamily}
\renewcommand{\cftchapterleader}{\sffamily\cftdotfill{\cftchapterdotsep}}
\renewcommand{\cftchapterafterpnum}{}
\makeatletter
\renewcommand{\cftchapterbreak}{\par\addpenalty{-\@highpenalty}}
\makeatother
\setpnumwidth{2.55em}
\setrmarg{3.55em}
\setcounter{tocdepth}{2}
\tableofcontents
```

BOOK TOC ENTRY

There are the usual kinds of commands for configuring the appearance of `\book` entries in the ToC.

`\cftbookname`

This is called before the book number is set in the ToC. The default definition is empty.

`\cftbeforebookskip` `\cftbookindent` `\cftbooknumwidth`

These lengths control the vertical space before a book entry, the indentation of the entry from the left margin, and the space for typesetting the number.

`\cftbookfont` `\cftbookpagefont`

These specify the fonts to be used for typesetting the number and title of a book entry in the ToC, and the page number.

`\cftbookpresnum` `\cftbookaftersnum` `\cftbookaftersnumb`

The book number is typeset in a box. `\cftbookpresnum` and `\cftbookaftersnum` are called within the box before and after the number. `\cftbookaftersnumb` is called after the box has been typeset. By default these commands do nothing.

`\cftbookleader` `\cftbookdotsep`

`\cftbookleader` typesets the leader between a book title and the page number in the ToC. By default this is a dotted leader with `\cftbookdotsep` between the dots. By default this is set for a book entry to produce no dots.

3.2 The index

The internal indexing code has been changed slightly to make things more efficient. Now any changes to indexed items will be immediately reflected in the `idx` file. Previously it took two LaTeX runs to achieve this.

The MakeIndex indexing program can handle a memoir hyperindex, but the xindy program cannot, as indicated by the following extract from an email sent to me by Frederic Connes:

```
... You use "|hyperspindexpage(\thepage)", which xindy doesn't
recognize as a valid markup-locref. And I don't see how to add
it, because xindy only accepts one argument in markup-locref
(if the number is not a page number, it will still point to a
page with that number), so replacing it with "|hyperpage" won't
work.

...It would be nice to be able to deactivate the
"|hyperspindexpage" part, as it causes xindy to crash.
```

I don't use xindy, but I have provided something that addresses the problem.

```
\memhyperindexfalse
```

Putting `\memhyperindexfalse` into the preamble will prevent any hyperindexing no matter what you have instructed the hyperref package to do.

Since then Frederic Connes has provided some code for when xindy will be used.

```
\xindyindex
```

Put the `\xindyindex` declaration in the pramble when you will be using xindy to process the raw index; there is no need to use `\memhyperindexfalse` as well as this.

3.3 Glossaries

Unlike for indexes, LaTeX provides less than minimal support for glossaries. It provides a `\makeglossary` command for initiating a glossary and a `\glossary` command which puts its argument, plus the page number, into a `glo` file, and that's it. memoir, combined with the MakeIndex program [CH88], enables you to generate and print a glossary in your document. The commands for creating a glossary are similar to those for indexes.

```
\makeglossary[⟨file⟩]
```

You have to put `\makeglossary` in your preamble if you want a glossary. This opens a file called by default `\jobname.glo`. If you use the optional ⟨*file*⟩ argument the file `file.glo` will be opened. A glossary `glo` file is analagous to an index `idx` file.

```
\printglossary[⟨file⟩]
```

To print a glossary call `\printglossary` which will print the glossary from file `\jobname.gls`, or from `file.gls` if the optional argument is used. A glossary `gls` file is analagous to an index `ind` file.

`\glossary[`⟨*file*⟩`](`⟨*key*⟩`){`⟨*term*⟩`}{`⟨*desc*⟩`}`

Use the `\glossary` command to add a ⟨*term*⟩ and its description, ⟨*desc*⟩, to a glossary file. By default this will be `\jobname.glo` but if the optional ⟨*file*⟩ argument is given then the information will be written to `file.glo`. The `(`⟨*key*⟩`)` argument is optional. If present then ⟨*key*⟩ will be added to the file to act as a sort key for the ⟨*term*⟩, otherwise ⟨*term*⟩ will be used as the sort key.

By using the optional ⟨*file*⟩ arguments you can have several glossaries, subject to TeX's limitations on the number of files that can be open at any one time.

A simple glossary entry might be:

```
\glossary{glossary}{A list of terms and their descriptions.}
```

The glossary facilites are designed so that the MakeIndex program can be used to convert the raw glossary data in a `glo` file into the printable glossary in a `gls` file.

`\begin{theglossary}` entry list `\end{theglossary}`

Glossary entries are typeset in a `theglossary` environment. It is assumed that a `gls` file will contain a complete `theglossary` environment, from `\begin{theglossary}` all the way through to `\end{theglossary}`.

`\glossitem{`⟨*term*⟩`}{`⟨*desc*⟩`}{`⟨*ref*⟩`}{`⟨*num*⟩`}`

A `\glossitem` is a glossary entry within a `theglossary` environment for a ⟨*term*⟩ with ⟨*description*⟩. The ⟨*num*⟩ argument is the page or section where the corresponding `\glossary` was issued. The ⟨*ref*⟩ argument, if not empty, might be the section or page number corresponding to the ⟨*num*⟩ page or section number. The default definition is

```
\newcommand{\glossitem}[4]{#1 #2 #3 #4}
```

which is not very exciting. You may well prefer to use your own definition.

3.4 Controlling the glossary

SETTING UP MAKEINDEX

If you just run MakeIndex on a `glo` file you will get lots of errors; MakeIndex has to be configured to read a `glo` file and generate a useful `gls` file as by default it expects to read an index `idx` file and produce an index `ind` file. A configuration file like an index `ist` file will be needed. There is no recommended extension for such a file but I have come to favour `gst`. The command line for MakeIndex to create a sorted glossary from the raw data in a `glo` file, say `fred.glo`, using a configuration file called, say `basic.gst`, is

```
makeindex -s basic.gst -o fred.gls fred.glo
```

For other jobs just change the file names appropriately.

So, what is in a `gst` file? The potential contents are given by Chen & Harrison [CH88] and also in the *Companion* [MG+04, Chap. 11]. At a minimum you need this:

Table 3.1: MakeIndex configuration file input parameters

Keyword	Default	Description
`keyword` *(s)*	`"\\indexentry"`	The argument to this command is a MakeIndex index entry
`arg_open` *(c)*	`'{'`	Argument start delimeter
`arg_close` *(c)*	`'}'`	Argument end delimeter
`range_open` *(c)*	`'('`	Start of an explicit page range
`range_close` *(c)*	`')'`	End of an explicit page range
`level` *(c)*	`'!'`	Character denoting a new subitem level
`actual` *(c)*	`'@'`	Character denoting that the following text is to appear in the actual index file
`encap` *(c)*	`'|'`	Character denoting that the rest of the argument is to be used as an encapsulating command for the page number
`quote` *(c)*	`'"'`	Character that escapes the following character
`escape` *(c)*	`'\\'`	Symbol with no special meaning unless followed by the `quote` character, when both characters will be printed. The `quote` and `escape` characters must be different.
`page_compositor` *(s)*	`"-"`	Composite number separator

(s) of type string, *(c)* of type character

```
%%% basic.gst basic makindex glossary style file
%%% Output style parameters
preamble "\\begin{theglossary}"
postamble "\n\\end{theglossary}\n"
item_0    "\n\\glossitem"
delim_0   "{\\memglonum{"
encap_suffix "}}}"
%%% Input style parameters
keyword "\\glossaryentry"
```

The `keyword` line says that each entry in an input (`glo`) file will be of the form:

```
\glossaryentry{entry text}{number}
```

and by a miracle of coding, this is what memoir will put in a `glo` file for each `\glossary` command.

The `preamble` and `postamble` lines tell the program to start and end its output file with `\begin{theglossary}` and `\end{theglossary}`, respectively. The `item_0` tells the program to start each output entry with `\glossitem`. The `delim_0` says that `{\memglonum{` sould be put between the end of the entry text and the (page) number. Finally `encap_suffix` requests `}}}` to be put after any 'encapsulated' (page) number.

A complete listing of the possible entries in a configuration file, also called a style file, for MakeIndex is in Table 3.1 and 3.2 with the exception of the output file page number setting keywords.

Table 3.2: MakeIndex configuration file output parameters

Keyword	Default	Description
`preamble` *(s)*	`"\\begin{theindex}\n"`	Text for the start of the output file
`postamble` *(s)*	`"\n\n\\end{theindex}\n"`	Text at the end of the output file
`group_skip` *(s)*	`"\n\n\\indexspace\n"`	Vertical space before a new letter group
`heading_prefix` *(s)*	`""`	Prefix for heading for a new letter group
`heading_suffix` *(s)*	`""`	Suffix for heading for a new letter group
`headings_flag` *(n)*	`0`	A value $= 0$ inserts nothing between letter groups. A value > 0 includes an uppercase instance of the new symbol, while a value < 0 includes a lowercase instance, all within `heading_prefix` and `heading_suffix`
`item_0` *(s)*	`"\n\item "`	Command inserted in front of a level 0 entry
`item_1` *(s)*	`"\n \subitem "`	As above for a level 1 entry
`item_2` *(s)*	`"\n  \subsubitem "`	As above for a level 2 entry
`item_01` *(s)*	`"\n \subitem "`	Command inserted in front of a level 1 entry starting at level 0
`item_12` *(s)*	`"\n  \subsubitem "`	Command inserted in front of a level 2 entry starting at level 1
`item_x1` *(s)*	`"\n \subitem "`	Command inserted in front of a level 1 entry when the parent level has no page numbers
`item_x2` *(s)*	`"\n \subitem "`	As above for a level 2 entry
`delim_0` *(s)*	`", "`	Delimiter between level 0 entry and first page number
`delim_1` *(s)*	`", "`	As above for level 1 entry
`delim_2` *(s)*	`", "`	As above for level 2 entry
`delim_n` *(s)*	`", "`	Delimiter between page numbers
`delim_r` *(s)*	`"-"`	Designator for a page range
`encap_prefix` *(s)*	`"\\"`	Prefix in front of a page encapsulator
`encap_infix` *(s)*	`"{"`	Infix for a page encapsulator
`encap_suffix` *(s)*	`"}"`	Suffix for a page encapsulator
`page_precedence` *(s)*	`"rnaRA"`	Page number precedence for sorting. `r` and `R` are lower- and uppercase roman; `a` and `A` are lower- and uppercase alphabetic; `n` is numeric
`line_max` *(n)*	`"72"`	Maximum length of an output line
`indent_space` *(s)*	`"\t\t"`	Indentation commands for wrapped lines
`indent_length` *(n)*	`"16"`	Indentation length for wrapped lines

(s) of type string, *(n)* of type number, `"\n"` and `"\t"` are newline and tab.

The gst file I have used for this document has a few more items than the basic one.

```
%%% memman.gst makindex glossary style file for memman and friends
%%% Output style parameters
preamble "\\begin{theglossary}"
postamble "\n\\end{theglossary}\n"
group_skip "\n\\glossaryspace\n"
item_0    "\n\\glossitem"
delim_0   "{\\memglonum{"
encap_suffix "}}}"
indent_space "\t"
indent_length 2
%%% Input style parameters
keyword "\\glossaryentry"
actual '?'
page_compositor "."
```

The group_skip line asks that \glossaryspace be put between the last entry for one letter and the first for the next letter. The indent_space and indent_length give a smaller indent for continuation lines in the output than the default.

The actual entry says that the input file will use ? instead of the default @ as the flag for separating a key from the start of the real entry. The page_compositor indicates that any compound numbers will be like 1.2.3 instead of the default 1-2-3.

RAW INPUT DATA

\@@wrglom@m{⟨*file*⟩}{⟨*key*⟩}{⟨*term*⟩}{⟨*desc*⟩}{⟨*ref*⟩}{⟨*num*⟩}

The \glossary macro writes its arguments to the aux file in the form of arguments to the \@@wrglom@m internal macro. In turn this calls a series of other macros that eventually write the data to the ⟨*file*⟩ glo file in the format (where @ is the actual flag):

```
\glossaryentry{key@{\memgloterm{term}} {\memglodesc{desc}}{\memgloref{ref}}
               |memglonumf}{num}
```

which MakeIndex then effectively converts into

```
\glossitem{\memgloterm{term}}{\memglodesc{desc}}{\memgloref{ref}}
          {\memglonum{\memglonumf{num}}}
```

\memgloterm{⟨*term*⟩}
\memglodesc{⟨*desc*⟩}
\memgloref{⟨*ref*⟩}
\memglonum{⟨*num*⟩}

These macros can be redefined to format the various parts of a glossary entry. Their default definitions are simply

```
\newcommand{\memgloterm}[1]{#1}
\newcommand{\memglodesc}[1]{#1}
\newcommand{\memgloref}[1]{#1}
\newcommand{\memglonum}[1]{#1}
```

For example, if you wanted the term in bold, the description in italics, and no numbers:

```
\renewcommand{\memgloterm}[1]{\textbf{#1}}
\renewcommand{\memglodesc}[1]{\textit{#1}}
\renewcommand{\memglonum}[1]{}
```

There are several macros that effect a glossary entry but which must not be directly modified (the `\memglonumf` shown above as part of the `\glossaryentry` is one of these). Each of the following `\changegloss...` macros takes an optional ⟨*file*⟩ argument. The changes to the underlying macro apply only to the glossary of that particular ⟨*file*⟩ (or the `\jobname` file if the argument is not present.

`\changeglossactual[`⟨*file*⟩`]{`⟨*char*⟩`}`
`\changeglossref[`⟨*file*⟩`]{`⟨*thecounter*⟩`}`
`\changeglossnum[`⟨*file*⟩`]{`⟨*thecounter*⟩`}`
`\changeglossnumformat[`⟨*file*⟩`]{`⟨*format*⟩`}`

`\changeglossactual` sets ⟨*char*⟩ as the `actual` character for the ⟨*file*⟩ glossary. It is initially `@`. This must match with the `actual` specified for the `gst` file that will be applied.

`\changeglossref` specifies that ⟨*thecounter*⟩ should be used to generate the ⟨*ref*⟩ for the ⟨*file*⟩ glossary. It is initially nothing.

`\changeglossnum` specifies that ⟨*thecounter*⟩ should be used to generate the ⟨*num*⟩ for the ⟨*file*⟩ glossary. It is initially `\thepage`.

`\changeglossnumformat` specifies that ⟨*format*⟩ should be used to format the ⟨*num*⟩ for the ⟨*file*⟩ glossary. The format of ⟨*format*⟩ is `|form`, where `|` is the `encap` character specified in the `gst` file, and `form` is a formatting command, taking one argument (the number), without any backslash. For example

```
\changeglossnumformat{|textbf}
```

to get bold numbers. It is initially set as `|memjustarg`, where this is defined as:

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

There must be a format defined for the ⟨*num*⟩ otherwise the arguments to `\glossitem` will not be set correctly.

The `\makeglossary` command uses the `\change...` commands to define the initial versions, so only use the `\change...` macros *after* `\makeglossary`. In this document an early version of the glossary was set up by

```
\makeglossary
\changeglossactual{?}
\makeatletter
\changeglossnum{\@currentlabel}
\makeatother
\changeglossnum{\thepage}
```

The first call of `\changeglossnum` makes the number the current numbered chapter, or numbered section, or numbered .... I didn't like that when I tried it, so the second call resets the number to the page number.

THE LISTING

The final glossary data in the `gls` file is typeset in the `theglossary` emnvironment, which is much like the `theindex` and `thebibliography` environments.

The environment starts off with a chapter-style unnumbered title. There are several macros for specifying what happens after that.

`\glossaryname`
`\glossarymark`
`\glossaryintoc` `\noglossaryintoc`

The title for the glossary is `\glossaryname` whose initial definition is
`\newcommand*{\glossaryname}{Glossary}`
`\glossarymark`, which by default does nothing, can be redefined to set marks for headers. The glossary title will be added to the ToC if the `\glossaryintoc` declaration is in force, but will not be added to the ToC following the `\noglossaryintoc`.

`\preglossaryhook`

The macro `\preglossaryhook` is called after the glossary title has been typeset. By default it does nothing, but you could redefine it to, for example, add some explanatory material before the entries start.

`\onecolglossarytrue` `\onecolglossaryfalse`
`\glossarycolsep` `\glossaryrule`

The glossary can be typeset in two columns (`\onecolglossaryfalse`) but by default (`\onecolglossarytrue`) it is set in one column. When two columns are used, the length `\glossarycolsep` is the distance between the columns and the length `\glossaryrule` is the width (default 0) of a vertical rule between the columns.

`\begintheglossaryhook`
`\atendtheglossaryhook`

The last thing that `\begin{theglossary}` does is call `\begintheglossaryhook`. Similarly, the first thing that is done at the end of the environment is to call `\atendtheglossaryhook`. By default these macros do nothing but you can redefine them.

For example, if you wanted the glossary in the form of a description list, the following will do that.

```
\renewcommand*{\begintheglossaryhook}{\begin{description}}
\renewcommand*{\atendtheglossaryhook}{\end{description}}
\renewcommand{\glossitem}[4]{\item[#1:] #2 #3 #4}
```

THE GLOSSARY FOR THIS DOCUMENT

The following is the code I have used to produce the glossary in this document.

This is the code that is in the preamble.

```
%%%% in the preamble
\makeglossary
\changeglossactual{?}
\changeglossnum{\thepage}
\changeglossnumformat{|hyperpage}%% for hyperlinks
\renewcommand*{\glossaryname}{Command summary}
```

```
\renewcommand*{\glossarymark}{\markboth{\glossaryname}{}}

\makeatletter
\renewcommand{\glossitem}[4]{%
  \sbox\@tempboxa{#1 \space #2 #3 #4}%
  \par\hangindent 2em
  \ifdim\wd\@tempboxa<0.8\linewidth
    #1 \space #2 #3 \dotfill #4\relax
  \else
    #1 \dotfill #4\\
    #2 #3
  \fi}
\makeatother
```

The redefinition of `\glossitem` works as follows (it is similar to code used in the setting of a `\caption`):

1. Put the whole entry into a temporary box.
2. Set up a hanging paragraph with 2em indentation after the first line.
3. Check if the length of the entry is less than 80% of the linewidth.
4. For a short entry set the name, description, and any reference then fill the remainder of the line with dots with the number at the right margin.
5. For a longer entry, set the title and number on a line, separated by a line of dots, then set the description (and reference) on the following lines.

In the document the raw data is collected by the `\glossary` commands in the body of the text. For instance, although I have not actually used the first two:

```
\glossary(cs)%
  {\cs{cs}\gmarg{name}}%
  {Typesets \texttt{name} as a macro name with preceding backslash,
   e.g., \cs{name}.}%
\glossary(gmarg)%
  {\cs{gmarg}\gmarg{arg}}%
  {Typesets \texttt{arg} as a required argument, e.g., \gmarg{arg}.}
\glossary(glossaryname)%
  {\cs{glossaryname}}%
  {Name for a glossary}%
\glossary(memgloterm)%
  {\cs{memgloterm}\gmarg{term}}%
  {Wrapper round a glossary term.}%
```

Any change to the glossary entries will be reflected in the `glo` produced from that LaTeX run. Then run MakeIndex on the `glo` file using the appropriate `gst` configuration file, and run LaTeX again to get the corrected, sorted and formatted result printed by `\printglossary`.

Four

Boxes and environments

4.1 Sidebars

It is now possible to control on which side of the page a sidebar gets placed.

`\sidebarmargin{`⟨*margin*⟩`}`

For onecolumn documents and with `\sidebaronesidefalse` you can use `\sidebarmargin` to specify which margin you want sidebars to be located. Posssible values for ⟨*margin*⟩ are: `left`, `right`, `inner` or `outer` with the obvious meanings. (`\sidebarmargin{right}` and `\sidebaronesidetrue` are equivalent.) There are no positioning options for twocolumn documents.

4.2 Boxed verbatims

`\bvendofpage{`⟨*code*⟩`}`

A boxed verbatim may extend across a page break and the `\bvendofpage` macro determines what happens at the bottom of the page just before the break. The default definitions is:

```
\newcommand{\bvendofpage}{\hrule\kern-.4pt}
```

which results in a horizontal rule being drawn. A `\hrule` takes 0.4pt of vertical space and the `\kern` of -0.4pt backs up by 0.4pt vertically, so as far as TEX is concerned no space has been used. You may change the macro to something that better matches your needs if necessary.

4.3 Verse

There are a couple of small additions to the code for verse.

`\vleftofline{`⟨*text*⟩`}`

A verse line may start with something, for example open quote marks, where it is desirable that it be ignored as far as the alignment of the remainder of the line is concerned[1] — a sort of 'hanging left punctuation'. When it is put at the start of a line in the `verse` environment the ⟨*text*⟩ of `\vleftofline` is typeset but ignored as far as horizontal indentation is concerned.

Compare the two settings below:

```
\noindent ``No, this is what was spoken by the prophet Joel:
\begin{verse}
``\,`\,``In the last days,'' God says, \\
``I will pour out my Spirit on all people. \\
Your sons and daughters will prophesy, \\
\ldots \\
And everyone who calls \ldots ''\,'
\end{verse}
```

"No, this is what was spoken by the prophet Joel:
"'"In the last days," God says,
"I will pour out my Spirit on all people.
Your sons and daughters will prophesy,
...
And everyone who calls ..."'

```
\noindent ``No, this is what was spoken by the prophet Joel:
\begin{verse}
\vleftofline{``\,`\,``}In the last days,'' God says, \\
\vleftofline{``}I will pour out my Spirit on all people. \\
Your sons and daughters will prophesy, \\
\ldots \\
And everyone who calls \ldots ''\,'
\end{verse}
```

"No, this is what was spoken by the prophet Joel:
"'"In the last days," God says,
"I will pour out my Spirit on all people.
Your sons and daughters will prophesy,
...
And everyone who calls ..."'

`\vleftmargin`

In the basic LaTeX `verse` environment the body of the verse is indented from the left of the text block by an amount `\leftmargini`, as is the text in many other environments based on the basic LaTeX `list` environment. For memoir's `verse` environment only, the default indent is set by the length `\vleftmargin` (which is initially set equal to `\leftmargini`). For poems with particularly long lines it could, for example, be advantageous to eliminate any indentation via:

```
\setlength{\vleftmargin}{0em}
```

[1] The problem was presented to me by Matthew Ford who also provided the example text.

If necessary the verse could even be moved into the left margin by giving `\vleftmargin` a negative length value, such as -0.3em.

Later, for poems with shorter lines the indentation can be reset to the default by:
`\setlength{\vleftmargin}{\leftmargini}`

`\verselinenumbersright` `\verselinenumbersleft`

Following the declaration `\verselinenumbersright`, which is the default, any verse line numbers will be set in the righthand margin. The `\verselinenumbersleft` declaration will set any subsequent line numbers to the left of the lines.

Five

Captions

5.1 Side captions

The class now provides for placing captions at the side of figures or tables, or other floats.

\begin{sidecaption}[⟨*fortoc*⟩]{⟨*title*⟩}[⟨*label*⟩] the body of the float \end{sidecaption}

The `sidecaption` environment is used for a sidecaption rather than a macro. The body of the float is put inside the environment. For example:

```
\begin{figure}
  \begin{sidecaption}{An illustration}[fig:ill]
    \centering
    \includegraphics{...}
  \end{sidecaption}
\end{figure}
```

whereby the caption, 'Figure N: An illustration', will be placed in the margin alongside the graphic, and for reference purposes will be given given the `\label fig:ill`.

\sidecapwidth \sidecapsep \sidecapmargin{⟨*margin*⟩} \ifscapmargleft \scapmargleftttrue \scapmargleftfalse

The caption is set in a box `\sidecapwidth` wide (the default is `\marginparwidth`) offset `\sidecapsep` (default `\marginparsep`) into the margin.

If the float is a single column float in a twocolumn document then the caption is always[1] placed in the adjacent margin, otherwise the `\sidecapmargin` command controls the margin where the sidecaption will be placed. The possible values for ⟨*margin*⟩ are one of: `left`, `right`, `inner`, or `outer`. If `left` or `right` is specified the caption will go into

[1] Well, nearly always. See the `\overridescapmargin` command later.

the left or right margin. If `inner` or `outer` is specified then in a two sided document the caption will be on different sides of the textblock according to whether it is a recto or verso page; in a one sided document the caption margin is fixed. The left margin is the default.

When the caption is to be set in the left margin, `\ifscapmargleft` is set `true`, and for a right margin it is set `false`.

`\setsidecappos{`⟨*pos*⟩`}`

By default a sidecaption is vertically centered with respect to the float it is captioning. This can be altered by using the `\setsidecappos` declaration. The allowed values for ⟨*pos*⟩ are:

`t` — the top of the caption is aligned with the top of the float

`c` — (the default) the center of the caption is aligned with the center of the float

`b` — the bottom of the caption is aligned with the bottom of the float

The other kinds of simple captions can also be put at the side of a float. The positioning and styling commands for these are exactly those for `sidecaption`. Bilingual captions can only be placed above or below the float.

`\begin{sidecontcaption}{`⟨*title*⟩`}[`⟨*label*⟩`]`
the body of the float
`\end{sidecontcaption}`

Sidecaptions may be continued with the `sidecontcaption` environment.

`\begin{sidenamedlegend}[`⟨*fortoc*⟩`]{`⟨*title*⟩`}`
the body of the float
`\end{sidenamedlegend}`

Named legends may be set at the side with the `sidenamedlegend` environment.

`\begin{sidelegend}{`⟨*title*⟩`}`
the body of the float
`\end{sidelegend}`

Legends may be set at the side with the `sidelegend` environment.

TWEAKS

`\sidecapstyle`

Just before the caption is set, the `\sidecapstyle` command is called. This may be used to set the styling for the particular caption. By default it sets captions that are in the left margin raggedleft, and those that are in the right margin are set raggedright. The default definition is:

```
\newcommand*{\sidecapstyle}{%
%%%  \captionnamefont{\bfseries}%
  \ifscapmargleft
    \captionstyle{\raggedleft}%
  \else
    \captionstyle{\raggedright}%
  \fi}
```

You can change the command to suit your purposes; for example, uncommenting the `\captionnamefont` line would result in the caption's float name being set in a bold font. See the manual for more information on what you can do to change the appearance of a caption.

`\overridescapmargin{`⟨*margin*⟩`}`
`\sidecapraise`

Sometimes the caption may not be placed exactly where you want it — it may be in the wrong margin or at the wrong height.

The command `\overridescapmargin` will force the following caption into the ⟨*margin*⟩ you specify which can only be `left` or `right`. In a twosided document where `\sidecapmargin` is `inner` or `outer` and the caption goes in the wrong margin, it is likely that the declaration `\strictpagechecktrue` will solve the problem. The wrong margin might be chosen in a twocolumn document where the float is in the second column; use

```
\overridescapmargin{right}
```

to fix this.

The caption may not be at quite the height you want with respect to the float. The caption will be raised by the length `\sidecapraise` in addition to the calculated movement (or lowered if `\sidecapraise` is negative).

`\sidecapfloatwidth{`⟨*length*⟩`}`

The float is set in a `minipage` with width `sidecapfloatwidth`, whose default definition is

```
\newcommand*{\sidecapfloatwidth}{\linewidth}
```

That is, the normal width is the same as the current `\linewidth`. For a narrow table, say, you may want to reduce this, for example to half by

```
\renewcommand*{\sidecapfloatwidth}{0.5\linewidth}
```

Note that `\sidecapfloatwidth` is a macro, not a length, so it must be altered by using a `\renewcommand*`, *not* by `\setlength`.

If you do reduce the `\sidecapfloatwidth` you may notice that the sidecaption is actualy placed a distance `\sidecapsep` with respect to the float's `minipage`, not with respect to the text block.

Table 5.1 was created by the following code.

```
\newlength{\mylength}
\setlength{\mylength}{\linewidth}
\addtolength{\mylength}{-\sidecapsep}
\addtolength{\mylength}{-\sidecapwidth}
```

Table 5.1: Permitted arguments for some sidecaption related commands

\sidecapmargin	\overridescapmargin
left	left
right	right
inner	
outer	

```
\begin{table}
  \sidecapmargin{left}%
  \renewcommand*{\sidecapfloatwidth}{\mylength}%
  \raggedleft
  \begin{sidecaption}{%
    Permitted arguments for some sidecaption related commands}[scap:one]
  \centering
  \begin{tabular}{cc} \toprule
  \cs{sidecapmargin} & \cs{overridescapmargin} \\ \midrule
  \texttt{left}      & \texttt{left}       \\
  \texttt{right}     & \texttt{right}       \\
  \texttt{inner}     &  \\
  \texttt{outer}     &  \\ \bottomrule
  \end{tabular}
\end{sidecaption}
\end{table}
```

The calculations on the `\mylength` length are so that the sidecaption and float will just fit inside the textblock.

Note that the `\raggedleft` command before the `sidecaption` environment makes the float's `minipage` be placed raggedleft (i.e., moved across to the right hand edge of the textblock) while the `\centering` centers the `tabular` within the minipage. You can get a variety of horizontal placements by judicious use of `\raggedright`, `\centering` and `\raggedleft` commands. If you do move the float sideways to leave space for the caption make sure that the caption will go to the side you want. In the example code I 'moved' the float to the right so I made sure that the caption would go on the left by explicitly setting
`\sidecapmargin{left}`

As far as TeX is concerned a sidecaption takes no horizontal space. If you use a sidecaption in a wrapped float from, say, the wrapfig package, make sure that the sidecaption gets placed where it won't be overlaid by the main text.

5.2 Caption title text

`\captiontitlefinal{`⟨*stuff*⟩`}`

`\captiontitlefinal{`⟨*stuff*⟩`}` will put ⟨*stuff*⟩ immediately at the end of a caption's title, but ⟨*stuff*⟩ will not appear in the LoF or LoT. The default is
`\captiontitlefinal{}`
but it could be called instead as, say

`\captiontitlefinal{.}`
to put a period (full stop) after the title.

Six

Miscellaneous

6.1 General

Font commands

The `\em` command is no longer deprecated (it was a misreading on my part to deprecate it in the first place).

`\eminnershape{`⟨*shape*⟩`}`

If the `\emph` commmand is used within italic text then the newly emphasized text will be typeset using the `\eminnershape` font shape. The default definition is:

```
\newcommand*{\eminnershape}{\upshape}
```

which you can change if you wish.

Footnotes

There was a question on CTT asking how to ensure that footnotes were at the bottom of the page when `\raggedbottom` was in effect. The solution is to add a `\vfill` to the `\footnoterule` macro, as below.

```
\renewcommand*{\footnoterule}{\kern-3pt\vfill
  \hrule width 0.4\columnwidth \kern 2.6pt}
```

Floats

A page which consists of floats (e.g., `table`, `figure`) with no body text is called a *floatpage*. By default, floats on a floatpage are centered vertically. To move the floats to the top of the page try

```
\makeatletter
  \setlength{\@fptop}{0pt}
  \setlength{\@dblfptop}{0pt}
\makeatother
```

To move the floats to the bottom of the page, replace `\@fptop` and `\@dblfptop` by `\@fpbot` and `\@dblfpbot` respectively.

One author thought it would be nice to be able to have different page headings according to whether the page was a floatpage, or there was a float at the top of the page, or a float at the bottom of a page or there was text at the top and bottom.

This, I think, is not a common requirement and, further, that to provide this involves changing parts of the LaTeX output routine — something only to be tackled by the bravest of the brave. If it were to be done then were best done in a package that could be easily ignored. The following is an outline of what might be done; I do not recommend it and if you try this and all your work dissappears then on your own head be it.

```
% notefloat.sty
\newif\iffloatattop
  \floatattopfalse
\newif\iffloatatbot
  \floatatbotfalse

\renewcommand*{\@addtotoporbot}{%
  \@getfpsbit \tw@
  \ifodd \@tempcnta
    \@flsetnum \@topnum
    \ifnum \@topnum>\z@
      \@tempswafalse
      \@flcheckspace \@toproom \@toplist
      \if@tempswa
        \@bitor\@currtype{\@midlist\@botlist}%
        \if@test
        \else
          \@flupdates \@topnum \@toproom \@toplist
          \@inserttrue
  \global\floatattoptrue
        \fi
      \fi
    \fi
  \fi
  \if@insert
  \else
    \@addtobot
  \fi}

\renewcommand*{\@addtobot}{%
  \@getfpsbit 4\relax
  \ifodd \@tempcnta
    \@flsetnum \@botnum
    \ifnum \@botnum>\z@
      \@tempswafalse
      \@flcheckspace \@botroom \@botlist
      \if@tempswa
```

```
        \global \maxdepth \z@
        \@flupdates \@botnum \@botroom \@botlist
        \@inserttrue
   \global\floatatbottrue
       \fi
     \fi
   \fi}

 \let\p@wold@output\@outputpage
 \renewcommand*{\@outputpage}{%
   \p@wold@output
   \global\floatattopfalse
   \global\floatatbotfalse}

 \endinput
```

`\floatattop` is probably set TRUE if there is a float at the top of the page and `\floatatbot` is probably set TRUE if there is a float at the bottom of the page.

TWO SIMPLE MACROS

There are two trivial macros that can be generally useful.

`\memjustarg{`⟨*text*⟩`}`
`\memgobble{`⟨*text*⟩`}`

The `\memjustarg` macro just uses its argument and is defined as:

```
\newcommand*{\memjustarg}[1]{#1}
```

The `\memgobble` macro gobbles down and swallows its argument. Its definition is:

```
\newcommand{\memgobble}[1]{}
```

Do *not* redefine either `\memjustarg` or `\memgobble`; if you do various pieces of code will behave in unexpected ways that you will not like.

6.2 For package writers

The facilities described in this section are for anyone to use but I suspect that they may be most useful to package developers.

EMULATING PACKAGES

`\EmulatedPackage{`⟨*package*⟩`}[`⟨*date*⟩`]`
`\EmulatedPackageWithOptions{`⟨*optionlist*⟩`}{`⟨*package*⟩`}[`⟨*date*⟩`]`

These commands are for package writers; they are based on a conversation with Donald Arseneau on CTT. They fool LaTeX into thinking that the ⟨*package*⟩ has already been loaded so it won't try loading it again. These are probably only useful if your package includes the actual code for ⟨*package*⟩. (memoir does include code from several packages and uses a similar internal command to ensure that the packages are not loaded following some later `\usepackage` command.)

EXTENDING A MACRO

```
\patchcommand{⟨macro⟩}{⟨start-code⟩}{⟨end-code⟩}
```

The `\patchcommand` is from the late Michael Downes' `patchcmd` package [Dow00]. It inserts the ⟨*start-code*⟩ at the start of the current definition of the macro ⟨*macro*⟩, and inserts ⟨*end-code*⟩ at the end of its current definition. The ⟨*macro*⟩ can have zero to nine parameters. If ⟨*macro*⟩ uses `\futurelet` (e.g., it is a starred command or takes an optional argument) only ⟨*start-code*⟩ is useful — ⟨*end-code*⟩ must be empty otherwise things get messed up. If ⟨*macro*⟩ has any delimited arguments then `\patchcommand` connot be used.

INSERTING CODE BEFORE AND AFTER A FILE, PACKAGE OR CLASS

The kernel provides two commands, `\AtBeginDocument` and `\AtEndDocument` which can only be used in the preamble, for inserting code at the start and end of the `document` environment.

The kernel also provides the macros `\AtEndOfPackage{`⟨*code*⟩`}` and `\AtEndOfClass{`⟨*code*⟩`}` for inserting code at the end of the current package or class. More precisely, these macros call the ⟨*code*⟩ after the package or class file has been input via `\InputIfFileExists`.

The class provides a more comprensive set of macros for code insertions, which should be used before the relevant file is called for.

```
\AtBeginFile{⟨file⟩}{⟨code⟩}
\AtEndFile{⟨file⟩}{⟨code⟩}
```

The `\AtBeginFile` macro inserts ⟨*code*⟩ just before the ⟨*file*⟩ file is `\input` (or `\included`, etc.). Similarly `\AtEndFile` inserts the ⟨*code*⟩ immediately after the ⟨*file*⟩. The ⟨*file*⟩ argument must be the same as used in the corresponding `\input` command. If ⟨*file*⟩ includes an extension, for example `fred.def`, then that is taken as the complete name, otherwise if there is no extension, for instance `fred`, then the `.tex` extension is automatically appended making the full name `fred.tex`.

The `\At...File` commands must be issued *before* the corresponding ⟨*file*⟩ is input otherwise nothing will happen.

```
\AtBeginPackage{⟨pack⟩}{⟨code⟩}
\AtEndPackage{⟨pack⟩}{⟨code⟩}
\RequireAtEndPackage{⟨pack⟩}{⟨code⟩}
```

The `\AtBeginPackage` command will insert ⟨*code*⟩ just before the ⟨*pack*⟩ package is used. Similarly `\AtEndPackage` will insert the ⟨*code*⟩ immediately after the ⟨*pack*⟩. The ⟨*pack*⟩ argument must be the same as used in the corresponding `\usepackage` command, that is, without any extension. The `\At...Package` commands must be issued *before* the corresponding ⟨*pack*⟩ is used otherwise nothing will happen.

The `\RequireAtEndPackage` command will, like `\AtEndPackage`, insert ⟨*code*⟩ at the end of the ⟨*pack*⟩ package if it has not yet been used. If the package has already been used then the ⟨*code*⟩ is called immediately.

\AtBeginClass{⟨*class*⟩}{⟨*code*⟩} \AtEndClass{⟨*class*⟩}{⟨*code*⟩} \RequireAtEndClass{⟨*class*⟩}{⟨*code*⟩}

The \AtBeginClass command will insert ⟨*code*⟩ just before the ⟨*class*⟩ class is used. Similarly \AtEndClass will insert the ⟨*code*⟩ immediately after the ⟨*class*⟩. The ⟨*class*⟩ argument must be the same as used in the corresponding \LoadClass command, that is, without any extension. The \At...Class commands must be issued *before* the corresponding ⟨*class*⟩ is used otherwise nothing will happen.

The \RequireAtEndClass command will, like \AtEndClass, insert ⟨*code*⟩ at the end of the ⟨*class*⟩ class if it has not yet been used. If the class has already been used then the ⟨*code*⟩ is called immediately.

There is an unfortunate interaction between the kernel's \AtEndOfPackage and the class's \AtEndPackage, and similarly for the \AtEndOfClass and \AtEndClass. I discovered this when I tried to automate using the memhfixc package if hyperref was being used by putting the following into the memoir code

```
\AtEndPackage{hyperref}{\usepackage{memhfixc}}
```

which caused all sorts of problems.

The kernel scheme scheme looks like this:

```
\newcommand{\usepackage}[1]{%
  ...
  \InputIfFileExists{#1}
<AtEndOfPackage code>}
```

The basic mechanism for implementing the class macros is by modifying the kernel's \InputIfFileExists macro, which internally uses a form of \input to read in the file, so that the inserted ⟨*code*⟩ comes immediately before and after the \input, somewhat like:

```
\renewcommand{\InputIfFileExists}[1]{%
  ...
  <before code> \input{#1} <after code>}
```

If \AtEndPackage is applied to a package that has an internal \AtEndOfPackage then the result can be sketched as:

```
\newcommand{\usepackage}[1]{%
  ...
  <before code>
  \input{#1}
  <after code>
  <AtEndOfPackage code>
}
```

In other words the body of the package is read in, the \AtEndPackage code is called, and then *after* that the \AtEndOfPackage code is called.

The hyperref package internally uses \AtEndOfPackage to read some files and memhfixc had to be input after these. A way to automate memhfixc after hyperref is:

```
\AtEndPackage{hyperref}{%
  \AtBeginDocument{\usepackage{memhfixc}}}
```

but this seems more trouble than it's worth.

6.3 Heading hooks

On 2nd September 2005 I posted two messages to the `comp.text.tex` newsgroup saying that I was creating a new version of memoir and that I would consider inserting hooks into the class code that package writers might find useful. I got no requests for any hooks or anything else from package writers. I therefore assume that no package author sees any problems if a memoir class document author uses the package.

However, I have provided macros that that may be useful for those who want to do things with the contents of section headings, captions, and the like. The macros are called within the relevant heading or caption code, and by default are defined to do nothing.

Hooks for the `\book` and `\book*` commands.

```
\membookinfo{⟨thebook⟩}{⟨fortoc⟩}{⟨title⟩}
\membookstarinfo{⟨title⟩}
```

Hooks for the `\part` and `\part*` commands.

```
\mempartinfo{⟨thepart⟩}{⟨fortoc⟩}{⟨title⟩}
\mempartstarinfo{⟨title⟩}
```

In many cases a `\mem...info` macro includes an argument related to the heading's number (⟨*thepart*⟩ for `\mempartinfo`). In certain circumstances, such as a `\chapter` in the `\frontmatter`, there might not be a number even though the normal unstarred version of the command is used. In these cases the number argument (⟨*thechapter*⟩ in the case of `\memchapinfo`) is left empty.

Hooks for the `\chapter` and `\chapter*` commands. Note that regular chapters and those as appendices are treated differently.

```
\memchapinfo{⟨thechapter⟩}{⟨fortoc⟩}{⟨forhead⟩}{⟨title⟩}
\memchapstarinfo{⟨fortoc⟩}{⟨title⟩}
\memappchapinfo{⟨thechapter⟩}{⟨fortoc⟩}{⟨forhead⟩}{⟨title⟩}
\memappchapstarinfo{⟨fortoc⟩}{⟨title⟩}
```

Hooks for `\section`, `\subsection`, etc., and their starred versions. ⟨*name*⟩ is the type of section (e.g., `section`, or `subsection`, or `subsubsection` or ...

```
\memsecinfo{⟨name⟩}{⟨thename⟩}{⟨fortoc⟩}{⟨forhead⟩}{⟨title⟩}
\memsecstarinfo{⟨name⟩}{⟨title⟩}
```

Hooks for appendix-like page headings.

```
\memapppageinfo{⟨title⟩}
\memapppagestarinfo{⟨title⟩}
\memleadpageinfo{⟨pstyle⟩}{⟨cmdname⟩}{⟨title⟩}
\memleadpagestarinfo{⟨pstyle⟩}{⟨cmdname⟩}{⟨title⟩}
```

Hooks for `\poemtitle`, `\PoemTitle`, and their starred versions.

```
\mempoeminfo{⟨title⟩}
\mempoemstarinfo{⟨title⟩}
\memPoemTitleinfo{⟨thepoem⟩}{⟨fortoc⟩}{⟨forhead⟩}{⟨title⟩}
\memPoemTitlestarinfo{⟨fortoc⟩}{⟨title⟩}
```

Hooks for the several kinds of `\caption` and `\legend` commands.

```
\memcaptioninfo{⟨type⟩}{⟨thetype⟩}{⟨fortoc⟩}{⟨title⟩}
\memlegendinfo{⟨title⟩}
\memnamedlegendinfo{⟨fortoc⟩}{⟨title⟩}
\membitwonumcaptioninfo{⟨type⟩}{⟨thetype⟩}{⟨fortoc1⟩}{⟨title1⟩}
                        {⟨name2⟩}{⟨fortoc2⟩}{⟨title2⟩}
\membionenumcaptioninfo{⟨type⟩}{⟨thetype⟩}{⟨fortoc1⟩}{⟨title1⟩}
                        {⟨name2⟩}{⟨fortoc2⟩}{⟨title2⟩}
\membicaptioninfo{⟨type⟩}{⟨thetype⟩}{⟨fortoc1⟩}{⟨title1⟩}{⟨name2⟩}{⟨title2⟩}
```

Command summary

Bibliography

[CB99] Warren Chappell and Robert Bringhurst. *A Short History of the Printed Word*. Hartley & Marks, 1999, ISBN 0-88179-154-7.

[CH88] Pehong Chen and Michael A. Harrison. *Index Preparation and processing*. September 1988. (Available from CTAN in `indexing/makeindex/paper`)

[Dow00] Michael J. Downes. *The patchcmd package*. July 2000. (Available from CTAN in `macros/latex/contrib/patchcmd`)

[CMS93] John Grossman (Managing Editor). *The Chicago Manual of Style*. Fourteenth Edition. University of Chicago Press, 1993.

[MG+04] Frank Mittelbach, Michael Goossens, et al. *The LaTeX companion: Second Edition*. Addison-Wesley, 2004. ISBN 0-201-36299-6.

Index

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