

The LaTeX memoir class for configurable book typesetting: Source code*

Peter Wilson[†]
Herries Press
(with the assistance of Lars Madsen)

2010/04/19

Abstract

The memoir class is designed for typesetting general books such as novels, biographies, histories, and so on, although as it supports all the functionality of the standard book class it can also be used for technical writing. It provides more functions than the standard class as well as presenting a more friendly interface for the book designer. It can also simulate the typesetting style of the standard article class.

The class was first released in mid 2001 and has been well used ever since.

Contents

1	Introduction	7
2	A driver for this document	8
3	Identification	9
4	Initial Code	9
4.1	Classes, Packages and Files	13
4.2	For package documentation	16
4.3	Checking the processor	17
4.4	Extending an existing macro	20
4.5	Support for the bidi package (RTL typesetting)	25

*This file (`memoir.dtx`) has version number v3.6, last revised 2010/04/19.

[†]`herries dot press at earthlink dot net`

5	Declaration of Options	26
5.1	Setting Paper Sizes	26
5.2	Choosing the type size	30
5.3	Two-side or one-side printing	30
5.4	Two column printing	31
5.5	The <code>draft</code> option	31
5.6	The <code>ms</code> option	31
5.7	The <code>showtrims</code> option	32
5.8	The <code>article</code> option	32
5.9	The <code>subfigure</code> option	32
5.10	The <code>openright</code> , <code>openleft</code> and <code>openany</code> options	32
5.11	Equation numbering on the left	33
5.12	Flush left math displays	33
5.13	Open bibliography	33
5.14	Old font commands	33
5.15	Extra font sizes	33
5.16	ETeX	33
6	Executing Options	34
7	Fonts and spaces	36
7.1	Fonts	36
7.2	Paragraphing	48
7.3	Vertical spacing	50
7.4	Footnotes	52
7.5	Floats	53
7.6	The measure	58
8	Page Layout	60
8.1	The <code>typeblock</code> and margins	60
8.2	Some predefined layouts	76
8.3	Float placement parameters	78
9	Page Styles	79
9.1	Marking conventions	79
9.2	Defining the page styles	80
9.3	Page numbering	97
10	Non-traditional spacing	98
10.1	Double spacing	98
10.2	Abnormal parskip	102
11	Titles	103

12 Parts, chapters and other divisions	109
12.1 Building blocks	109
12.2 Mark commands	110
12.3 Define Counters	111
12.4 Front, main and back matter	112
12.5 Book	114
12.6 Part	117
12.7 Chapter	120
12.7.1 Chapter styling	124
12.8 Lower level headings	135
12.8.1 Anonymous headings	142
12.9 Division head styles	146
12.10 Appendices	153
12.11 Appendixpage-like pages	156
12.12 Paragraphs	158
12.12.1 Normal (block) paragraphs	158
12.12.2 Centered lines	159
12.12.3 Ragged	160
12.12.4 Hanging	161
12.12.5 Miscellaneous	161
13 Lists	162
13.1 General List Parameters	162
13.2 Enumerate	168
13.3 Itemize	171
13.4 Description	172
13.5 Quotation	173
13.6 Quote	173
13.7 Theorem	173
13.8 Listing of symbols and abbreviations	174
14 Abstracts	174
15 Verse	178
15.1 Environments	178
15.2 Patterns	183
15.3 Titles	187
16 Setting parameters for existing environments	191
16.1 Array and tabular	191
16.2 Tabbing	192
16.3 Minipage	192
16.4 Framed boxes	192
16.5 Equation and eqnarray	192

17 Array and tabular	193
17.1 Array	193
17.1.1 The construction of the preamble	193
17.1.2 Building and calling <code>\halign</code>	205
17.1.3 The line separator <code>\\</code>	207
17.1.4 Spanning several columns	208
17.1.5 The environment definitions	208
17.1.6 Defining your own column specifiers	210
17.1.7 The <code>*</code> -form	212
17.2 Getting the spacing around rules right	213
17.3 D column specifiers	215
17.4 Support for delimiters	217
17.5 The <code>tabularx</code> environment	219
17.6 Fear's rules	227
17.6.1 Full width rules	229
17.6.2 Special subrules	231
17.7 Input files into tabulars, and <code>etex</code>	234
17.8 Continuous tabulars	235
17.8.1 Horizontal lines	237
17.9 Automated tabulations	238
18 Floating objects	243
18.1 Floats	244
18.2 Captions	247
18.2.1 Continuation captions and legends	251
18.2.2 Non-float captions	253
18.2.3 Bilingual captions	253
18.2.4 Support for the <code>subfigure</code> package functionality	256
18.2.5 Side captions	264
19 Epigraphs	273
19.1 Epigraphs before a chapter title	275
20 The deprecated font commands	277
21 Cross Referencing	279
21.1 Label referencing	279
21.2 Title referencing	280
22 Table of Contents, etc.	282
22.1 New List of ...	283
22.2 Table of Contents	286
22.3 List entries	287
22.4 Support for the <code>subfigure</code> package	297
22.5 Switching page numbering	297
22.6 Chapter precis	298

22.7 Adding things to the ToC	299
22.8 ToC and divisional numbering	300
23 Bibliography	304
23.1 Use with the natbib and chapterbib packages	306
24 The index	307
25 Page bottom	314
25.1 Widows and sloppybottom	315
26 Glossaries	315
27 Notes/Marginalia	320
27.1 A simple interface for specifying locations	320
27.2 Marginpars	322
27.2.1 Marginpar – margin interface	324
27.3 A fixed marginpar	325
27.4 Sidebars	328
27.5 Footnotes	333
27.6 Major extensions for footnotes	341
27.6.1 Two column footnotes	350
27.6.2 Three column footnotes	353
27.6.3 Paragraphed footnotes	355
27.7 Nasty insert bits	359
27.8 Side footnotes	363
27.8.1 Extension to the regular footnote	363
27.9 Bottom aligned side footnotes	364
27.9.1 The sidefootnote macros	368
27.10 End notes	372
28 Change marks	376
28.1 Print control	377
28.2 Change marking	377
29 Trimming marks	378
29.1 Quark marks	382
30 Verbatims, boxes, and files	385
30.1 Modified version of the verbatim package	385
30.1.1 Preliminaries	385
30.1.2 The <code>verbatim</code> and <code>verbatim*</code> environments	388
30.1.3 The <code>comment</code> environment	391
30.1.4 The main loop	392
30.1.5 The <code>\verbatiminput</code> command	399
30.2 Writing and boxing verbatim	402
30.3 The shortvrb package	403

30.4	General verbatim boxing and line numbering	405
30.5	The framed package	410
30.6	The newfile package	421
31	Utilities	426
31.1	Extra ‘provide’ commands	426
31.2	Changing counters	427
31.3	Odd/even page checking	429
31.4	Checking for empty arguments	431
31.5	Changing the page layout in the document	431
31.6	Temporarily changing the text width	432
31.7	Centering text	433
31.8	Moving from the current page	434
31.9	Needing space at the bottom of a page	435
31.10	Overlong lines	435
31.11	Text spacing commands	436
31.12	Fractions and subscripts	436
31.13	Numbers to names	437
31.14	A fix for two column headings	450
31.15	Time of day	451
31.16	Sequential sheet (page) numbers	452
31.17	Leaves per gathering	453
32	Initialization	454
32.1	Words and phrases	454
32.2	Date	455
32.3	Two column mode	455
32.4	The page style and counters	456
32.5	Single or double sided printing	456
32.6	Floats	457
32.7	The article option	458
32.8	The ms option	458
32.9	Emulated packages	459
32.10	Interaction with the caption package	460
32.11	Interaction with the float package	461
32.12	Patch file	461
33	Glossary Makeindex style file	461

List of Tables

1	Shorthand font point size commands	37
2	Arguments and results for <code>\setrectanglesize</code>	61
3	Arguments and results for <code>\setfillsize</code>	62
4	Document division levels	111

5	Classes of preamble tokens	194
---	--------------------------------------	-----

1 Introduction

This document provides the commented source for the LaTeX `memoir` class, which is designed for typesetting general books such as novels, biographies, histories, and so on. It has all the functionality of the standard `book` class and, as well as providing some extra functions, also provides a more friendly interface for the document designer. As it can encompass everything that the `book` class provides it may also be used for technical writing.

The default appearance of a document typeset with this class is the same as if it had been typeset with the `book` class and it can be made to simulate the `article` class.. The class, though, includes extra facilities that make it easy to change the appearance of such things as the page headers and footers, the style of chapter and other sectional headings, and the style of captions. It also makes it easy to both change the style of the Table of Contents, List of Figures, etc., as well as creating new kinds of ‘List of. . .’. New types of floats, if needed, can be created very simply. Epigraphs can be put into the document in a variety of styles.

The class provides a variety of page, chapter and captioning styles that you can choose from if you don’t want to create your own.

Documents can be typeset in 9pt, 10pt, 11pt, 12pt, 14pt or 17pt font sizes, and if you have scaleable fonts available they can be set at any size. There is a reasonably intuitive means of setting the margins and placement of the text on a page. There is an option to put trim marks on the printed pages if the stock sheets need to be trimmed down to the final page size. For those whose publishers like a manuscript to look as though it was typewritten, there is an option to do this (double spacing, ragged right, no hyphenation, fixed width font). There is also an unsophisticated means of flagging any revisions to the text.

As this is a new class, by default it does not support the old LaTeX v2.09 font commands, namely the `\bf`, `\sl`, `\it` and `\sc` commands; it warns about using the `\em` command but does support it.

I hope that apart from the font commands the class is compatible with ‘standard’ LaTeX.

Development of this class would never have been started without the wonderful work done by Leslie Lamport and others [LMB99] from whom I have learned a great deal and borrowed much code.

Sections 2 through 4 describe some administrative elements and code for general use later in the specification. The macros forming the class file are defined in sections 5 through 32.

This manual is typeset according to the conventions of the `LATEX` `DOCSTRIP` utility which enables the automatic extraction of the `LATEX` macro source files [GMS94].

2 A driver for this document

The next series of code contains the documentation driver file for L^AT_EX, i.e., the file that will produce the documentation you are currently reading. This will be extracted from this file by the DOCSTRIP program.

```

1 (*driver)
2 \documentclass[twoside]{ltxdoc}
3 \usepackage[T1]{fontenc}
4 \usepackage{url}
5 %%\usepackage[draft=false,
6 %%      plainpages=false,
7 %%      pdfpagelabels,
8 %%      bookmarksnumbered,
9 %%      hyperindex=true
10 %%      ]{hyperref} % Doesn't work with indexing of \DescribeMacro
11 \providecommand{\phantomsection}{} % just in case hyperref not used

```

We do want an index, using linenumbers, but not update information.

```

12 \EnableCrossrefs
13 \CodelineIndex
14 %% \RecordChanges

```

We had better have page headings to aid navigation, but I don't like Uppercased titles.

```

15 \makeatletter
16 \@mparswitchfalse
17 \makeatother
18 \renewcommand{\MakeUppercase}[1]{#1}
19 \pagestyle{headings}

```

We may use so many docstrip modules that we set the StandardModuleDepth counter to 1.

```

20 \setcounter{StandardModuleDepth}{1}

```

Some commonly used abbreviations

```

21 \newcommand*\Lopt[1]{\textsf {#1}} % typeset an option
22 \newcommand*\file[1]{\texttt {#1}} % typeset a file
23 \newcommand*\Lcount[1]{\textsl {\small#1}} % typeset a counter
24 \newcommand*\pstyle[1]{\textsl {#1}} % typeset a pagestyle
25 \newcommand*\Lenv[1]{\texttt {#1}} % typeset an environment
26 \newcommand*\Lpack[1]{\textsf {#1}} % typeset a package
27 \newcommand*\ctt{\textsc{ctt}} % comp.text.tex
28 \newenvironment{PW}{\em{}}
29 \newcommand*\theTeXbook{\textit{The \TeX book}}

```

We want the full details printed.

```

30 \begin{document}
31 \DeleteShortVerb{\|}
32 \raggedbottom
33 \raggedright
34 \DocInput{memoir.dtx}

```



```

35 \PrintIndex
36 %% \PrintChanges
37 \end{document}
38 </driver>

```

3 Identification

The memoir document class can only be used with LaTeX2e, so we make sure that an appropriate message is displayed when another T_EX format is used.

```

39 <class>\NeedsTeXFormat{LaTeX2e}

```

Announce the name, option files and version for LaTeX2e files:

```

40 <class>\ProvidesClass{memoir}%
41 <class> [2010/04/19 v3.6 configurable book, report, article document class]
42 <9pt>\ProvidesFile{mem9.clo}%
43 <9pt> [2008/01/30 v0.4 memoir class 9pt size option]
44 <10pt>\ProvidesFile{mem10.clo}%
45 <10pt> [2008/01/30 v0.3 memoir class 10pt size option]
46 <11pt>\ProvidesFile{mem11.clo}%
47 <11pt> [2008/01/30 v0.3 memoir class 11pt size option]
48 <12pt>\ProvidesFile{mem12.clo}%
49 <12pt> [2008/01/30 v0.4 memoir class 12pt size option]
50 <14pt>\ProvidesFile{mem14.clo}%
51 <14pt> [2008/01/30 v0.4 memoir class 14pt size option]
52 <17pt>\ProvidesFile{mem17.clo}%
53 <17pt> [2008/01/30 v0.3 memoir class 17pt size option]
54 <20pt>\ProvidesFile{mem20.clo}%
55 <20pt> [2008/01/31 v0.1 memoir class 20pt size option]
56 <25pt>\ProvidesFile{mem25.clo}%
57 <25pt> [2008/01/31 v0.1 memoir class 25pt size option]
58 <30pt>\ProvidesFile{mem30.clo}%
59 <30pt> [2008/01/31 v0.1 memoir class 30pt size option]
60 <36pt>\ProvidesFile{mem36.clo}%
61 <36pt> [2008/01/31 v0.1 memoir class 36pt size option]
62 <48pt>\ProvidesFile{mem48.clo}%
63 <48pt> [2008/01/31 v0.1 memoir class 48pt size option]
64 <60pt>\ProvidesFile{mem60.clo}%
65 <60pt> [2008/01/31 v0.1 memoir class 60pt size option]

```

4 Initial Code

```

66 <*class>

```

Note (2001/08/03): Old versions of the `amsmath` package did odd things with `\@tempa`, `\@tempb` and `\@tempc`. I have now replaced any use of these with `\@memtempa`, etc.

In this part we define a few commands that are used later on.

<code>\@ptsize</code>	The <code>\@ptsize</code> control sequence is normally used to store the second digit of the
<code>\@emptsize</code>	pointsize we are typesetting in. So, normally, it's value is one of 0, 1 or 2.
	<code>\@emptsize</code> stores the full pointsize.
	67 <code>\newcommand*{\@ptsize}{}</code>
	68 <code>\newcommand*{\@emptsize}{}</code>
	Any new lengths that depend on the point size option must be declared before
	the options are executed.
<code>\onelineskip</code>	The length <code>\onelineskip</code> is the vertical space taken by a normal line of text.
<code>\lxvchars</code>	The lengths <code>\lxvchars</code> and <code>\xlvchars</code> are the approximate lengths required for
<code>\xlvchars</code>	typesetting lines with either 65 or 45 characters.
	69 <code>\newlength{\onelineskip}</code>
	70 <code>\newlength{\lxvchars}</code>
	71 <code>\newlength{\xlvchars}</code>
<code>\@memcnta</code>	We need a scratch count register and a scratch counter.
<code>\c@memmarkcntra</code>	72 <code>\newcount\@memcnta</code>
	73 <code>\newcounter{\@memmarkcntra}</code>
<code>\if@restonecol</code>	When the document has to be printed in two columns, we sometimes have to
	temporarily switch to one column. This switch is used to remember to switch
	back.
	74 <code>\newif\if@restonecol</code>
<code>\if@openright</code>	This is TRUE if chapters are to start on righthand (recto) pages; this is the
	default. FALSE means chapters can start on any page.
	75 <code>\newif\if@openright</code>
	76 <code>\@openrighttrue</code>
	77
<code>\if@openleft</code>	This is TRUE if chapters are to start on lefthand (verso) pages.
	78 <code>\newif\if@openleft</code>
	79 <code>\@openleftfalse</code>
	80
<code>\if@mainmatter</code>	This is TRUE if the main part of the document is being currently procesed; this
	is the default.
	81 <code>\newif\if@mainmatter</code>
	82 <code>\@mainmattertrue</code>
	83
<code>\if@memoldfont</code>	This is TRUE if the <code>oldfontcommands</code> option is used.
	84 <code>\newif\if@memoldfont</code>
	85 <code>\@memoldfontfalse</code>
	86

`\ifextrafontsizes` This is TRUE if the `extrafontsizes` option is used.

```

87 \newif\ifextrafontsizes
88 \extrafontsizesfalse
89

```

`\@memerror` Two macros to save some space when reporting errors or warnings. The macros
`\@memwarn` take the same arguments, ignoring the first in each case, as `\ClassError` and
`\ClassWarning`, e.g.,
`\@memwarn{Message}`
instead of
`\ClassWarning{memoir}{Message}`.

```

90 \newcommand*{\@memerror}{\ClassError{memoir}}
91 \newcommand*{\@memwarn}{\ClassWarning{memoir}}
92

```

`\ifsamename` The macro `\nametest{⟨name1⟩}{⟨name2⟩}` tests whether the characters
`\nametest` forming the two arguments are the same or not. If they are the same then
`\ifsamename` is set TRUE, otherwise it is set FALSE. `⟨name1⟩` and `⟨name2⟩`
must both be either macro names (including the backslash) or must both not be
macro names. That is, you cannot do `\nametest{cs}{cs}`.

```

93 \newif\ifsamename
94 \newcommand{\nametest}[2]{%
95   \samenamefalse%
96   \begingroup%
97   \def\@memtempa{#1}\def\@memtempb{#2}%
98   \ifx \@memtempa\@memtempb%
99     \endgroup%
100    \samename true%
101  \else%
102    \endgroup%
103  \fi}
104

```

`\ifm@m@And` These are for ‘if A and B’, ‘if A or B’ and ‘if A xor B’.

`\m@m@Andtrue` For A and B:

```

\m@m@Andfalse \m@m@Andfalse
\ifm@m@Or \ifA
\m@m@Ortrue \ifB
\m@m@Orfalse \m@m@Andtrue
\ifm@m@Xor \fi
\m@m@Xortrue \fi
\m@m@Xorfalse

```

For A or B:

```

\m@m@Ortrue
\ifA
\else
\ifB

```

```

\else
\m@m@Orfalse
\fi
\fi

```

For A xor B:

```

\m@m@Xortrue
\ifA
\ifB
\m@m@Xfalse
\fi
\else
\ifB
\else
\m@m@Xorfalse
\fi
\fi

```

```

105 \newif\ifm@m@And
106 \newif\ifm@m@Or
107 \newif\ifm@m@Xor
108

```

`\kill@lastcounter` `\kill@lastcounter{<cnt>}`¹ kills the counter `<cnt>`. This macro is an extension to the suggestions as it also makes the last count register reusable. *The macro must only be used when the last allocated counter is to be killed.* For example with:

```

\newcounter{fred} \kill@lastcounter{fred} \newcounter{fred}

```

there is no error with the second `\newcounter`.

```

109 \newcommand{\kill@lastcounter}[1]{%

```

Deallocate the last counter register. From *The T_EXbook*, `\count10` is the number of the last register. The counter should be zeroed before being deallocated otherwise an immediately following `\newcount` may not be zero when allocated².

```

110 \count\count10 \z@
111 \advance\count10 \m@ne

```

Set the internal counter definition to `\relax`.

```

112 \expandafter\let\csname c@#1\endcsname\relax}
113

```

```

\@name@p@xdef Utility definition macros, along the lines of the kernel's \@namedef.
\@name@unresp@xdef 114 \newcommand{\@name@p@xdef}[1]{%
\@namelet 115 \expandafter\protected@xdef\csname #1\endcsname}
\@name@longdef

```

¹With thanks to Stefan Ulrich (ulrich@cis.uni-muenchen.de) who answered a question on this on CTT, 2001/07/09. (Also answered by Heiko Oberdiek, oberdiek@ruf.uni-freiburg.de).

²Discovered and fix provided by Robert Schlicht on 2005/08/31.

```

116 \newcommand{\@name@unresp@xdef}[1]{%
117   \expandafter\unrestored@protected@xdef\csname #1\endcsname}
118 \newcommand{\@namelet}[1]{%
119   \expandafter\let\csname #1\endcsname}
120 \newcommand{\@namelongdef}[1]{%
121   \long\expandafter\def\csname #1\endcsname}

\memletcmdtxt More utility macros (for the user but perhaps better not to tell them about it in
\memletttxttxt the manual)
\memletttxcmd \memletcmdtxt{\cmd}{txt} -> \let\cmd\txt
               \memletttxtxt{\txta}{txtb} -> \let\txta\txtb
               \memletttxcmd{\txt}{\cmd} -> \let\txt\cmd

122 \newcommand*{\memletcmdtxt}[2]{\expandafter\let\expandafter#1\csname#2\endcsname}
123 \newcommand*{\memletttxtxt}[1]{\expandafter\memletcmdtxt\csname#1\endcsname}
124 \newcommand*{\memletttxcmd}[2]{\expandafter\let\csname#1\endcsname#2}

\@nameedef A shorthand for using \protected@edef.
125 \newcommand{\@nameedef}[1]{%
126   \expandafter\protected@edef\csname #1\endcsname}
127

\memjustarg Utility macros that are not to be changed by any user!
\memgobble 128 \newcommand{\memjustarg}[1]{#1}
           129 \newcommand{\memgobble}[1]{}
           130

```

4.1 Classes, Packages and Files

```

\@memfakeusepackage \@memfakeusepackage{\pack} fools LATEX into thinking that the <pack.sty>
package has been loaded.
131 \newcommand*{\@memfakeusepackage}[1]{%
132   \@namelet{ver@#1.sty}\@empty}
133

```

```

\EmulatedPackage There was some discussion on CTT3 about how to prevent loading of a package.
\emulatedPackageWithOptions My response was to use \@memfakusepackage. Donald Arseneau came up with
this comprehensive solution (there was a little discussion about naming with my
final suggestion as here).
\EmulatedPackage{\package}[\date]
\EmulatedPackageWithOptions{\option-list}{\package}[\date]

134 \providecommand*\EmulatedPackage{}
135 \renewcommand*\EmulatedPackage}[1]{%
136   \@ifnextchar[{\@emulated@package{#1}}%
137     {\@emulated@package{#1}[\@empty]}%
138 }
139 \providecommand*\EmulatedPackageWithOptions{}

```

³ *Re: breakurl + pdfetex for generating .dvi*, March 2005.

```

140 \renewcommand*{\EmulatedPackageWithOptions}[2]{%
141   \PassOptionsToPackage{#1}{#2}%
142   \EmulatedPackage{#2}%
143 }

```

`\@emulated@package` The workhorse for the user commands.

```

144 \def\@emulated@package#1[#2]{%
145   \expandafter\xdef\csname ver@#1.\@pkgextension\endcsname{#2}%
146   \@ifundefined{opt@#1.\@pkgextension}%
147     {\@namedef{opt@#1.\@pkgextension}{}}{}%
148   \wlog{Package #1 \ifx\@empty#2\else[#2] \fi
149     \if,\csname opt@#1.\@pkgextension\endcsname,\else
150     (with options \csname opt@#1.\@pkgextension\endcsname) \fi
151     emulated by \@currname.}%
152 }
153 \@onlypreamble\EmulatedPackage
154 \@onlypreamble\EmulatedPackageWithOptions
155 \@onlypreamble\@emulated@package
156

```

`\DisemulatePackage` `\DisemulatePackage{<package>}` undoes a previous `\EmulatedPackage...`

```

157 \newcommand*{\DisemulatePackage}[1]{%
158   \@namelet{ver@#1.\@pkgextension}\relax}
159 \@onlypreamble\DisemulatePackage
160

```

Code, based on a hint from Morten Høgholm⁴ that the `scrfile` package from `koma-script` includes commands such as

`\AfterFile{<file>}{<code>}`

plus before and after classes and packages.

Extend the `\InputIfFileExists` macro to call hooks before and the actual inputting. The kernel version (as of 2005/11/21) is:

```

\newcommand{\InputIfFileExists}[2]{%
  \IfFileExists{#1}%
    {\#2\@addtofilelist{#1}\@input \@filef@und}}

```

`\InputIfFileExists` Effectively all file input is handled by this macro. Extend it by adding pre- and post- input hooks.

```

161 \renewcommand{\InputIfFileExists}[2]{%
162   \IfFileExists{#1}%
163     {\#2\@addtofilelist{#1}\m@matbeginf{#1}%
164       \@input \@filef@und
165       \m@matendf{#1}%
166       \killm@matf{#1}}}
167

```

⁴CTT posting 2005/11/08, *Re: Checking for packages from a class... revisited*

```

\m@matbeginf \m@matbeginf{<file>} calls macro \file-m@mfb if it is defined. Similarly
\m@matendf \m@matendf{<file>} calls \file-m@mfe if it is defined.
168 \newcommand{\m@matbeginf}[1]{\@ifundefined{#1-m@mfb}{}%
169   {\@nameuse{#1-m@mfb}}}
170 \newcommand{\m@matendf}[1]{\@ifundefined{#1-m@mfe}{}%
171   {\@nameuse{#1-m@mfe}}}
172

\killm@matf \killm@matf{<file>} undefines the \file-m@mfb and \file-m@mfe macros.
173 \newcommand*{\killm@matf}[1]{%
174   \@namelet{#1-m@mfb}\relax
175   \@namelet{#1-m@mfe}\relax}
176

\AtBeginFile \AtBeginFile{<file>}{<code>} inserts <code> just before <file> is input. Similarly
\AtEndFile \AtEndFile inserts just after input.
177 \newcommand{\AtBeginFile}[2]{\@ifundefined{#1-m@mfb}%
178   {\@namedef{#1-m@mfb}{#2}}%
179   {\expandafter\addtodef\csname #1-m@mfb\endcsname}{#2}}
180 \newcommand{\AtEndFile}[2]{\@ifundefined{#1-m@mfe}%
181   {\@namedef{#1-m@mfe}{#2}}%
182   {\expandafter\addtodef\csname #1-m@mfe\endcsname}{#2}}
183

\AtBeginPackage \AtBeginPackage{<pack>}{<code>} inserts <code> just before the <pack> package
\AtEndPackage is input, and \AtEndPackage is the equivalent for after input.
\RequireAtEndPackage \RequireAtEndPackage inserts <code> either at the end of <pack>, or
immediately if <pack> has already been input.
184 \newcommand{\AtBeginPackage}[2]{%
185   \AtBeginFile{#1.\@pkgextension}{#2}}
186 \newcommand{\AtEndPackage}[2]{%
187   \AtEndFile{#1.\@pkgextension}{#2}}
188 \newcommand{\RequireAtEndPackage}[2]{%
189   \@ifpackageloaded{#1}{#2}%
190   {\AtEndFile{#1.\@pkgextension}{#2}}}
191

\AtBeginClass \AtBeginClass{<class>}{<code>} and friends are the class equivalents of
\AtEndClass \AtBeginPackage and friends.
\RequireAtEndClass 192 \newcommand{\AtBeginClass}[2]{%
193   \AtBeginFile{#1.\@clsextension}{#2}}
194 \newcommand{\AtEndClass}[2]{%
195   \AtEndFile{#1.\@clsextension}{#2}}
196 \newcommand{\RequireAtEndClass}[2]{%
197   \@ifclassloaded{#1}{#2}%
198   {\AtEndFile{#1.\@clsextension}{#2}}}
199

```

`\phantomsection` A command needed if the `hyperref` package is used, for putting before certain `\addcontentsline` commands.

```
200 \newcommand{\phantomsection}{}
201
```

`\nofiles` The kernel's `\nofiles`, which surprisingly may be called *before* the class is loaded, lets `\makeindex` to `\relax`. This could cause problems⁵ with my initial version of `\makeindex`. Here's the kernel `\nofiles` definition:

```
\def\nofiles{%
  \@fileswfalse% flag for suppressing \immediate \writes
  \typeout{No auxiliary output files.^^J}%
  \long\def\protected@write##1##2##3%
    {\write\m@ne{}\if@nobreak\ifvmode\nobreak\fi\fi}%
  \let\makeindex\relax
  \let\makeglossary\relax}
```

To get it to work for when `\makeindex`'s optional argument is used I let `\makeindex` itself deal with `\nofiles` (and also `\makeglossary`).

```
202 \renewcommand*{\nofiles}{%
203   \@fileswfalse% flag for suppressing \immediate \writes
204   \typeout{No auxiliary output files.^^J}%
205   \long\def\protected@write##1##2##3%
206     {\write\m@ne{}\if@nobreak\ifvmode\nobreak\fi\fi}%
207   }
208
```

`\memsetcounter` A wrapper round `\setcounter`. I want to use this in the aux file.

```
209 \newcommand*{\memsetcounter}[2]{\setcounter{#1}{#2}}
210 \AtBeginDocument{\immediate\write\@mainaux{%
211   \string\providecommand*\string\memsetcounter}[2]{}}}
212
```

4.2 For package documentation

Some macros that may be useful for documenting LaTeX code. These have principally come from `doc.dtx`.

```
\bs Prints \
213 \def\bs{\texttt{\char'\}}}
```

`\l@nohyphenation` From `doc.dtx`

`\meta` `\meta{⟨arg⟩}` prints `⟨arg⟩`.

```
\meta@font@select 214 \ifx\l@nohyphenation\undefined
215   \newlanguage\l@nohyphenation
216 \fi
```

⁵As reported by Heiko Oberdiek on 2005/07/09.


```

217 \DeclareRobustCommand{\meta}[1]{%
218   \ensuremath\langle
219   \ifmmode \expandafter \nfss@text \fi
220   {%
221     \meta@font@select
222     \edef\meta@hyphen@restore
223       {\hyphenchar\the\font\the\hyphenchar\font}%
224     \hyphenchar\font\m@ne
225     \language\l@nohyphenation
226     #1\}%
227   \meta@hyphen@restore
228 } \ensuremath\rangle
229 }
230 \def\meta@font@select{\itshape}
231
\marg Robust versions of the doc.dtx macros.
\oarg \marg{<arg>} prints {<arg>}
\parg \oarg{<arg>} prints [<arg>]
\parg{<arg>} prints (<arg>)

232 \DeclareRobustCommand{\marg}[1]{%
233   {\ttfamily\char'\{} \meta{#1} {\ttfamily\char'\}}
234 \DeclareRobustCommand{\oarg}[1]{%
235   {\ttfamily\char'\[] \meta{#1} {\ttfamily\char'\}}
236 \DeclareRobustCommand{\parg}[1]{%
237   {\ttfamily\char'\() \meta{#1} {\ttfamily\char'\)}}

\cs \cs{arg} prints \arg.
238 \DeclareRobustCommand{\cs}[1]{\texttt{\char'\#1}}

\cmdprint From Heiko Oberdiek CTT 2001/05/26 (print and index a command)
\cmd \cmdprint{\fred} prints \fred
\cmd{\fred} prints and indexes \fred. NOTE It assumes that ? is the 'actual'
character for MakeIndex (it is normally @ but that is not much use if a command
includes @ as part of its name).

239 \newcommand{\cmdprint}[1]{\texttt{\string#1}}
240 \newcommand{\cmd}[1]{\cmdprint{#1}%
241   \index{\expandafter\@gobble\string#1?\string\cmdprint{\string#1}}}
242

```

4.3 Checking the processor

Prior to 2008/07/22 the `hyperref` package was modified in such a way as to produce warnings about versions of the `ifpdf` and `ifxetex` packages when `hyperref` was used with versions v1.618033, v1.61803, or v1.618 mempatch v4.9 of the class. `hyperref`'s author refused to make any changes to his package, insisting that emulating a package was 'ugly hacking'. He did, though, provide an example code that could be added to memoir to use a package if it was available.

I have used it for the emulations in this section. There will be no further changes to the class to accomodate any further changes to `hyperref` — it’s easy for a package to determine what class is being used and make arrangements accordingly, especially as there are not all that many classes. There are hundreds of packages that might be used with a particular class and it would be senseless for a class to attempt to take account of everything that might be used with it.

```

\ifpdf This can be used to check whether or not a document is being processed by
\pdftrue LATEX or pdfLATEX.
\pdffalse 243 \newif\ifm@mifpdf
244 \m@mifpdffalse
245 \IfFileExists{ifpdf.sty}{\RequirePackage{ifpdf}\relax}{%
246 \ClassWarningNoLine{memoir}{%
247 Package ‘ifpdf’ is not installed.\MessageBreak
248 The package is being emulated}%
249 \m@mifpdftrue
250 \expandafter\newif\csname ifpdf\endcsname
251 \pdffalse
252 \ifx\pdfoutput\undefined
253 \else
254 \ifx\pdfoutput\@undefined
255 \else
256 \ifx\pdfoutput\relax
257 \else
258 \ifnum\pdfoutput>0\relax
259 \pdftrue
260 \fi
261 \fi
262 \fi
263 \fi
264 %%\EmulatedPackage{ifpdf}[2008/07/23]
265 }
266

\ifetex Check if etex is being used. This is based on the check for pdf(latex).
\etextrue Someone may have created a package for this, or perhaps not.
\etexfalse 267 \newif\ifm@mifetex
268 \m@mifetexfalse
269 \IfFileExists{ifetex.sty}{\RequirePackage{ifetex}\relax}{%
270 \ClassInfo{memoir}{%
271 An ‘ifetex’ package is being emulated}%
272 \m@mifetextrue
273 \newif\ifetex
274 \etexfalse
275 \ifx\etexversion\undefined
276 \else
277 \ifx\etexversion\@undefined
278 \else
279 \ifx\etexversion\relax

```

```

280     \else
281         \ifnum\TeXversion>0\relax
282             \etexttrue
283         \fi
284     \fi
285 \fi
286 \fi
287 %%\EmulatedPackage{ifetex}[2008/07/23]
288 }
289

```

Having supplied an `\ifpdf` to check if pdfLaTeX is being used, here's one for XeTeX.

```

\ifxetex    Checks if XeTeX is being used.
\etexttrue 290 \newif\ifm@mifxetex
\etextfalse 291 \m@mifxetexfalse
\RequireXeTeX 292 \IfFileExists{ifxetex.sty}{\RequirePackage{ifxetex}\relax}{%
293     \ClassWarningNoLine{memoir}{%
294         The 'ifxetex' package is not installed.\MessageBreak
295         The package is being emulated}%
296 \expandafter\newif\csname ifxetex\endcsname
297 \@ifundefined{XeTeXrevision}{\etextfalse}{\etexttrue}
298 \m@mifxetexttrue%
    Check for XeTeX from the ifxetex package.
299 %%\EmulatedPackage{ifxetex}[2008/07/23]
300 }
301 \ifm@mifxetex%
302 % ifxetex package not found, emulate \RequireXeTeX
303 \def\RequireXeTeX{%
304     \ifxetex\else
305     \@memerror{XeTeX is required to process this document}%
306         {Try again with xelatex, not (pdf)latex.\MessageBreak
307         Or try removing any XeTeX package(s).}
308     \fi}
309 \fi
310

```

```

\ifluatex    Check if luatex is being used. This is based on the check for pdf(latex).
\luatextrue  Someone may have created a package for this, or perhaps not.
\luatexfalse 311 \newif\ifm@mifluatex
312     \m@mifluatexfalse
313 \IfFileExists{ifluatex.sty}{\RequirePackage{ifluatex}\relax}{%
314     \ClassWarningNoLine{memoir}{%
315         The 'ifluatex' package is not installed.\MessageBreak
316         The package is being emulated}%
317 \m@mifluatexttrue
318 \expandafter\newif\csname ifluatex\endcsname
319 \luatexfalse

```

```

320 \ifx\luatexversion\undefined
321 \else
322   \ifx\luatexversion\undefined
323   \else
324     \ifx\luatexversion\relax
325     \else
326       \luatextrue
327     \fi
328   \fi
329 \fi
330 %%\EmulatedPackage{ifluatex}[2008/07/23]
331 }
332

```

4.4 Extending an existing macro

My usual technique for modifying the code of an existing macro was to use `\renewcommand`. However it slowly became apparent that many of such modifications merely consisted of adding some code at either the start or end of the existing definition. In June 2000 I posed a question on the `comp.text.tex` newsgroup about this, and Michael Downes⁶ and Heiko Oberdiek⁷ were kind enough to respond. The responses arrived almost simultaneously, both supplying methods for adding code at the end of a macro defined to take one argument. Michael Downes' response included a remark about it probably being too much effort to formulate a general way of doing this. Heiko Oberdiek almost immediately replied with such a generalisation.

The following code, which I have extracted from a package that I wrote but which I did not submit to CTAN, incorporates the work of both Michael and Heiko. I have added a bit. All errors are my responsibility. Michael went on to write the `patchcmd` package which is a generalisation of the facilities below and so there was no need for my more restricted package.

`\wo@dmacro` Heiko Oberdiek suggested⁸ that $\langle macro \rangle$ should be tested for being the name of a macro, rather than using the normal L^AT_EX test against it being undefined. He supplied the code for performing the macro name test.

`\wo@dmacro` is a helper for handling the string 'macro':

```

333 \edef\wo@dmacro{%
334   \string m\string a\string c\string r\string o\string :%
335 }
336

```

`\wo@difmacro@begingroup` This seems to take two arguments:
`\wo@difmacro@begingroup{ $\langle macro \rangle$ }{ $\langle code \rangle$ }`, where $\langle macro \rangle$ should be the name of a defined macro and $\langle code \rangle$ is code to be executed if and only if $\langle macro \rangle$ is a defined macro.

⁶epsmjd@ams.org

⁷oberdiek@ruf.uni-freiburg.de

⁸Via email on 2000/07/12.

```

337 \def\wo@difmacro@begingroup#1{%
    Start a group.
338   \begingroup
    Test if #1 is a defined macro; \wo@dparsemacro does most of the work.
339   \edef\x{%
340     \noexpand\wo@dparsemacro\meaning#1\wo@dmacro\string -}%
341   }%
342   \x\@nil{#1}%
343 }
344

```

`\wo@dparsemacro` This is called by `\wo@difmacro@begingroup`. If $\langle macro \rangle$ is not a defined macro it prints a warning and closes the group begun by `\wo@difmacro@begingroup`. Otherwise it process the $\langle code \rangle$ which *must* close the group.

```

345 \begingroup
346   \edef\x{\endgroup
347     \def\noexpand\wo@dparsemacro##1\wo@dmacro##2\string -}%
348   \x#3\@nil#4{%
349     \ifx\#3\%
350       \endgroup
351       \@memwarn{\string '\string #4\string ' is not a macro}%
352       \expandafter\@gobble
353     \else
354       \expandafter\@firstofone
355     \fi
356 }
357

```

`\addtodef` The command `\addtodef{ $\langle macro \rangle$ }{ $\langle start-stuff \rangle$ }{ $\langle end-stuff \rangle$ }` adds stuff at the start and/or end of an argumentless macro. Initially the code for appending was `\addtodef*` a reimplementaion of the kernel `\g@addto@macro` command (from `ltxclass.dtx`). Later, at Michael Downes' suggestion, I combined appending and prepending stuff into a single command. Actually, there is no real need to have both `\addtodef` and `\addtodef*` as the body of the command being amended has no argument, but both versions are provided for consistency. The kernel `\@star@or@long` and `\l@ngrel@x` commands (in `ltxdefns.dtx`) are used to handle the potential `*` after the command name.

```

358 \def\addtodef{\@star@or@long\wo@daddtodef}
    The \@star@or@long command dealt with a possible * and now \wo@daddtodef
    does the work. It picks up the three arguments that the user thinks belong to
    \addtodef, namely  $\langle macro \rangle$ ,  $\langle start-stuff \rangle$  and  $\langle end-stuff \rangle$ .
359 \long\def\wo@daddtodef#1#2#3{%
    Check if  $\langle macro \rangle$  has been defined.
360   \wo@difmacro@begingroup{#1}{%

```

If $\langle macro \rangle$ is defined then store the tokens corresponding to the body of $\langle macro \rangle$ and the extra $\langle -stuff \rangle$ in token registers.

```
361 \temptokena{#2}%
362 \toks@ \expandafter{#1#3}%
```

Do an expanded definition for $\backslash x$, so that calling $\backslash x$ will $\backslash def$ a new version of $\langle macro \rangle$, whose body consists of the saved tokens (i.e., the original body plus the extra stuff). As Michael Downes noted, single letter control sequences do not take any of T_EX's hash table space. $\backslash longrel@x$ has been previously set to either $\backslash long$ or $\backslash relax$ by $\backslash @star@or@long$, and so may make the new definition of $\langle macro \rangle$ to be $\backslash long$.

```
363 \edef\x{\endgroup
364 \longrel@x\def\noexpand#1{\the\temptokena \the\toks@}}%
```

Finally, call $\backslash x$ to perform the new definition.

```
365 \x
366 }%
367 }
368
```

$\backslash addtoiargdef$ This adds stuff at the start and/or end of a macro that takes one argument. It is
 $\backslash addtoiargdef*$ a modification of Michael Downes' appending code⁹ which was:
 $\backslash wo@daddtoiargdef$

```
\def\appendef#1#2{%
  \begingroup
  \toks@ \expandafter{#1{##1}#2}%
  \edef\x{\endgroup \def\noexpand#1###1{\the\toks@}}%
  \x}
```

```
369 \def\addtoiargdef{\@star@or@long\wo@daddtoiargdef}
370 \long\def\wo@daddtoiargdef#1#2#3{%
371 \wo@difmacro@begingroup{#1}{%
372 \temptokena{#2}%
373 \toks@ \expandafter{#1{##1}#3}%
374 \edef\x{\endgroup
375 \longrel@x\def\noexpand#1###1{\the\temptokena \the\toks@}}%
376 \x
377 }%
378 }
379
```

For the record, Heiko Oberdiek produced the following version¹⁰ which avoids the use of a token register.

```
\newcommand{\appendiargdef}[2]{%
  \long\expandafter\def\expandafter#1\expandafter
```

⁹Posted to CTT on 15 June 2000,

URL: <http://www.dejanews.com/getdoc.xp?AN=635057844>.

¹⁰Posted to CTT on 16 June 2000,

URL: <http://www.dejanews.com/getdoc.xp?AN=635095381>.

```
##\expandafter1\expandafter{#1{##1}#2}}
```

Continuing the saga, Michael came up with a pretty general solution for modifying a macro with any number (up to 9) of arguments which he sent to me on 2000/07/13; this became the `patchcmd` package which is on CTAN. I'm sure that Michael would have given me permission to include it here if he had not tragically passed away in 2003. I miss his encouragement, advice, and skills.

```
\patchcmd \patchcmd{<macro>}{<start-stuff>}{<end-stuff>} inserts <start-stuff> at the
beginning of the definition of <macro> and <end-stuff> at the end of the
definition. <macro> can have 0–9 arguments and be defined by \newcommand
(and associates) or with \DeclareRobustCommand. For macros that use
\futurelet (e.g., those with starred forms or optional arguments) only
prepending works — any non-empty <end-stuff> will mess things up. \patchcmd
does not work with macros that have delimited arguments.

380 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Michael Downes' patchcmd 2000/07/31 v1.03 %%%%%%%%%%
381 \newcommand{\patchcommand}[1]{%
382   \expandafter\patchcmd@a\meaning#1?->@\@nil#1%
383 }

\patchcmd@a
384 \long\def\patchcmd@a#1#2#3->#4#5\@nil#6{%
385   \ifx @#4\relax \patchcmdError#6#1%
386   \expandafter\@gobbletwo % discard the other two arguments
387   \else
388     \if 1#2\toks@\{\patchcmd@e{}\#6}% 1 in this position means \long
389     \else \toks@\{\patchcmd@e*#6}%    not \long
390     \fi
391     \patchcmd@b #3@#4#5 ? ? ? \@nil#6%
392     \expandafter\the\expandafter\toks@
393     \fi}

\patchcmd@b
394 \def\patchcmd@b#1:#2@#3#4 #5#6 #7 #8\@nil#9{%
395   \if \ifx @#7@\expandafter
396     \ifx\cename #6\endcsname#9T\else F\fi\else F\fi T%
397     \toks@\expandafter{\expandafter\patchcommand\cename #6 \endcsname}%
398   \else
399     \ifx @#2@% No arguments
400       \toks@\expandafter{\the\toks@ 0}%
401     \else
402       \patchcmd@c 0#2{\string##}0%
403     \fi
404     \fi}

\patchcmd@c
405 \def\patchcmd@c#1#2#3{%
```

```

406 \if\string###2%      % yes it's a # token
407 \ifodd 0#31 % and it's followed by a number
408 \if 0#3\patchcmd@d#1\fi % number=0? then we're done
409 \else \patchcmd@d D% # not a number: must be a delimited arg
410 \fi
411 \else \patchcmd@d D% not a # token: must be a delimited arg
412 \fi
413 \patchcmd@c#3}

\patchcmd@d
414 \def\patchcmd@d#1{%
415 \if D#1%
416 %%% \PackageError{patchcmd}{Cannot change a macro that has
417 %%% delimited arguments}\@ehd
418 \@@memerror{%
419 Cannot change a macro that has delimited arguments}\@ehd}
420 \else
421 \toks@\expandafter{\the\toks@ #1}%
422 \fi
423 \begingroup
424 \aftergroup\@gobble
425 \let\patchcmd@c\endgroup}

\patchcmd@e
426 \def\patchcmd@e#1#2#3#4#5{%
427 \begingroup
428 \edef\@#1{%
429 \@temptokena\noexpand\expandafter{%
430 \noexpand#2%
431 \ifnum#3>0 {###1}\ifnum#3>1 {###2}\ifnum#3>2 {###3}%
432 \ifnum#3>3 {###4}\ifnum#3>4 {###5}\ifnum#3>5 {###6}%
433 \ifnum#3>6 {###7}\ifnum#3>7 {###8}\ifnum#3>8 {###9}%
434 \fi\fi\fi\fi\fi\fi\fi\fi\fi
435 ##1%
436 }%
437 }
438 \@{#5}%
439 \edef\@#1{\endgroup
440 \noexpand\renewcommand#1\noexpand#2\ifcase#3 \else [#3]\fi
441 {##1\the\@temptokena}}%
442 \@{#4}%
443 }

\patchcmdError
444 \long\def\patchcmdError#1#2{%
445 \begingroup
446 \toks@{Not redefinable}%
447 \ifcat\relax\noexpand#1% Is it a control sequence?
448 \begingroup

```



```

449 \let#1=?\ifx ?\relax % Is it "\relax"?
450 \endgroup % accept current value of \toks@
451 \else \endgroup
452 \if\ifx\relax#1u\else #2\fi u%
453 \toks@{Not defined}%
454 \fi
455 \fi
456 \fi
457 \edef\@{\endgroup
458 %\noexpand\PackageError{patchcmd}{%
459 %\the\toks@: \string#1}\noexpand\@ehd}%
460 \noexpand\@memerror{%
461 \the\toks@: \string#1}\noexpand\@ehd}%
462 \@}
463
464 %%%%%%%%% end of patchcmd code %%%%%%%%%
465

```

Prevent later loading of the original patchcmd package.

```

466 %\@memfakeusepackage{patchcmd}
467

```

4.5 Support for the bidi package (RTL typesetting)

The bidi package (system) enables bidirectional typesetting. As part of being able to accomplish this it often needs to exchange, for example, `\leftskip` and `\rightskip` depending on whether it is setting left-to-right (LTR) or right-to-left (RTL). In the case of memoir the initial bidi system implementation used a `bidimemoir` class developed by Vafa Khalighi. Now, with Vafa's help, the necessary changes have been incorporated in memoir; the `bidimemoir` class is no longer needed.

```

\memRTLleftskip These are the hooks required to support the bidi package.
\memRTLrightskip 468 \newcommand*{\memRTLleftskip}{\leftskip}
\memRTLvleftskip 469 \newcommand*{\memRTLrightskip}{\rightskip}
\memRTLvrightskip 470 \newcommand*{\memRTLvleftskip}{\leftskip}
\memRTLraggedright 471 \newcommand*{\memRTLvrightskip}{\rightskip}
\memRTLraggedleft 472 \newcommand*{\memRTLraggedright}{\raggedright}
473 \newcommand*{\memRTLraggedleft}{\raggedleft}
474

```

It is expected that the bidi package will redefine these as:

```

\renewcommand*{\memRTLleftskip}{\if@RTL\rightskip\else\leftskip\fi}
\renewcommand*{\memRTLrightskip}{\if@RTL\leftskip\else\rightskip\fi}
\renewcommand*{\memRTLvleftskip}{\if@RTL\vrightskip\else\vleftskip\fi}
\renewcommand*{\memRTLvrightskip}{\if@RTL\vleftskip\else\vrightskip\fi}
\renewcommand*{\memRTLraggedright}{\if@RTL\raggedleft\else\raggedright\fi}
\renewcommand*{\memRTLraggedleft}{\if@RTL\raggedright\else\raggedleft\fi}

```

5 Declaration of Options

5.1 Setting Paper Sizes

The variables `\paperheight` and `\paperwidth` should reflect the physical paper size after trimming.

Option `letterpaper` will be the default.

`\stockheight` `\stockwidth` The lengths `\stockheight` and `\stockwidth` should be the height and width of the stock sheet before trimming. For example, this is the physical size of a single sheet that might be laser-printed. The lengths `\trimtop` and `\trimedge` are the amount that will be trimmed off the top and fore edge of the physical sheet. For desk printer output the size of the trimmed sheet is often the same as the physical sheet. In other words the `\paper...` and `\stock...` sizes are the same and the trims are zero. This class assumes that this is the normal case.

```
475 \newlength{\stockheight}
476 \newlength{\stockwidth}
477 \newlength{\trimtop}
478 \newlength{\trimedge}
479
```

The class provides an extended range of stock sizes. It may be useful at some point to have macros for these.

```
\stockdbill US stock sizes.
\stockstatement 480 \newcommand*\stockdbill {\stockheight=7in \stockwidth=3in}
\stockexecutive 481 \newcommand*\stockstatement {\stockheight=8.5in \stockwidth=5.5in}
\stockletter 482 \newcommand*\stockexecutive {\stockheight=10.5in \stockwidth=7.25in}
\stockold 483 \newcommand*\stockletter {\stockheight=11in \stockwidth=8.5in}
\stocklegal 484 \newcommand*\stockold {\stockheight=12in \stockwidth=9in}
\stockledger 485 \newcommand*\stocklegal {\stockheight=14in \stockwidth=8.5in}
\stockbroadsheet 486 \newcommand*\stockledger {\stockheight=17in \stockwidth=11in}
487 \newcommand*\stockbroadsheet {\stockheight=22in \stockwidth=17in}
488

\stockpottvo Traditional British octavo sizes.
\stockfoolscapvo 489 \newcommand*\stockpottvo {\stockheight=6.25in \stockwidth=4in}
\stockcrownvo 490 \newcommand*\stockfoolscapvo {\stockheight=6.75in \stockwidth=4.25in}
\stockpostvo 491 \newcommand*\stockcrownvo {\stockheight=7.5in \stockwidth=5in}
\stocklargecrownvo 492 \newcommand*\stockpostvo {\stockheight=8in \stockwidth=5in}
\stocklargepostvo 493 \newcommand*\stocklargecrownvo {\stockheight=8in \stockwidth=5.25in}
\stocksmalldemyvo 494 \newcommand*\stocklargepostvo {\stockheight=8.25in \stockwidth=5.25in}
495 \newcommand*\stocksmalldemyvo {\stockheight=8.5in \stockwidth=5.675in}

\stockdemyvo
\stockmediumvo 496 \newcommand*\stockdemyvo {\stockheight=8.75in \stockwidth=5.675in}
\stocksmallroyalvo 497 \newcommand*\stockmediumvo {\stockheight=9in \stockwidth=5.75in}
\stockroyalvo 498 \newcommand*\stocksmallroyalvo {\stockheight=9.25in \stockwidth=6.175in}
\stocksuperroyalvo 499 \newcommand*\stockroyalvo {\stockheight=10in \stockwidth=6.25in}
\stockimperialvo
```

```

500 \newcommand*{\stocksuperroyalvo}{\stockheight=10.25in \stockwidth=6.75in}
501 \newcommand*{\stockimperialvo} {\stockheight=11in \stockwidth=7.5in}
502

```

```

\stockmcrownvo Metric stock sizes.
\stockmlargecrownvo 503 \newcommand*{\stockmcrownvo} {\stockheight=186mm \stockwidth=123mm}
\stockmdemyvo 504 \newcommand*{\stockmlargecrownvo} {\stockheight=198mm \stockwidth=129mm}
\stocksmallroyalvo 505 \newcommand*{\stockmdemyvo} {\stockheight=216mm \stockwidth=138mm}
506 \newcommand*{\stocksmallroyalvo} {\stockheight=234mm \stockwidth=156mm}
507

```

```

\stockao The A series, A0 — A6.
\stockai 508 \newcommand*{\stockao} {\stockheight=1189mm \stockwidth=841mm}
\stockaii 509 \newcommand*{\stockai} {\stockheight=841mm \stockwidth=594mm}
\stockaiiii 510 \newcommand*{\stockaii} {\stockheight=594mm \stockwidth=420mm}
\stockaiiv 511 \newcommand*{\stockaiiii} {\stockheight=420mm \stockwidth=297mm}
\stockav 512 \newcommand*{\stockaiiv} {\stockheight=297mm \stockwidth=210mm}
\stockavi 513 \newcommand*{\stockav} {\stockheight=210mm \stockwidth=148mm}
514 \newcommand*{\stockavi} {\stockheight=148mm \stockwidth=105mm}
515

```

```

\stockbo The B series, B0 — B6.
\stockbi 516 \newcommand*{\stockbo} {\stockheight=1414mm \stockwidth=1000mm}
\stockbii 517 \newcommand*{\stockbi} {\stockheight=1000mm \stockwidth=707mm}
\stockbiii 518 \newcommand*{\stockbii} {\stockheight=707mm \stockwidth=500mm}
\stockbiv 519 \newcommand*{\stockbiii} {\stockheight=500mm \stockwidth=353mm}
\stockbv 520 \newcommand*{\stockbiv} {\stockheight=353mm \stockwidth=250mm}
\stockbvi 521 \newcommand*{\stockbv} {\stockheight=250mm \stockwidth=176mm}
522 \newcommand*{\stockbvi} {\stockheight=176mm \stockwidth=125mm}
523

```

It will be useful also to provide macros that set the page sizes.

```

\pagedbill S page sizes.
\pagestatement 524 \newcommand*{\pagedbill} {\paperheight=7in \paperwidth=3in}
\pageexecutive 525 \newcommand*{\pagestatement} {\paperheight=8.5in \paperwidth=5.5in}
\pageletter 526 \newcommand*{\pageexecutive} {\paperheight=10.5in \paperwidth=7.25in}
\pageold 527 \newcommand*{\pageletter} {\paperheight=11in \paperwidth=8.5in}
\pagelegal 528 \newcommand*{\pageold} {\paperheight=12in \paperwidth=9in}
\pageledger 529 \newcommand*{\pagelegal} {\paperheight=14in \paperwidth=8.5in}
\pagebroadsheet 530 \newcommand*{\pageledger} {\paperheight=17in \paperwidth=11in}
531 \newcommand*{\pagebroadsheet} {\paperheight=22in \paperwidth=17in}
532

```

```

\pagepottvo British traditional page sizes, octavo.
\pagefoolscapvo 533 \newcommand*{\pagepottvo} {\paperheight=6.25in \paperwidth=4in}
\pagecrownvo 534 \newcommand*{\pagefoolscapvo} {\paperheight=6.75in \paperwidth=4.25in}
\pagepostvo 535 \newcommand*{\pagecrownvo} {\paperheight=7.5in \paperwidth=5in}
\pagelargecrownvo 536 \newcommand*{\pagepostvo} {\paperheight=8in \paperwidth=5in}
\pagelargepostvo
\pagesmalldemyvo

```

```

537 \newcommand*{\pagelargecrownvo}{\paperheight=8in \paperwidth=5.25in}
538 \newcommand*{\pagelargepostvo}{\paperheight=8.25in \paperwidth=5.25in}
539 \newcommand*{\pagesmalldemyvo}{\paperheight=8.5in \paperwidth=5.675in}

\pagedemyvo
\pagemediumvo 540 \newcommand*{\pagedemyvo}{\paperheight=8.75in \paperwidth=5.675in}
\pagesmallroyalvo 541 \newcommand*{\pagemediumvo}{\paperheight=9in \paperwidth=5.75in}
\pageroyalvo 542 \newcommand*{\pagesmallroyalvo}{\paperheight=9.25in \paperwidth=6.175in}
\pagesuperroyalvo 543 \newcommand*{\pageroyalvo}{\paperheight=10in \paperwidth=6.25in}
\pageimperialvo 544 \newcommand*{\pagesuperroyalvo}{\paperheight=10.25in \paperwidth=6.75in}
545 \newcommand*{\pageimperialvo}{\paperheight=11in \paperwidth=7.5in}
546

\pagemcrownvo Metric sizes.
\pagemlargecrownvo 547 \newcommand*{\pagemcrownvo}{\paperheight=186mm \paperwidth=123mm}
\pagemdemyvo 548 \newcommand*{\pagemlargecrownvo}{\paperheight=198mm \paperwidth=129mm}
\pagesmallroyalvo 549 \newcommand*{\pagemdemyvo}{\paperheight=216mm \paperwidth=138mm}
550 \newcommand*{\pagesmallroyalvo}{\paperheight=234mm \paperwidth=156mm}
551

\pageao The A series, A0 — A6.
\pageai 552 \newcommand*{\pageao}{\paperheight=1189mm \paperwidth=841mm}
\pageaai 553 \newcommand*{\pageai}{\paperheight=841mm \paperwidth=594mm}
\pageaiii 554 \newcommand*{\pageaai}{\paperheight=594mm \paperwidth=420mm}
\pageaiv 555 \newcommand*{\pageaiii}{\paperheight=420mm \paperwidth=297mm}
\pageav 556 \newcommand*{\pageaiv}{\paperheight=297mm \paperwidth=210mm}
\pageavi 557 \newcommand*{\pageav}{\paperheight=210mm \paperwidth=148mm}
558 \newcommand*{\pageavi}{\paperheight=148mm \paperwidth=105mm}
559

\pagebo The B series, B0 — B6.
\pagebi 560 \newcommand*{\pagebo}{\paperheight=1414mm \paperwidth=1000mm}
\pagebii 561 \newcommand*{\pagebi}{\paperheight=1000mm \paperwidth=707mm}
\pagebiii 562 \newcommand*{\pagebii}{\paperheight=707mm \paperwidth=500mm}
\pagebiv 563 \newcommand*{\pagebiii}{\paperheight=500mm \paperwidth=353mm}
\pagebv 564 \newcommand*{\pagebiv}{\paperheight=353mm \paperwidth=250mm}
\pagebvi 565 \newcommand*{\pagebv}{\paperheight=250mm \paperwidth=176mm}
566 \newcommand*{\pagebvi}{\paperheight=176mm \paperwidth=125mm}
567

Declare the stock size options.
Metric paper stock sizes.
568 \DeclareOption{a0paper}{\stockao}
569 \DeclareOption{a1paper}{\stockai}
570 \DeclareOption{a2paper}{\stockaii}
571 \DeclareOption{a3paper}{\stockaiii}
572 \DeclareOption{a4paper}{\stockaiv}
573 \DeclareOption{a5paper}{\stockav}
574 \DeclareOption{a6paper}{\stockavi}

```

```

575 \DeclareOption{b0paper}{\stockbo}
576 \DeclareOption{b1paper}{\stockbi}
577 \DeclareOption{b2paper}{\stockbii}
578 \DeclareOption{b3paper}{\stockbiii}
579 \DeclareOption{b4paper}{\stockbiv}
580 \DeclareOption{b5paper}{\stockbv}
581 \DeclareOption{b6paper}{\stockbvi}
582 \DeclareOption{mcrownvopaper}{\stockmcrownvo}
583 \DeclareOption{mlargecrownvopaper}{\stockmlargecrownvo}
584 \DeclareOption{mdemyvopaper}{\stockmdemyvo}
585 \DeclareOption{msmallroyalvopaper}{\stockmsmallroyalvo}
586

```

US paper stock sizes.

```

587 \DeclareOption{dbillpaper}{\stockdbill}
588 \DeclareOption{statementpaper}{\stockstatement}
589 \DeclareOption{executivepaper}{\stockexecutive}
590 \DeclareOption{letterpaper}{\stockletter}
591 \DeclareOption{oldpaper}{\stockold}
592 \DeclareOption{legalpaper}{\stocklegal}
593 \DeclareOption{ledgerpaper}{\stockledger}
594 \DeclareOption{broadsheetpaper}{\stockbroadsheet}
595

```

British octavo stock paper sizes.

```

596 \DeclareOption{pottvopaper}{\stockpottvo}
597 \DeclareOption{foolscapvopaper}{\stockfoolscapvo}
598 \DeclareOption{crownvopaper}{\stockcrownvo}
599 \DeclareOption{postvopaper}{\stockpostvo}
600 \DeclareOption{largecrownvopaper}{\stocklargecrownvo}
601 \DeclareOption{largepostvopaper}{\stocklargepostvo}
602 \DeclareOption{smalldemyvopaper}{\stocksmalldemyvo}
603 \DeclareOption{demyvopaper}{\stockdemyvo}
604 \DeclareOption{mediumvopaper}{\stockmediumvo}
605 \DeclareOption{smallroyalvopaper}{\stocksmallroyalvo}
606 \DeclareOption{royalvopaper}{\stockroyalvo}
607 \DeclareOption{superroyalvopaper}{\stocksuperroyalvo}
608 \DeclareOption{imperialvopaper}{\stockimperialvo}
609

```

Ebook.

```

610 \DeclareOption{ebook}
611   {\setlength\stockheight {9in}%
612    \setlength\stockwidth  {6in}}
613

```

The landscape option switches the values of the height and width, assuming that the dimensions were originally given for portrait orientation. (At the suggestion of Wilhelm Müller made this independent of the option sequence). .

```

\ifmemlandscape
\memlandscape true
\memlandscape false

```

```

614 \newif\ifmemlandscape
615 \memlandscapefalse

616 \DeclareOption{landscape}{\memlandscapetrue}
617 \DeclareOption{portrait}{\memlandscapefalse}
618

```

5.2 Choosing the type size

The type size options are handled by defining `\@ptsize` to contain the last digit of the size in question and branching on `\ifcase` statements. This is done for historical reasons to stay compatible with other packages that use the `\@ptsize` variable to select special actions. It makes the declarations of size options less than 10pt or more than 20pt difficult. In this class 9 is used for the 9pt option, assuming that the class will never define a 19pt option. For larger options the full complement of digits are used.

In any event, `\@memptsize` holds the complete size.

Option 10pt will be the default.

```

619 \renewcommand*{\@ptsize}{0}
620 \renewcommand*{\@memptsize}{10}
621 \DeclareOption{9pt}{\renewcommand*{\@ptsize}{9}\renewcommand*{\@memptsize}{9}}
622 \DeclareOption{10pt}{\renewcommand*{\@ptsize}{0}\renewcommand*{\@memptsize}{10}}
623 \DeclareOption{11pt}{\renewcommand*{\@ptsize}{1}\renewcommand*{\@memptsize}{11}}
624 \DeclareOption{12pt}{\renewcommand*{\@ptsize}{2}\renewcommand*{\@memptsize}{12}}
625 \DeclareOption{14pt}{\renewcommand*{\@ptsize}{4}\renewcommand*{\@memptsize}{14}}
626 \DeclareOption{17pt}{\renewcommand*{\@ptsize}{7}\renewcommand*{\@memptsize}{17}}
627 \DeclareOption{20pt}{\renewcommand*{\@ptsize}{20}\renewcommand*{\@memptsize}{20}}
628 \DeclareOption{25pt}{\renewcommand*{\@ptsize}{25}\renewcommand*{\@memptsize}{25}}
629 \DeclareOption{30pt}{\renewcommand*{\@ptsize}{30}\renewcommand*{\@memptsize}{30}}
630 \DeclareOption{36pt}{\renewcommand*{\@ptsize}{36}\renewcommand*{\@memptsize}{36}}
631 \DeclareOption{48pt}{\renewcommand*{\@ptsize}{48}\renewcommand*{\@memptsize}{48}}
632 \DeclareOption{60pt}{\renewcommand*{\@ptsize}{60}\renewcommand*{\@memptsize}{60}}
633

```

`\if@nyptsizeopt` For any point size:
`\@nyptsizeopttrue` 634 \newif\if@nyptsizeopt
`\@nyptsizeoptfalse` 635 \@nyptsizeoptfalse
`\anyptfilebase` 636 \providecommand*{\anyptfilebase}{mem}
`\anyptsize` 637 \providecommand*{\anyptsize}{10}
638 \DeclareOption{*pt}{\@nyptsizeopttrue}
639

5.3 Two-side or one-side printing

For two-sided printing we use the switch `\if@twoside`. In addition we have to set the `\if@mparswitch` to get any margin paragraphs into the outside margin. The default is `twoside`.

```

\if@twoside
\if@mparswitch 640 \DeclareOption{twoside}{\@twosidetrue \@mparswitchtrue}
641 \DeclareOption{oneside}{\@twosidefalse \@mparswitchfalse}

```

5.4 Two column printing

Two-column and one-column printing is again realized via a switch which is defined in the kernel. The default is single column printing.

```

\if@twocolumn
642 \DeclareOption{onecolumn}{\@twocolumnfalse}
643 \DeclareOption{twocolumn}{\@twocolumntrue}

```

5.5 The draft option

If the user requests draft we show any overfull boxes, marginal notes are allowed, and any copyright notices are not printed. For symmetry, we also define a `final` option which is the default.

The user can use the `\ifdraftdoc` flag to add additional effects:

```
\ifdraftdoc <additional code> \fi
```

```

\ifdraftdoc
644 \newif\ifdraftdoc\draftdocfalse
645 \setlength{\overfullrule}{\z@}
646 \DeclareOption{final}{\setlength{\overfullrule}{\z@}
647 \draftdocfalse
648 \msdocfalse}
649 \DeclareOption{draft}{\setlength{\overfullrule}{5pt}%
650 \draftdoctrue
651 \msdocfalse}

```

5.6 The ms option

`\ifmsdoc` The `ms` option makes the document look as though it was produced on a typewriter. We use a flag for remembering this. The user may also use the flag for specifying `ms` effects.

```

652 \newif\ifmsdoc
653 \msdocfalse
654 \DeclareOption{ms}{%
655 \msdoctrue
656 \draftdocfalse
657 \setlength{\overfullrule}{\z@}
658 }
659

```

5.7 The showtrims option

The `showtrims` option will display crosses at the corners of the logical pages showing where the stock should be trimmed.

```
\ifshowtrims
660 \newif\ifshowtrims
661   \showtrimsfalse
662 \DeclareOption{showtrims}{\showtrimstrue}
663
```

5.8 The article option

The `article` option typesets as a simulation of the `article` class.

```
\ifartopt   \ifartopt is a flag (TRUE) if the article option is called.
664 \newif\ifartopt
665   \artoptfalse
666 \DeclareOption{article}{\artopttrue}
667
```

5.9 The subfigure option

The `subfigure` option has been made a no-op since version 1.1 and finally removed in version 1.61803. It was:

```
\DeclareOption{subfigure}{%
  \ClassWarningNoLine{memoir}{The subfigure option is not required}}
```

5.10 The openright, openleft and openany options

The `openright` option specifies that Chapters must begin on recto pages. The `openleft` option specifies that Chapters must begin on verso pages and the `openany` option lets Chapters start on any page.

```
668 \DeclareOption{openright}{\@openrighttrue}
669 \DeclareOption{openany}{\@openrightfalse}
670 \DeclareOption{openleft}{\@openlefttrue}

\openright  Commands that can be used to change the option in the middle of the
\openany    document11.
\openleft  671 \newcommand{\openright}{\@openrighttrue\@openleftfalse%
672   \gdef\clearforchapter{\cleartorecto}}
673 \newcommand{\openany}{\@openrightfalse\@openleftfalse%
674   \gdef\clearforchapter{\clearpage}}
675 \newcommand{\openleft}{\@openlefttrue
676   \gdef\clearforchapter{\cleartoverso}}
677
```

¹¹Openleft provided to meet a request by Vladimir G. Ivanovic (vladimir@acm.org) in September 2001.

5.11 Equation numbering on the left

The `leqno` option prints equation numbers on the left. This is implemented via an external class option file.

```
678 \DeclareOption{leqno}{\input{leqno.clo}}
```

5.12 Flush left math displays

The `fleqn` option redefines the displayed math environments so that they are left adjusted with an indent of `\mathindent` from the current left margin. This is implemented via an external class option file.

```
679 \DeclareOption{fleqn}{\input{fleqn.clo}}
```

5.13 Open bibliography

The `openbib` option redefines the `thebibliography` so that each block starts on a new line, and succeeding lines in a block are indented by `\bibindent`.

```
680 \DeclareOption{openbib}{%
681   \AtEndOfClass{%
682     \renewcommand\@openbib@code{%
683       \advance\leftmargin\bibindent
684       \itemindent -\bibindent
685       \listparindent \itemindent
686       \parsep \z@
687     }%
688     \renewcommand\newblock{\par}}
689
```

5.14 Old font commands

The `oldfontcommands` option enables commands like `\bf` and friends.

```
690 \DeclareOption{oldfontcommands}{\@memoldfonttrue}
```

5.15 Extra font sizes

The `extrafontsizes` options indicates that extended font sizes (above 25pt) are available.

```
691 \DeclareOption{extrafontsizes}{\extrafontsizestrue}
692
```

5.16 ETeX

If found we automatically load the eTeX package right after executing the class options. Though, some odd L^AT_EX installations may have the eTeX package, but is not based on eTeX (eventhough that is what the LaTeX-project have recommended for several years now), so we add a dead mans switch to disable the loading.

```

\ifmem@noetex
\mem@noetextrue 693 \newif\ifmem@noetex
\mem@noetexfalse 694 \mem@noetexfalse
695 \DeclareOption{noetex}{\mem@noetextrue}
696

```

6 Executing Options

Here we execute the default options to initialize certain variables.

```

697 \ExecuteOptions{final,letterpaper,10pt,onecolumn,openright,twoside,
698 portrait}

```

The `\ProcessOptions` command causes the execution of the code for every option `foo` which is declared and for which the user typed the `foo` option in his `\documentclass` command. For every option `bar` he typed, which is not declared, the option is assumed to be a global option. All options will be passed as document options to any `\usepackage` command in the document preamble. `\ProcessOptions*` processes the options in the order they are given in the `\documentclass` command, instead of the definition order.

```

699 \ProcessOptions*
700 \ifmemlandscape
701 \setlength\@tempdima {\stockheight}
702 \setlength\stockheight{\stockwidth}
703 \setlength\stockwidth {\@tempdima}
704 \fi
705

```

Load the eTeX package if found, and if the user has not explicitly asked us not to.

```

706 \ifmem@noetex\relax\else
707 \IfFileExists{etex.sty}{\RequirePackage{etex}}{}
708 \fi

```

`\memoirpostopthook` A user could define this *before* the `\documentclass` command to do something at this point. For example, to set up a new stock size:

```

\def\memoirpostopthook{\stockheight=44in \stockwidth=34in}
709 \providecommand*\memoirpostopthook{}
710 \memoirpostopthook
711

```

`\cleartorecto` A repeat of `\cleardoublepage`; clears to a recto (odd-numbered) page.

```

712 \def\cleartorecto{\clearpage\if@twoside \ifodd\c@page\else
713 \hbox{}\thispagestyle{cleared}%
714 \newpage\if@twocolumn\hbox{}\newpage\fi\fi\fi}
715

```

`\cleartoverso` Clears to a verso (even-numbered) page.

```

716 \def\cleartoverso{\clearpage\if@twoside
717 \ifodd\c@page\hbox{}\thispagestyle{cleared}%

```

```

718 \newpage\if@twocolumn\hbox{}\newpage\fi\fi\fi}
719

```

Set `\clearforchapter` according to the options.

```

720 \if@openleft
721 \openleft
722 \else
723 \if@openright
724 \openright
725 \else
726 \openany
727 \fi
728 \fi
729

```

```

\@ivpt
\@xxxpt 730 \newcommand*\@ivpt}{4}
\@xxxvipt 731 \newcommand*\@xxxpt}{30}
\@xlvipt 732 \newcommand*\@xxxvipt}{36}
\@lxpt 733 \newcommand*\@xlvipt}{48}
\@lxxipt 734 \newcommand*\@lxpt}{60}
\@lxxxivpt 735 \newcommand*\@lxxipt}{72}
\@xcvipt 736 \newcommand*\@lxxxivpt}{84}
\@cviipt 737 \newcommand*\@xcvipt}{96}
\@cxipt 738 \newcommand*\@cviipt}{108}
\@cxxxpt 739 \newcommand*\@cxipt}{120}
\@cxxxiipt 740 \newcommand*\@cxxxpt}{132}
741

```

Now that all the options have been executed we can load the chosen class option file that contains all size dependent code. The larger sizes are only made available for the `extrafontsizes` option.

```

\memfontfamily The default font and coding as set by the kernel is cmr and OT1 respectively,
\memfontenc giving Knuth's original Computer Modern Roman font at a set of fixed sizes
\memfontpack (maximum of 24.88pt). If we are to have any size font we have to go to a
scaleable font. As the Latin Modern font is a scaleable version of Computer
Modern, and all modern LaTeX distributions include it, I am using this by
default.

```

By defining these three macros *before* the `\documentclass` any other font and package can be used.

```

742 \providecommand*\memfontfamily}{lmr}
743 \providecommand*\memfontenc}{T1}
744 \providecommand*\memfontpack}{lmodern}
745

```

Deal with the `'anyptsize' (*pt)` option first. In this case the macros `\anyptfilebase` and `\anyptsize` should have been defined by the author *before* the `\documentclass` command, and a file

`\anyptfilebase\anyptsize.clo` should exist (if not the file will default to `mem10.clo`).

```

746 \if@nyptsizeopt
747   \newcommand*{\@nyptclofile}{\anyptfilebase\anyptsize.clo}
748   \IfFileExists{\@nyptclofile}{\def\@memptsize{\anyptsize}}{%
749     \@memerror{You have used the ‘*pt’ option but \MessageBreak
750               file \@nyptclofile\space can’t be found}%
751     {I’ll use mem10.clo instead}
752   \renewcommand*{\@nyptclofile}{mem10.clo}%
753   \def\@memptsize{10}%
754 }
755 \renewcommand*{\@ptsize}{\@memptsize}
756 \usefont{\memfontenc}{\memfontfamily}{m}{n}
757 \input{\@nyptclofile}
758 \usepackage{\memfontpack}\usepackage[\memfontenc]{fontenc}
759 \else
  Now for the fixed pt size options.
760   \ifextrafontsizes
761     \usefont{\memfontenc}{\memfontfamily}{m}{n}
762     \input{mem\@memptsize.clo}
763     \usepackage{\memfontpack}\usepackage[\memfontenc]{fontenc}
764   \else
765     \ifnum\@memptsize > 17\relax
766       \@memerror{The ‘extrafontsizes’ option is required to use \MessageBreak
767                 the ‘\@memptsize pt’ option}%
768       {The 17pt option will be used instead}
769       \input{mem17.clo}
770     \else
771       \ifnum\@ptsize = 9\relax
772         \input{mem\@ptsize.clo}
773       \else
774         \input{mem1\@ptsize.clo}
775       \fi
776     \fi
777   \fi
778 \fi
779
780 \end{class}

```

7 Fonts and spaces

In this section we deal with most of the aspects that are related to font sizes, and spacing that is related to the size of the body font.

7.1 Fonts

L^AT_EX offers the user commands to change the size of the font, relative to the ‘main’ size. Each relative size changing command `\size` executes the command

Table 1: Shorthand font point size commands

				<code>\@ivpt</code>	4
<code>\@vpt</code>	5	<code>\@vipt</code>	6	<code>\@viipt</code>	7
<code>\@viipt</code>	8	<code>\@ixpt</code>	9	<code>\@xpt</code>	10
<code>\@xipt</code>	10.95	<code>\@xiipt</code>	12	<code>\@xivpt</code>	14.4
<code>\@xviipt</code>	17.28	<code>\@xxpt</code>	20.74	<code>\@xxvpt</code>	24.88
<code>\@xxxpt</code>	30	<code>\@xxxvipt</code>	36	<code>\@xlvipt</code>	48
<code>\@lxpt</code>	60	<code>\@lxxiipt</code>	72	<code>\@lxxxivpt</code>	84
<code>\@xcvipt</code>	96	<code>\@cviipt</code>	108	<code>\@cxxpt</code>	120
<code>\@cxxxiipt</code>	132				

`\@setfontsize\size` *<font-size>* *<baselineskip>* where:

<font-size> The absolute size of the font to use from now on.

<baselineskip> The normal value of `\baselineskip` for the size of the font selected. (The actual value will be `\baselinestretch * <baselineskip>`.)

A number of commands, defined in the L^AT_EX kernel, shorten the following definitions and are used throughout. They are listed in the first part of Table 1. Those in the second part are introduced by the class.

```

\normalsize The user level command for the main size is \normalsize. Internally LATEX uses
\@normalsize \@normalsize when it refers to the main size. \@normalsize will be defined to
work like \normalsize if the latter is redefined from its default definition (that
just issues an error message). Otherwise, in the standard classes \@normalsize
simply selects a 10pt/12pt size, but here it selects among the wider range.
The \normalsize macro also sets new values for
\abovedisplayskip, \abovedisplayshortskip and \belowdisplayshortskip.
For the larger sizes I have set \abovedisplayskip to the font size and
\belowdisplayshortskip to 0.5\onelineskip.
781 <*9pt | 10pt | 11pt | 12pt | 14pt | 17pt | 20pt | 25pt | 30pt | 36pt | 48pt | 60pt>
782 \renewcommand{\normalsize}{%
783 <*9pt>
784   \@setfontsize\normalsize\@ixpt\@xpt
785   \abovedisplayskip 9\p@ \@plus 2\p@ \@minus 4.5\p@
786   \abovedisplayshortskip \z@ \@plus 3\p@
787   \belowdisplayshortskip 5.5\p@ \@plus 2.5\p@ \@minus 3\p@
788 </9pt>
789 <*10pt>
790   \@setfontsize\normalsize\@xpt\@xipt
791   \abovedisplayskip 10\p@ \@plus 2\p@ \@minus 5\p@
792   \abovedisplayshortskip \z@ \@plus 3\p@
793   \belowdisplayshortskip 6\p@ \@plus 3\p@ \@minus 3\p@
794 </10pt>
795 <*11pt>

```

```

796 \setfontsize\normalsize\@xipt{13.6}%
797 \abovedisplayskip 11\p@ \@plus3\p@ \@minus6\p@
798 \abovedisplayshortskip \z@ \@plus3\p@
799 \belowdisplayshortskip 6.5\p@ \@plus3.5\p@ \@minus3\p@
800 (/11pt)
801 (*12pt)
802 \setfontsize\normalsize\@xipt{14.5}%
803 \abovedisplayskip 12\p@ \@plus3\p@ \@minus7\p@
804 \abovedisplayshortskip \z@ \@plus3\p@
805 \belowdisplayshortskip 6.5\p@ \@plus3.5\p@ \@minus3\p@
806 (/12pt)
807 (*14pt)
808 \setfontsize\normalsize\@xivpt{17.5}%
809 \abovedisplayskip 14\p@ \@plus3\p@ \@minus8\p@
810 \abovedisplayshortskip \z@ \@plus3\p@
811 \belowdisplayshortskip 7\p@ \@plus3.5\p@ \@minus3\p@
812 (/14pt)
813 (*17pt)
814 \setfontsize\normalsize\@xvipt{22}%
815 \abovedisplayskip 15\p@ \@plus4\p@ \@minus8\p@
816 \abovedisplayshortskip \z@ \@plus4\p@
817 \belowdisplayshortskip 8\p@ \@plus4\p@ \@minus3\p@
818 (/17pt)
819 (*20pt)
820 \setfontsize\normalsize\@xxpt{25}%
821 \abovedisplayskip 20\p@ \@plus5\p@ \@minus9\p@
822 \abovedisplayshortskip \z@ \@plus5\p@
823 \belowdisplayshortskip 12.5\p@ \@plus6\p@ \@minus3\p@
824 (/20pt)
825 (*25pt)
826 \setfontsize\normalsize\@xxvpt{30}%
827 \abovedisplayskip 25\p@ \@plus6\p@ \@minus10\p@
828 \abovedisplayshortskip \z@ \@plus6\p@
829 \belowdisplayshortskip 15\p@ \@plus7.5\p@ \@minus4\p@
830 (/25pt)
831 (*30pt)
832 \setfontsize\normalsize\@xxxpt{37}%
833 \abovedisplayskip 30\p@ \@plus7\p@ \@minus11\p@
834 \abovedisplayshortskip \z@ \@plus7\p@
835 \belowdisplayshortskip 18\p@ \@plus9\p@ \@minus4\p@
836 (/30pt)
837 (*36pt)
838 \setfontsize\normalsize\@xxxvipt{45}%
839 \abovedisplayskip 36\p@ \@plus8\p@ \@minus12\p@
840 \abovedisplayshortskip \z@ \@plus8\p@
841 \belowdisplayshortskip 22\p@ \@plus11\p@ \@minus5\p@
842 (/36pt)
843 (*48pt)
844 \setfontsize\normalsize\@xlvipt{60}%
845 \abovedisplayskip 48\p@ \@plus9\p@ \@minus13\p@

```

```

846 \abovedisplayshortskip \z@ \@plus9\p@
847 \belowdisplayshortskip 30\p@ \@plus15\p@ \@minus7\p@
848 </48pt>
849 <*60pt>
850 \@setfontsize\normalsize\@lpt{72}%
851 \abovedisplayskip 60\p@ \@plus10\p@ \@minus14\p@
852 \abovedisplayshortskip \z@ \@plus10\p@
853 \belowdisplayshortskip 36\p@ \@plus18\p@ \@minus9\p@
854 </60pt>

```

The `\belowdisplayskip` is always equal to the `\abovedisplayskip`. The parameters of the first level list are always given by `\@listI`.

```

855 \belowdisplayskip \abovedisplayskip
856 \let\@listi\@listI

```

We initially choose the `normalsize` font.

```

857 \normalsize

```

`\small` This code is similar to that for `\normalsize`. (Robert Schlicht¹² noted that in some cases `%` signs were missing after `\@setfontsize`).

For the larger sizes I have set `\topsep` to $2/3\text{onelineskip}$ and `\parsep` to $1/2\text{topsep}$.

```

858 \newcommand{\small}{%
859 <*9pt>
860 \@setfontsize\small\@viipt{9.5}%
861 \abovedisplayskip 6\p@ \@plus 2\p@ \@minus 4\p@
862 \abovedisplayshortskip \z@ \@plus 2\p@
863 \belowdisplayshortskip 4\p@ \@plus 2\p@ \@minus 2\p@
864 \def\@listi{\leftmargin\leftmargini
865             \topsep 2\p@ \@plus 2\p@ \@minus 2\p@
866             \parsep 1\p@ \@plus\p@ \@minus\p@
867             \itemsep \parsep
868 %%             \itemindent\z@
869             }%
870 </9pt>
871 <*10pt>
872 \@setfontsize\small\@ixpt{11}%
873 \abovedisplayskip 8.5\p@ \@plus3\p@ \@minus4\p@
874 \abovedisplayshortskip \z@ \@plus2\p@
875 \belowdisplayshortskip 4\p@ \@plus2\p@ \@minus2\p@
876 \def\@listi{\leftmargin\leftmargini
877             \topsep 4\p@ \@plus2\p@ \@minus2\p@
878             \parsep 2\p@ \@plus\p@ \@minus\p@
879             \itemsep \parsep
880 %%             \itemindent\z@
881             }%
882 </10pt>
883 <*11pt>

```

¹²w.m.l@gmx.net, via email on 2004/03/11.

```

884 \setfontsize\small\@xpt\@xipt
885 \abovedisplayskip 10\p@ \@plus2\p@ \@minus5\p@
886 \abovedisplayshortskip \z@ \@plus3\p@
887 \belowdisplayshortskip 6\p@ \@plus3\p@ \@minus3\p@
888 \def\@listi{\leftmargin\leftmargini
889             \topsep 6\p@ \@plus2\p@ \@minus2\p@
890             \parsep 3\p@ \@plus2\p@ \@minus\p@
891             \itemsep \parsep
892 %%             \itemindent\z@
893             }%
894 </11pt>
895 <*12pt>
896 \setfontsize\small\@xipt{13.6}%
897 \abovedisplayskip 11\p@ \@plus3\p@ \@minus6\p@
898 \abovedisplayshortskip \z@ \@plus3\p@
899 \belowdisplayshortskip 6.5\p@ \@plus3.5\p@ \@minus3\p@
900 \def\@listi{\leftmargin\leftmargini
901             \topsep 9\p@ \@plus3\p@ \@minus5\p@
902             \parsep 4.5\p@ \@plus2\p@ \@minus\p@
903             \itemsep \parsep
904 %%             \itemindent\z@
905             }%
906 </12pt>
907 <*14pt>
908 \setfontsize\small\@xipt{14.5}%
909 \abovedisplayskip 12\p@ \@plus3\p@ \@minus7\p@
910 \abovedisplayshortskip \z@ \@plus3\p@
911 \belowdisplayshortskip 6.5\p@ \@plus3.5\p@ \@minus3\p@
912 \def\@listi{\leftmargin\leftmargini
913             \topsep 11\p@ \@plus4\p@ \@minus6\p@
914             \parsep 6\p@ \@plus3\p@ \@minus\p@
915             \itemsep \parsep
916 %%             \itemindent\z@
917             }%
918 </14pt>
919 <*17pt>
920 \setfontsize\small\@xivpt{17}%
921 \abovedisplayskip 12\p@ \@plus3\p@ \@minus7\p@
922 \abovedisplayshortskip \z@ \@plus3\p@
923 \belowdisplayshortskip 7\p@ \@plus4\p@ \@minus3\p@
924 \def\@listi{\leftmargin\leftmargini
925             \topsep 11\p@ \@plus4\p@ \@minus6\p@
926             \parsep 6\p@ \@plus3\p@ \@minus\p@
927             \itemsep \parsep
928 %%             \itemindent\z@
929             }%
930 </17pt>
931 <*20pt>
932 \setfontsize\small\@xvipt{22}%
933 \abovedisplayskip 17\p@ \@plus4\p@ \@minus8\p@

```



```

934 \abovedisplayshortskip \z@ \@plus4\p@
935 \belowdisplayshortskip 11\p@ \@plus4\p@ \@minus3\p@
936 \def\@listi{\leftmargin\leftmargini
937         \topsep 14\p@ \@plus5\p@ \@minus7\p@
938         \parsep 7\p@ \@plus2\p@ \@minus3\p@
939         \itemsep \parsep}%
940 </20pt>
941 < *25pt>
942 \setfontsize\small\@xxpt{25}%
943 \abovedisplayskip 20\p@ \@plus5\p@ \@minus9\p@
944 \abovedisplayshortskip \z@ \@plus5\p@
945 \belowdisplayshortskip 12.5\p@ \@plus6\p@ \@minus3\p@
946 \def\@listi{\leftmargin\leftmargini
947         \topsep 16\p@ \@plus5\p@ \@minus8\p@
948         \parsep 8\p@ \@plus3\p@ \@minus4\p@
949         \itemsep \parsep}%
950 </25pt>
951 < *30pt>
952 \setfontsize\small\@xxvpt{30}%
953 \abovedisplayskip 25\p@ \@plus6\p@ \@minus10\p@
954 \abovedisplayshortskip \z@ \@plus6\p@
955 \belowdisplayshortskip 15\p@ \@plus7.5\p@ \@minus4\p@
956 \def\@listi{\leftmargin\leftmargini
957         \topsep 20\p@ \@plus7\p@ \@minus10\p@
958         \parsep 10\p@ \@plus4\p@ \@minus5\p@
959         \itemsep \parsep}%
960 </30pt>
961 < *36pt>
962 \setfontsize\small\@xxxpt{37}%
963 \abovedisplayskip 30\p@ \@plus7\p@ \@minus11\p@
964 \abovedisplayshortskip \z@ \@plus7\p@
965 \belowdisplayshortskip 18\p@ \@plus9\p@ \@minus4\p@
966 \def\@listi{\leftmargin\leftmargini
967         \topsep 24\p@ \@plus8\p@ \@minus12\p@
968         \parsep 12\p@ \@plus4\p@ \@minus6\p@
969         \itemsep \parsep}%
970 </36pt>
971 < *48pt>
972 \setfontsize\small\@xxxvipt{45}%
973 \abovedisplayskip 36\p@ \@plus8\p@ \@minus12\p@
974 \abovedisplayshortskip \z@ \@plus8\p@
975 \belowdisplayshortskip 22\p@ \@plus11\p@ \@minus5\p@
976 \def\@listi{\leftmargin\leftmargini
977         \topsep 30\p@ \@plus10\p@ \@minus15\p@
978         \parsep 15\p@ \@plus5\p@ \@minus7\p@
979         \itemsep \parsep}%
980 </48pt>
981 < *60pt>
982 \setfontsize\small\@xlvipt{60}%
983 \abovedisplayskip 48\p@ \@plus9\p@ \@minus13\p@

```

```

984 \abovedisplayshortskip \z@ \@plus9\p@
985 \belowdisplayshortskip 30\p@ \@plus15\p@ \@minus7\p@
986 \def\@listi{\leftmargin\leftmargini
987         \topsep 40\p@ \@plus13\p@ \@minus20\p@
988         \parsep 20\p@ \@plus6\p@ \@minus10\p@
989         \itemsep \parsep}%
990 </60pt>
991 \belowdisplayskip \abovedisplayskip
992 }

```

\footnotesize This code is similar to that for \small.

```

993 \newcommand{\footnotesize}{%
994 (*9pt)
995 \setfontsize\footnotesize\@viipt{8}%
996 \abovedisplayskip 6\p@ \@plus 2\p@ \@minus 4\p@
997 \abovedisplayshortskip \z@ \@plus 2\p@
998 \belowdisplayshortskip 4\p@ \@plus 2\p@ \@minus 2\p@
999 \def\@listi{\leftmargin\leftmargini
1000         \topsep 2\p@ \@plus 2\p@ \@minus 2\p@
1001         \parsep 1\p@ \@plus\p@ \@minus\p@
1002         \itemsep \parsep
1003 %% \itemindent\z@
1004 }%
1005 </9pt>
1006 (*10pt)
1007 \setfontsize\footnotesize\@viiipt{9.5}%
1008 \abovedisplayskip 6\p@ \@plus2\p@ \@minus4\p@
1009 \abovedisplayshortskip \z@ \@plus\p@
1010 \belowdisplayshortskip 3\p@ \@plus\p@ \@minus2\p@
1011 \def\@listi{\leftmargin\leftmargini
1012         \topsep 3\p@ \@plus\p@ \@minus\p@
1013         \parsep 2\p@ \@plus\p@ \@minus\p@
1014         \itemsep \parsep
1015 %% \itemindent\z@
1016 }%
1017 </10pt>
1018 (*11pt)
1019 \setfontsize\footnotesize\@ixpt{11}%
1020 \abovedisplayskip 8\p@ \@plus2\p@ \@minus4\p@
1021 \abovedisplayshortskip \z@ \@plus\p@
1022 \belowdisplayshortskip 4\p@ \@plus2\p@ \@minus2\p@
1023 \def\@listi{\leftmargin\leftmargini
1024         \topsep 4\p@ \@plus2\p@ \@minus2\p@
1025         \parsep 2\p@ \@plus\p@ \@minus\p@
1026         \itemsep \parsep
1027 %% \itemindent\z@
1028 }%
1029 </11pt>
1030 (*12pt)
1031 \setfontsize\footnotesize\@xpt\@xipt

```

```

1032 \abovedisplayskip 10\p@ \@plus2\p@ \@minus5\p@
1033 \abovedisplayshortskip \z@ \@plus3\p@
1034 \belowdisplayshortskip 6\p@ \@plus3\p@ \@minus3\p@
1035 \def\@listi{\leftmargin\leftmargini
1036         \topsep 6\p@ \@plus2\p@ \@minus2\p@
1037         \parsep 3\p@ \@plus2\p@ \@minus\p@
1038         \itemsep \parsep
1039 %%         \itemindent\z@
1040         }%
1041 </12pt>
1042 < *14pt>
1043 \setfontsize\footnotesize\@xipt{13.6}%
1044 \abovedisplayskip 11\p@ \@plus3\p@ \@minus6\p@
1045 \abovedisplayshortskip \z@ \@plus3\p@
1046 \belowdisplayshortskip 6.5\p@ \@plus3.5\p@ \@minus3\p@
1047 \def\@listi{\leftmargin\leftmargini
1048         \topsep 6.5\p@ \@plus3.5\p@ \@minus3\p@
1049         \parsep 4\p@ \@plus3\p@ \@minus\p@
1050         \itemsep \parsep
1051 %%         \itemindent\z@
1052         }%
1053 </14pt>
1054 < *17pt>
1055 \setfontsize\footnotesize\@xipt{14}%
1056 \abovedisplayskip 12\p@ \@plus3\p@ \@minus7\p@
1057 \abovedisplayshortskip \z@ \@plus3\p@
1058 \belowdisplayshortskip 6.5\p@ \@plus3.5\p@ \@minus3\p@
1059 \def\@listi{\leftmargin\leftmargini
1060         \topsep 6.5\p@ \@plus3.5\p@ \@minus3\p@
1061         \parsep 4\p@ \@plus3\p@ \@minus\p@
1062         \itemsep \parsep
1063 %%         \itemindent\z@
1064         }%
1065 </17pt>
1066 < *20pt>
1067 \setfontsize\footnotesize\@xivpt{17.5}%
1068 \abovedisplayskip 14\p@ \@plus3\p@ \@minus8\p@
1069 \abovedisplayshortskip \z@ \@plus3\p@
1070 \belowdisplayshortskip 7\p@ \@plus3.5\p@ \@minus3\p@
1071 \def\@listi{\leftmargin\leftmargini
1072         \topsep 12\p@ \@plus4\p@ \@minus6\p@
1073         \parsep 6\p@ \@plus2\p@ \@minus3\p@
1074         \itemsep \parsep}%
1075 </20pt>
1076 < *25pt>
1077 \setfontsize\footnotesize\@xvipt{22}%
1078 \abovedisplayskip 17\p@ \@plus4\p@ \@minus8\p@
1079 \abovedisplayshortskip \z@ \@plus4\p@
1080 \belowdisplayshortskip 11\p@ \@plus4\p@ \@minus3\p@
1081 \def\@listi{\leftmargin\leftmargini

```

```

1082             \topsep 14\p@ \@plus5\p@ \@minus7\p@
1083             \parsep 7\p@ \@plus2\p@ \@minus3\p@
1084             \itemsep \parsep}%
1085 \
```

`\miniscule` These are all much simpler than the previous macros, they just select a new
`\scriptsize` fontsize, but leave the parameters for displays and lists alone. The class provides
`\tiny` two additional sizes, `\miniscule` and `\HUGE`, with respect to the usual set. For
`\large`
`\Large`
`\LARGE`
`\huge`
`\Huge`
`\HUGE`

the larger sizes (e.g., 72pt and above) I have made the `\baselineskip` approximately 20 percent larger than the pt size.

```

1128 <*9pt>
1129 \ifextrafontsizes
1130   \newcommand*{\miniscule}{\@setfontsize\miniscule\@ivpt{5}}
1131 \else
1132   \newcommand*{\miniscule}{\@setfontsize\miniscule\@vpt{6}}
1133 \fi
1134 \newcommand*{\tiny}{\@setfontsize\tiny\@vpt{6}}
1135 \newcommand*{\scriptsize}{\@setfontsize\scriptsize\@vpt{7}}
1136 \newcommand*{\large}{\@setfontsize\large\@xpt{12}}
1137 \newcommand*{\Large}{\@setfontsize\Large\@xipt{13.6}}
1138 \newcommand*{\LARGE}{\@setfontsize\LARGE\@xiipt{14.5}}
1139 \newcommand*{\huge}{\@setfontsize\huge\@xivpt{18}}
1140 \newcommand*{\Huge}{\@setfontsize\Huge\@xxvpt{22}}
1141 \newcommand*{\HUGE}{\@setfontsize\HUGE\@xxvpt{25}}
1142 </9pt>
1143 <*10pt>
1144 \newcommand*{\miniscule}{\@setfontsize\miniscule\@vpt{6}}
1145 \newcommand*{\tiny}{\@setfontsize\tiny\@vpt{7}}
1146 \newcommand*{\scriptsize}{\@setfontsize\scriptsize\@viipt{8}}
1147 \newcommand*{\large}{\@setfontsize\large\@xipt{13.6}}
1148 \newcommand*{\Large}{\@setfontsize\Large\@xiipt{14.5}}
1149 \newcommand*{\LARGE}{\@setfontsize\LARGE\@xivpt{18}}
1150 \newcommand*{\huge}{\@setfontsize\huge\@xxvpt{22}}
1151 \newcommand*{\Huge}{\@setfontsize\Huge\@xxvpt{25}}
1152 \newcommand*{\HUGE}{\@setfontsize\HUGE\@xxvpt{30}}
1153 </10pt>
1154 <*11pt>
1155 \newcommand*{\miniscule}{\@setfontsize\miniscule\@vpt{7}}
1156 \newcommand*{\tiny}{\@setfontsize\tiny\@viipt{8}}
1157 \newcommand*{\scriptsize}{\@setfontsize\scriptsize\@viiipt{9.5}}
1158 \newcommand*{\large}{\@setfontsize\large\@xiipt{14.5}}
1159 \newcommand*{\Large}{\@setfontsize\Large\@xivpt{18}}
1160 \newcommand*{\LARGE}{\@setfontsize\LARGE\@xxvpt{22}}
1161 \newcommand*{\huge}{\@setfontsize\huge\@xxvpt{25}}
1162 \newcommand*{\Huge}{\@setfontsize\Huge\@xxvpt{30}}
1163 \ifextrafontsizes
1164   \newcommand*{\HUGE}{\@setfontsize\HUGE\@xxxvpt{37}}
1165 \else
1166   \let\HUGE=\Huge
1167 \fi
1168 </11pt>
1169 <*12pt>
1170 \newcommand*{\miniscule}{\@setfontsize\miniscule\@viipt{8}}
1171 \newcommand*{\tiny}{\@setfontsize\tiny\@viiipt{9.5}}
1172 \newcommand*{\scriptsize}{\@setfontsize\scriptsize\@ixpt\@xpt}
1173 \newcommand*{\large}{\@setfontsize\large\@xivpt{18}}
1174 \newcommand*{\Large}{\@setfontsize\Large\@xxvpt{22}}

```

```

1175 \newcommand*{\LARGE}{\@setfontsize\LARGE\@xxpt{25}}
1176 \newcommand*{\huge}{\@setfontsize\huge\@xxvpt{30}}
1177 \ifextrafontsizes
1178   \newcommand*{\Huge}{\@setfontsize\Huge\@xxxpt{37}}
1179   \newcommand*{\HUGE}{\@setfontsize\HUGE\@xxxvpt{45}}
1180 \else
1181   \let\Huge=\huge
1182   \let\HUGE=\huge
1183 \fi
1184 </12pt>
1185 (*14pt)
1186 \newcommand*{\miniscule}{\@setfontsize\miniscule\@viipt{9.5}}
1187 \newcommand*{\tiny}{\@setfontsize\tiny\@ixpt{10}}
1188 \newcommand*{\scriptsize}{\@setfontsize\scriptsize\@xpt{12}}
1189 \newcommand*{\large}{\@setfontsize\large\@xviipt{22}}
1190 \newcommand*{\Large}{\@setfontsize\Large\@xxpt{25}}
1191 \newcommand*{\LARGE}{\@setfontsize\LARGE\@xxvpt{30}}
1192 \ifextrafontsizes
1193   \newcommand*{\huge}{\@setfontsize\Huge\@xxxpt{37}}
1194   \newcommand*{\Huge}{\@setfontsize\Huge\@xxxvpt{45}}
1195   \newcommand*{\HUGE}{\@setfontsize\HUGE\@xlvipt{60}}
1196 \else
1197   \let\huge=\LARGE
1198   \let\Huge=\LARGE
1199   \let\HUGE=\LARGE
1200 \fi
1201 </14pt>
1202 (*17pt)
1203 \newcommand*{\miniscule}{\@setfontsize\miniscule\@ixpt{10}}
1204 \newcommand*{\tiny}{\@setfontsize\tiny\@xpt{12}}
1205 \newcommand*{\scriptsize}{\@setfontsize\scriptsize\@xipt{13.6}}
1206 \newcommand*{\large}{\@setfontsize\large\@xxpt{25}}
1207 \newcommand*{\Large}{\@setfontsize\Large\@xxvpt{30}}
1208 \ifextrafontsizes
1209   \newcommand*{\LARGE}{\@setfontsize\LARGE\@xxxpt{37}}
1210   \newcommand*{\huge}{\@setfontsize\huge\@xxxvpt{45}}
1211   \newcommand*{\Huge}{\@setfontsize\Huge\@xlvipt{60}}
1212   \newcommand*{\HUGE}{\@setfontsize\HUGE\@lxxpt{72}}
1213 \else
1214   \let\LARGE=\Large
1215   \let\huge=\Large
1216   \let\Huge=\Large
1217   \let\HUGE=\Large
1218 \fi
1219 </17pt>
1220 (*20pt)
1221 \newcommand*{\miniscule}{\@setfontsize\miniscule\@xpt{12}}
1222 \newcommand*{\tiny}{\@setfontsize\tiny\@xipt{13.6}}
1223 \newcommand*{\scriptsize}{\@setfontsize\scriptsize\@xiipt{14.5}}
1224 \newcommand*{\large}{\@setfontsize\large\@xxvpt{30}}

```

```

1225 \newcommand*{\Large}{\@setfontsize\Large\@xxxpt{37}}
1226 \newcommand*{\LARGE}{\@setfontsize\LARGE\@xxxvpt{45}}
1227 \newcommand*{\huge}{\@setfontsize\huge\@xlviipt{60}}
1228 \newcommand*{\Huge}{\@setfontsize\Huge\@lxxpt{72}}
1229 \newcommand*{\HUGE}{\@setfontsize\HUGE\@lxxiipt{86}}
1230 </20pt>
1231 (*25pt)
1232 \newcommand*{\miniscule}{\@setfontsize\miniscule\@xipt{13.6}}
1233 \newcommand*{\tiny}{\@setfontsize\tiny\@xipt{14.5}}
1234 \newcommand*{\scriptsize}{\@setfontsize\scriptsize\@xivpt{17.5}}
1235 \newcommand*{\large}{\@setfontsize\large\@xxxpt{37}}
1236 \newcommand*{\Large}{\@setfontsize\Large\@xxxvpt{45}}
1237 \newcommand*{\LARGE}{\@setfontsize\LARGE\@xlviipt{60}}
1238 \newcommand*{\huge}{\@setfontsize\huge\@lxxpt{72}}
1239 \newcommand*{\Huge}{\@setfontsize\Huge\@lxxiipt{86}}
1240 \newcommand*{\HUGE}{\@setfontsize\HUGE\@lxxxvpt{100}}
1241 </25pt>
1242 (*30pt)
1243 \newcommand*{\miniscule}{\@setfontsize\miniscule\@xipt{14.5}}
1244 \newcommand*{\tiny}{\@setfontsize\tiny\@xivpt{17.5}}
1245 \newcommand*{\scriptsize}{\@setfontsize\scriptsize\@xviipt{22}}
1246 \newcommand*{\large}{\@setfontsize\large\@xxxvpt{45}}
1247 \newcommand*{\Large}{\@setfontsize\Large\@xlviipt{60}}
1248 \newcommand*{\LARGE}{\@setfontsize\LARGE\@lxxpt{72}}
1249 \newcommand*{\huge}{\@setfontsize\huge\@lxxiipt{86}}
1250 \newcommand*{\Huge}{\@setfontsize\Huge\@lxxxvpt{100}}
1251 \newcommand*{\HUGE}{\@setfontsize\HUGE\@xcvpt{116}}
1252 </30pt>
1253 (*36pt)
1254 \newcommand*{\miniscule}{\@setfontsize\miniscule\@xivpt{17.5}}
1255 \newcommand*{\tiny}{\@setfontsize\tiny\@xviipt{22}}
1256 \newcommand*{\scriptsize}{\@setfontsize\scriptsize\@xxpt{25}}
1257 \newcommand*{\large}{\@setfontsize\large\@xlviipt{60}}
1258 \newcommand*{\Large}{\@setfontsize\Large\@lxxpt{72}}
1259 \newcommand*{\LARGE}{\@setfontsize\LARGE\@lxxiipt{86}}
1260 \newcommand*{\huge}{\@setfontsize\huge\@lxxxvpt{100}}
1261 \newcommand*{\Huge}{\@setfontsize\Huge\@xcvpt{116}}
1262 \newcommand*{\HUGE}{\@setfontsize\HUGE\@cviipt{132}}
1263 </36pt>
1264 (*48pt)
1265 \newcommand*{\miniscule}{\@setfontsize\miniscule\@xviipt{22}}
1266 \newcommand*{\tiny}{\@setfontsize\tiny\@xxpt{25}}
1267 \newcommand*{\scriptsize}{\@setfontsize\scriptsize\@xxvpt{30}}
1268 \newcommand*{\large}{\@setfontsize\large\@lxxpt{72}}
1269 \newcommand*{\Large}{\@setfontsize\Large\@lxxiipt{86}}
1270 \newcommand*{\LARGE}{\@setfontsize\LARGE\@lxxxvpt{100}}
1271 \newcommand*{\huge}{\@setfontsize\huge\@xcvpt{116}}
1272 \newcommand*{\Huge}{\@setfontsize\Huge\@cviipt{132}}
1273 \newcommand*{\HUGE}{\@setfontsize\HUGE\@cxxpt{144}}
1274 </48pt>

```

```

1275 (*60pt)
1276 \newcommand*{\miniscule}{\@setfontsize\miniscule\@xxpt{25}}
1277 \newcommand*{\tiny}{\@setfontsize\tiny\@xxvpt{30}}
1278 \newcommand*{\scriptsize}{\@setfontsize\scriptsize\@xxxpt{37}}
1279 \newcommand*{\large}{\@setfontsize\large\@lxxiip{86}}
1280 \newcommand*{\Large}{\@setfontsize\Large\@lxxxivpt{100}}
1281 \newcommand*{\LARGE}{\@setfontsize\LARGE\@xcvipt{116}}
1282 \newcommand*{\huge}{\@setfontsize\huge\@cviipt{132}}
1283 \newcommand*{\Huge}{\@setfontsize\Huge\@cxxxpt{144}}
1284 \newcommand*{\HUGE}{\@setfontsize\HUGE\@cxxxiipt{162}}
1285 \end{fontsize}

1286 \setlength{\fontsize}{9pt | 10pt | 11pt | 12pt | 14pt | 17pt | 20pt | 25pt | 30pt | 36pt | 48pt | 60pt}
1287 \end{fontsize}

```

`\captionsize` This internal command holds the font size for captions.

```
1288 \newcommand{\captionsize}{\normalsize}
```

7.2 Paragraphing

`\lineskip` These parameters control \TeX 's behaviour when two lines tend to come too close together.

```

1289 \setlength{\lineskip}{1\p@}
1290 \setlength{\normallineskip}{1\p@}

```

`\baselinestretch` This is used as a multiplier for `\baselineskip`. The default is to *not* stretch the baselines.

```
1291 \renewcommand{\baselinestretch}{1}
```

`\parskip` `\parskip` is additional vertical space between paragraphs; default is zero.

`\onelineskip` `\onelineskip` is the default space between baselines.

```

1292 \setlength{\parskip}{0\p@ \@plus \p@}
1293 \end{fontsize}
1294 (*9pt | 10pt | 11pt | 12pt | 14pt | 17pt | 20pt | 25pt | 30pt | 36pt | 48pt | 60pt)
1295 (9pt)\setlength{\onelineskip}{\@xpt\p@}
1296 (10pt)\setlength{\onelineskip}{\@xipt\p@}
1297 (11pt)\setlength{\onelineskip}{13.6\p@}
1298 (12pt)\setlength{\onelineskip}{14.5\p@}
1299 (14pt)\setlength{\onelineskip}{17.5\p@}
1300 (17pt)\setlength{\onelineskip}{22\p@}
1301 (20pt)\setlength{\onelineskip}{25\p@}
1302 (25pt)\setlength{\onelineskip}{30\p@}
1303 (30pt)\setlength{\onelineskip}{37\p@}
1304 (36pt)\setlength{\onelineskip}{45\p@}
1305 (48pt)\setlength{\onelineskip}{60\p@}
1306 (60pt)\setlength{\onelineskip}{72\p@}

```

`\parindent` `\parskip` gives extra vertical space between paragraphs and `\parindent` is the width of the paragraph indentation.


```

1307 \if@twocolumn
1308   \setlength\parindent{1em}
1309 \else
1310 <9pt>   \setlength\parindent{12\p@}
1311 <10pt>  \setlength\parindent{15\p@}
1312 <11pt>  \setlength\parindent{17\p@}
1313 <12pt>  \setlength\parindent{1.5em}
1314 <14pt>  \setlength\parindent{1.5em}
1315 <17pt>  \setlength\parindent{1.5em}
1316 <20pt>  \setlength\parindent{1.5em}
1317 <25pt>  \setlength\parindent{1.5em}
1318 <30pt>  \setlength\parindent{1.5em}
1319 <36pt>  \setlength\parindent{1.5em}
1320 <48pt>  \setlength\parindent{1.5em}
1321 <60pt>  \setlength\parindent{1.5em}
1322 \fi
1323 </9pt | 10pt | 11pt | 12pt | 14pt | 17pt | 20pt | 25pt | 30pt | 36pt | 48pt | 60pt>

\smallskipamount The values for these three parameters are set in the LaTeX kernel. Historically
\medskipamount   they have been size invariant, but I have changed them for the larger sizes
\bigskipamount   (\smallskipamount is 1/4 the fontsize, and the others each double up on the
                  next lower).

1324 <*9pt | 10pt | 11pt | 12pt | 14pt>
1325 \setlength\smallskipamount{3\p@ \@plus 1\p@ \@minus 1\p@}
1326 \setlength\medskipamount{6\p@ \@plus 2\p@ \@minus 2\p@}
1327 \setlength\bigskipamount{12\p@ \@plus 4\p@ \@minus 4\p@}
1328 </9pt | 10pt | 11pt | 12pt | 14pt>
1329 <*17pt | 20pt | 25pt | 30pt | 36pt | 48pt | 60pt>
1330 <*17pt>
1331 \setlength\smallskipamount{4\p@ \@plus 1\p@ \@minus 1\p@}
1332 \setlength\medskipamount{8\p@ \@plus 2\p@ \@minus 2\p@}
1333 \setlength\bigskipamount{17\p@ \@plus 4\p@ \@minus 4\p@}
1334 </17pt>
1335 <*20pt>
1336 \setlength\smallskipamount{5\p@ \@plus 1\p@ \@minus 1\p@}
1337 \setlength\medskipamount{10\p@ \@plus 2\p@ \@minus 2\p@}
1338 \setlength\bigskipamount{20\p@ \@plus 4\p@ \@minus 4\p@}
1339 </20pt>
1340 <*25pt>
1341 \setlength\smallskipamount{6\p@ \@plus 1\p@ \@minus 1\p@}
1342 \setlength\medskipamount{12\p@ \@plus 2\p@ \@minus 2\p@}
1343 \setlength\bigskipamount{24\p@ \@plus 4\p@ \@minus 4\p@}
1344 </25pt>
1345 <*30pt>
1346 \setlength\smallskipamount{5\p@ \@plus 1\p@ \@minus 1\p@}
1347 \setlength\medskipamount{10\p@ \@plus 2\p@ \@minus 2\p@}
1348 \setlength\bigskipamount{20\p@ \@plus 4\p@ \@minus 4\p@}
1349 </30pt>
1350 <*36pt>
1351 \setlength\smallskipamount{9\p@ \@plus 2\p@ \@minus 2\p@}

```

```

1352 \setlength\medskipamount{18\p@ \@plus4\p@ \@minus4\p@}
1353 \setlength\bigskipamount{36\p@ \@plus8\p@ \@minus8\p@}
1354 </36pt>
1355 <*48pt>
1356 \setlength\smallskipamount{12\p@ \@plus3\p@ \@minus3\p@}
1357 \setlength\medskipamount{24\p@ \@plus6\p@ \@minus6\p@}
1358 \setlength\bigskipamount{48\p@ \@plus12\p@ \@minus12\p@}
1359 </48pt>
1360 <*60pt>
1361 \setlength\smallskipamount{15\p@ \@plus4\p@ \@minus4\p@}
1362 \setlength\medskipamount{30\p@ \@plus8\p@ \@minus8\p@}
1363 \setlength\bigskipamount{60\p@ \@plus16\p@ \@minus16\p@}
1364 </60pt>
1365 </17pt | 20pt | 25pt | 30pt | 36pt | 48pt | 60pt>

\@lowpenalty The commands \nolinebreak and \nopagebreak put in penalties to discourage
\@medpenalty these breaks at the point they are put in. They use \@lowpenalty,
\@highpenalty \@medpenalty or \@highpenalty, dependent on their argument.

1366 (*class)
1367 \@lowpenalty 51
1368 \@medpenalty 151
1369 \@highpenalty 301

\clubpenalty These penalties are used to discourage club and widow lines. The default values
\widowpenalty are 150 each, but we want stronger discouragement.

1370 \clubpenalty 1000
1371 \widowpenalty 1000

\displaywidowpenalty Discourage, but do not prevent, widows in front of a math display and forbid
\predisplaypenalty breaking directly in front of a display. Allow break after a display without a
\postdisplaypenalty penalty. The default values are used, therefore we only show them here.

1372 % \displaywidowpenalty 50
1373 % \predisplaypenalty 10000
1374 % \postdisplaypenalty 0

\interlinepenalty Allow the breaking of a page in the middle of a paragraph.

1375 % \interlinepenalty 0

\brokenpenalty We allow the breaking of a page after a hyphenated line.

1376 % \brokenpenalty 100

```

7.3 Vertical spacing

`\headheight` The `\headheight` is the height of the box that will contain the running head. In this class it is point size dependent — `\onelineskip` (normally it is a constant 12pt).

`\headsep` The `\headsep` is the distance between the bottom of the running head and the top of the text. For the larger sizes this is 1.8 times the fontsize.

`\topskip`

`\footskip`

The `\topskip` is the `\baselineskip` for the first line on a page; L^AT_EX's output routine will not work properly if it has the value 0pt, so do not do that! For the larger sizes this is the font size.

The distance from the baseline of the box which contains the running footer to the baseline of last line of text is controlled by the `\footskip`. For the larger sizes this is 2.5 times the font size.

```

1377 </class>
1378 <*9pt | 10pt | 11pt | 12pt | 14pt | 17pt | 20pt | 25pt | 30pt | 36pt | 48pt | 60pt>
1379 <*9pt>
1380 \setlength\headheight{11\p@}
1381 \setlength\headsep{.225in}
1382 \setlength\topskip{9\p@}
1383 \setlength\footskip{.33in}
1384 </9pt>
1385 <*10pt>
1386 \setlength\headheight{12\p@}
1387 \setlength\headsep{.25in}
1388 \setlength\topskip{10\p@}
1389 \setlength\footskip{.35in}
1390 </10pt>
1391 <*11pt>
1392 \setlength\headheight{13.6\p@}
1393 \setlength\headsep{.275in}
1394 \setlength\topskip{11\p@}
1395 \setlength\footskip{.38in}
1396 </11pt>
1397 <*12pt>
1398 \setlength\headheight{14\p@}
1399 \setlength\headsep{.275in}
1400 \setlength\topskip{12\p@}
1401 \setlength\footskip{.30\p@}
1402 </12pt>
1403 <*14pt>
1404 \setlength\headheight{17.5\p@}
1405 \setlength\headsep{.30in}
1406 \setlength\topskip{14.4\p@}
1407 \setlength\footskip{.4in}
1408 </14pt>
1409 <*17pt>
1410 \setlength\headheight{22\p@}
1411 \setlength\headsep{.30in}
1412 \setlength\topskip{14.4\p@}
1413 \setlength\footskip{.4in}
1414 </17pt>
1415 <*20pt>
1416 \setlength\headheight{25\p@}
1417 \setlength\headsep{.36\p@}
1418 \setlength\topskip{20\p@}
1419 \setlength\footskip{.50\p@}

```

```

1420 </20pt>
1421 < *25pt>
1422 \setlength\headheight{30\p@}
1423 \setlength\headsep{45\p@}
1424 \setlength\topskip{25\p@}
1425 \setlength\footskip{62\p@}
1426 </25pt>
1427 < *30pt>
1428 \setlength\headheight{37\p@}
1429 \setlength\headsep{54\p@}
1430 \setlength\topskip{30\p@}
1431 \setlength\footskip{75\p@}
1432 </30pt>
1433 < *36pt>
1434 \setlength\headheight{45\p@}
1435 \setlength\headsep{65\p@}
1436 \setlength\topskip{36\p@}
1437 \setlength\footskip{90\p@}
1438 </36pt>
1439 < *48pt>
1440 \setlength\headheight{60\p@}
1441 \setlength\headsep{86\p@}
1442 \setlength\topskip{48\p@}
1443 \setlength\footskip{120\p@}
1444 </48pt>
1445 < *60pt>
1446 \setlength\headheight{72\p@}
1447 \setlength\headsep{108\p@}
1448 \setlength\topskip{60\p@}
1449 \setlength\footskip{150\p@}
1450 </60pt>

```

`\maxdepth` The \TeX primitive register `\maxdepth` has a function that is similar to that of `\topskip`. The register `@maxdepth` should always contain a copy of `\maxdepth`. In both plain \TeX and \LaTeX 2.09 `\maxdepth` had a fixed value of `4pt`; in native \LaTeX 2e mode we let the value depend on the typesize. We set it so that `\maxdepth + \topskip = typesize \times 1.5`. As it happens, in these classes `\topskip` is equal to the typesize, therefor we set `\maxdepth` to half the value of `\topskip`.

```

1451 \setlength\maxdepth{.5\topskip}
1452 \setlength@maxdepth\maxdepth

```

7.4 Footnotes

`\footnotesep` `\footnotesep` is the height of the strut placed at the beginning of every footnote. It equals the height of a normal `\footnotesize` strut, so no extra space appears between footnotes.

```

1453 <9pt>\setlength\footnotesep{5.6\p@}

```

```

1454 <10pt>\setlength\footnotesep{6.65\p@}
1455 <11pt>\setlength\footnotesep{7.7\p@}
1456 <12pt>\setlength\footnotesep{8.4\p@}
1457 <14pt>\setlength\footnotesep{9.5\p@}
1458 <17pt>\setlength\footnotesep{10.15\p@}
1459 <20pt>\setlength\footnotesep{12.6\p@}
1460 <25pt>\setlength\footnotesep{15.4\p@}
1461 <30pt>\setlength\footnotesep{17.5\p@}
1462 <36pt>\setlength\footnotesep{21\p@}
1463 <48pt>\setlength\footnotesep{25.9\p@}
1464 <60pt>\setlength\footnotesep{31.5\p@}

```

`\footins` `\skip\footins` is the space between the last line of the main text and the top of the first footnote.

```

1465 <9pt>\setlength{\skip\footins}{8\p@ \@plus 2\p@ \@minus 2\p@}
1466 <10pt>\setlength{\skip\footins}{9\p@ \@plus 4\p@ \@minus 2\p@}
1467 <11pt>\setlength{\skip\footins}{10\p@ \@plus 4\p@ \@minus 2\p@}
1468 <12pt>\setlength{\skip\footins}{10.8\p@ \@plus 4\p@ \@minus 2\p@}
1469 <14pt>\setlength{\skip\footins}{13\p@ \@plus 4\p@ \@minus 2\p@}
1470 <17pt>\setlength{\skip\footins}{16\p@ \@plus 5\p@ \@minus 3\p@}
1471 <20pt>\setlength{\skip\footins}{19\p@ \@plus 7\p@ \@minus 3\p@}
1472 <25pt>\setlength{\skip\footins}{24\p@ \@plus 8\p@ \@minus 4\p@}
1473 <30pt>\setlength{\skip\footins}{29\p@ \@plus 10\p@ \@minus 5\p@}
1474 <36pt>\setlength{\skip\footins}{35\p@ \@plus 12\p@ \@minus 6\p@}
1475 <48pt>\setlength{\skip\footins}{47\p@ \@plus 16\p@ \@minus 8\p@}
1476 <60pt>\setlength{\skip\footins}{59\p@ \@plus 20\p@ \@minus 10\p@}

```

7.5 Floats

Floats on a text page

`\floatsep` When a floating object is placed on a page with text, these parameters control
`\textfloatsep` the separation between the float and the other objects on the page. These
`\intextsep` parameters are used for both one-column mode and single-column floats in
two-column mode.

`\floatsep` is the space between adjacent floats that are moved to the top or bottom of the text page. For the larger sizes this is `\bigskip`.

`\textfloatsep` is the space between the main text and floats at the top or bottom of the page. For the larger sizes this is `1.45\onelineskip`.

`\intextsep` is the space between in-text floats and the text.

```

1477 <*9pt>
1478 \setlength\floatsep {12\p@ \@plus 2\p@ \@minus 2\p@}
1479 \setlength\textfloatsep{20\p@ \@plus 2\p@ \@minus 4\p@}
1480 \setlength\intextsep {12\p@ \@plus 2\p@ \@minus 2\p@}
1481 </9pt>
1482 <*10pt>
1483 \setlength\floatsep {12\p@ \@plus 2\p@ \@minus 2\p@}
1484 \setlength\textfloatsep{20\p@ \@plus 2\p@ \@minus 4\p@}
1485 \setlength\intextsep {12\p@ \@plus 2\p@ \@minus 2\p@}

```

```

1486 </10pt>
1487 < *11pt>
1488 \setlength\floatsep {12\p@ \@plus 2\p@ \@minus 2\p@}
1489 \setlength\textfloatsep{20\p@ \@plus 2\p@ \@minus 4\p@}
1490 \setlength\intextsep {12\p@ \@plus 2\p@ \@minus 2\p@}
1491 </11pt>
1492 < *12pt>
1493 \setlength\floatsep {12\p@ \@plus 2\p@ \@minus 2\p@}
1494 \setlength\textfloatsep{20\p@ \@plus 2\p@ \@minus 4\p@}
1495 \setlength\intextsep {14\p@ \@plus 4\p@ \@minus 4\p@}
1496 </12pt>
1497 < *14pt>
1498 \setlength\floatsep {14\p@ \@plus 4\p@ \@minus 4\p@}
1499 \setlength\textfloatsep{20\p@ \@plus 4\p@ \@minus 4\p@}
1500 \setlength\intextsep {14\p@ \@plus 4\p@ \@minus 4\p@}
1501 </14pt>
1502 < *17pt>
1503 \setlength\floatsep {15\p@ \@plus 4\p@ \@minus 4\p@}
1504 \setlength\textfloatsep{25\p@ \@plus 4\p@ \@minus 5\p@}
1505 \intextsep \floatsep
1506 </17pt>
1507 < *20pt>
1508 \setlength\floatsep {20\p@ \@plus 4\p@ \@minus 4\p@}
1509 \setlength\textfloatsep{36\p@ \@plus 4\p@ \@minus 8\p@}
1510 \intextsep \floatsep
1511 </20pt>
1512 < *25pt>
1513 \setlength\floatsep {24\p@ \@plus 4\p@ \@minus 4\p@}
1514 \setlength\textfloatsep{43\p@ \@plus 4\p@ \@minus 8\p@}
1515 \intextsep \floatsep
1516 </25pt>
1517 < *30pt>
1518 \setlength\floatsep {30\p@ \@plus 6\p@ \@minus 6\p@}
1519 \setlength\textfloatsep{54\p@ \@plus 6\p@ \@minus 12\p@}
1520 \intextsep \floatsep
1521 </30pt>
1522 < *36pt>
1523 \setlength\floatsep {36\p@ \@plus 8\p@ \@minus 8\p@}
1524 \setlength\textfloatsep{65\p@ \@plus 8\p@ \@minus 16\p@}
1525 \intextsep \floatsep
1526 </36pt>
1527 < *48pt>
1528 \setlength\floatsep {48\p@ \@plus 12\p@ \@minus 12\p@}
1529 \setlength\textfloatsep{87\p@ \@plus 12\p@ \@minus 24\p@}
1530 \intextsep \floatsep
1531 </48pt>
1532 < *60pt>
1533 \setlength\floatsep {60\p@ \@plus 16\p@ \@minus 16\p@}
1534 \setlength\textfloatsep{104\p@ \@plus 16\p@ \@minus 32\p@}
1535 \intextsep \floatsep

```

1536 $\langle/60pt\rangle$

$\backslash\text{dblfloatsep}$ When floating objects that span the whole $\backslash\text{textwidth}$ are placed on a text page
 $\backslash\text{dbltextfloatsep}$ and L^AT_EX is in twocolumn mode the separation between the float and the text is
 controlled by $\backslash\text{dblfloatsep}$ and $\backslash\text{dbltextfloatsep}$.

$\backslash\text{dblfloatsep}$ is the space between adjacent floats that are moved to the top or
 bottom of the text page.

$\backslash\text{dbltextfloatsep}$ is the space between the main text and floats at the top or
 bottom of the page.

1537 $\langle*9pt\rangle$

1538 $\backslash\text{setlength}\backslash\text{dblfloatsep}\quad\{12\backslash\text{p@}\backslash\text{@plus}\ 2\backslash\text{p@}\backslash\text{@minus}\ 2\backslash\text{p@}\}$

1539 $\backslash\text{setlength}\backslash\text{dbltextfloatsep}\{20\backslash\text{p@}\backslash\text{@plus}\ 2\backslash\text{p@}\backslash\text{@minus}\ 4\backslash\text{p@}\}$

1540 $\langle/9pt\rangle$

1541 $\langle*10pt\rangle$

1542 $\backslash\text{setlength}\backslash\text{dblfloatsep}\quad\{12\backslash\text{p@}\backslash\text{@plus}\ 2\backslash\text{p@}\backslash\text{@minus}\ 2\backslash\text{p@}\}$

1543 $\backslash\text{setlength}\backslash\text{dbltextfloatsep}\{20\backslash\text{p@}\backslash\text{@plus}\ 2\backslash\text{p@}\backslash\text{@minus}\ 4\backslash\text{p@}\}$

1544 $\langle/10pt\rangle$

1545 $\langle*11pt\rangle$

1546 $\backslash\text{setlength}\backslash\text{dblfloatsep}\quad\{12\backslash\text{p@}\backslash\text{@plus}\ 2\backslash\text{p@}\backslash\text{@minus}\ 2\backslash\text{p@}\}$

1547 $\backslash\text{setlength}\backslash\text{dbltextfloatsep}\{20\backslash\text{p@}\backslash\text{@plus}\ 2\backslash\text{p@}\backslash\text{@minus}\ 4\backslash\text{p@}\}$

1548 $\langle/11pt\rangle$

1549 $\langle*12pt\rangle$

1550 $\backslash\text{setlength}\backslash\text{dblfloatsep}\quad\{14\backslash\text{p@}\backslash\text{@plus}\ 2\backslash\text{p@}\backslash\text{@minus}\ 4\backslash\text{p@}\}$

1551 $\backslash\text{setlength}\backslash\text{dbltextfloatsep}\{20\backslash\text{p@}\backslash\text{@plus}\ 2\backslash\text{p@}\backslash\text{@minus}\ 4\backslash\text{p@}\}$

1552 $\langle/12pt\rangle$

1553 $\langle*14pt\rangle$

1554 $\backslash\text{setlength}\backslash\text{dblfloatsep}\quad\{14\backslash\text{p@}\backslash\text{@plus}\ 2\backslash\text{p@}\backslash\text{@minus}\ 4\backslash\text{p@}\}$

1555 $\backslash\text{setlength}\backslash\text{dbltextfloatsep}\{20\backslash\text{p@}\backslash\text{@plus}\ 2\backslash\text{p@}\backslash\text{@minus}\ 4\backslash\text{p@}\}$

1556 $\langle/14pt\rangle$

1557 $\langle*17pt\rangle$

1558 $\backslash\text{setlength}\backslash\text{dblfloatsep}\quad\{15\backslash\text{p@}\backslash\text{@plus}\ 4\backslash\text{p@}\backslash\text{@minus}\ 4\backslash\text{p@}\}$

1559 $\backslash\text{setlength}\backslash\text{dbltextfloatsep}\{25\backslash\text{p@}\backslash\text{@plus}\ 5\backslash\text{p@}\backslash\text{@minus}\ 5\backslash\text{p@}\}$

1560 $\langle/17pt\rangle$

1561 $\langle*20pt\rangle$

1562 $\backslash\text{dblfloatsep}\ \backslash\text{floatsep}$

1563 $\backslash\text{dbltextfloatsep}\ \backslash\text{textfloatsep}$

1564 $\langle/20pt\rangle$

1565 $\langle*25pt\rangle$

1566 $\backslash\text{dblfloatsep}\ \backslash\text{floatsep}$

1567 $\backslash\text{dbltextfloatsep}\ \backslash\text{textfloatsep}$

1568 $\langle/25pt\rangle$

1569 $\langle*30pt\rangle$

1570 $\backslash\text{dblfloatsep}\ \backslash\text{floatsep}$

1571 $\backslash\text{dbltextfloatsep}\ \backslash\text{textfloatsep}$

1572 $\langle/30pt\rangle$

1573 $\langle*36pt\rangle$

1574 $\backslash\text{dblfloatsep}\ \backslash\text{floatsep}$

1575 $\backslash\text{dbltextfloatsep}\ \backslash\text{textfloatsep}$

1576 $\langle/36pt\rangle$

```

1577 (*48pt)
1578 \dblfloatsep \floatsep
1579 \dbltextfloatsep \textfloatsep
1580 (/48pt)
1581 (*60pt)
1582 \dblfloatsep \floatsep
1583 \dbltextfloatsep \textfloatsep
1584 (/60pt)

```

Floats on their own page or column

`\@fptop` When floating objects are placed on separate pages the layout of such pages is controlled by these parameters. At the top of the page `\@fptop` amount of stretchable whitespace is inserted, at the bottom of the page we get an `\@fpbot` amount of stretchable whitespace. Between adjacent floats the `\@fpsep` is inserted. For the larger sizes `\@fpsep` is `.7\onelineskip`. These parameters are used for the placement of floating objects in one column mode, or in single column floats in two column mode. Note that at least one of the two parameters `\@fptop` and `\@fpbot` should contain a plus `...fil` to allow filling the remaining empty space.

```

1585 (*9pt)
1586 \setlength\@fptop{0\p@ \@plus 1fil}
1587 \setlength\@fpsep{9\p@ \@plus 2fil}
1588 \setlength\@fpbot{0\p@ \@plus 1fil}
1589 (/9pt)
1590 (*10pt)
1591 \setlength\@fptop{0\p@ \@plus 1fil}
1592 \setlength\@fpsep{8\p@ \@plus 2fil}
1593 \setlength\@fpbot{0\p@ \@plus 1fil}
1594 (/10pt)
1595 (*11pt)
1596 \setlength\@fptop{0\p@ \@plus 1fil}
1597 \setlength\@fpsep{8\p@ \@plus 2fil}
1598 \setlength\@fpbot{0\p@ \@plus 1fil}
1599 (/11pt)
1600 (*12pt)
1601 \setlength\@fptop{0\p@ \@plus 1fil}
1602 \setlength\@fpsep{10\p@ \@plus 2fil}
1603 \setlength\@fpbot{0\p@ \@plus 1fil}
1604 (/12pt)
1605 (*14pt)
1606 \setlength\@fptop{0\p@ \@plus 1fil}
1607 \setlength\@fpsep{10\p@ \@plus 2fil}
1608 \setlength\@fpbot{0\p@ \@plus 1fil}
1609 (/14pt)
1610 (*17pt)
1611 \setlength\@fptop{0\p@ \@plus 1fil}
1612 \setlength\@fpsep{12\p@ \@plus 2fil}
1613 \setlength\@fpbot{0\p@ \@plus 1fil}

```



```

1614 </17pt>
1615 <*20pt>
1616 \setlength\@fptop{0\p@ \@plus 1fil}
1617 \setlength\@fpsep{17\p@ \@plus 2fil}
1618 \@fpbot \@fptop
1619 </20pt>
1620 <*25pt>
1621 \setlength\@fptop{0\p@ \@plus 1fil}
1622 \setlength\@fpsep{21\p@ \@plus 2fil}
1623 \@fpbot \@fptop
1624 </25pt>
1625 <*30pt>
1626 \setlength\@fptop{0\p@ \@plus 1fil}
1627 \setlength\@fpsep{26\p@ \@plus 2fil}
1628 \@fpbot \@fptop
1629 </30pt>
1630 <*36pt>
1631 \setlength\@fptop{0\p@ \@plus 1fil}
1632 \setlength\@fpsep{31\p@ \@plus 2fil}
1633 \@fpbot \@fptop
1634 </36pt>
1635 <*48pt>
1636 \setlength\@fptop{0\p@ \@plus 1fil}
1637 \setlength\@fpsep{42\p@ \@plus 2fil}
1638 \@fpbot \@fptop
1639 </48pt>
1640 <*60pt>
1641 \setlength\@fptop{0\p@ \@plus 1fil}
1642 \setlength\@fpsep{50\p@ \@plus 2fil}
1643 \@fpbot \@fptop
1644 </60pt>

```

\@dblftop Double column floats in two column mode are handled with similar parameters.

```

\@dblfpsep 1645 <*9pt>
\@dblfpbot 1646 \setlength\@dblftop{0\p@ \@plus 1fil}
1647 \setlength\@dblfpsep{7\p@ \@plus 2fil}
1648 \setlength\@dblfpbot{0\p@ \@plus 1fil}
1649 </9pt>
1650 <*10pt>
1651 \setlength\@dblftop{0\p@ \@plus 1fil}
1652 \setlength\@dblfpsep{8\p@ \@plus 2fil}
1653 \setlength\@dblfpbot{0\p@ \@plus 1fil}
1654 </10pt>
1655 <*11pt>
1656 \setlength\@dblftop{0\p@ \@plus 1fil}
1657 \setlength\@dblfpsep{8\p@ \@plus 2fil}
1658 \setlength\@dblfpbot{0\p@ \@plus 1fil}
1659 </11pt>
1660 <*12pt>
1661 \setlength\@dblftop{0\p@ \@plus 1fil}

```

```

1662 \setlength\@dblfpsep{10\p@ \@plus 2fil}
1663 \setlength\@dblfpbot{0\p@ \@plus 1fil}
1664 </12pt>
1665 <*14pt>
1666 \setlength\@dblfpsep{0\p@ \@plus 1fil}
1667 \setlength\@dblfpsep{12\p@ \@plus 2fil}
1668 \setlength\@dblfpbot{0\p@ \@plus 1fil}
1669 </14pt>
1670 <*17pt>
1671 \setlength\@dblfpsep{0\p@ \@plus 1fil}
1672 \setlength\@dblfpsep{12\p@ \@plus 2fil}
1673 \setlength\@dblfpbot{0\p@ \@plus 1fil}
1674 </17pt>
1675 <*20pt>
1676 \@dblfpsep \@fpsep
1677 \@dblfpsep \@fpsep
1678 \@dblfpbot \@dblfpsep
1679 </20pt>
1680 <*25pt>
1681 \@dblfpsep \@fpsep
1682 \@dblfpsep \@fpsep
1683 \@dblfpbot \@dblfpsep
1684 </25pt>
1685 <*30pt>
1686 \@dblfpsep \@fpsep
1687 \@dblfpsep \@fpsep
1688 \@dblfpbot \@dblfpsep
1689 </30pt>
1690 <*36pt>
1691 \@dblfpsep \@fpsep
1692 \@dblfpsep \@fpsep
1693 \@dblfpbot \@dblfpsep
1694 </36pt>
1695 <*48pt>
1696 \@dblfpsep \@fpsep
1697 \@dblfpsep \@fpsep
1698 \@dblfpbot \@dblfpsep
1699 </48pt>
1700 <*60pt>
1701 \@dblfpsep \@fpsep
1702 \@dblfpsep \@fpsep
1703 \@dblfpbot \@dblfpsep
1704 </60pt>

```

7.6 The measure

The width of a line of text (and therefore the text block) is termed the *measure*.

`\lxvchars` The length `\lxvchars` is the approximate length of a normal text line containing 65 characters (a typesetters rule of thumb is that there should be about 60–70

characters per line).

```

1705 <9pt>\setlength\lxvchars{276\p@} %
1706 <10pt>\setlength\lxvchars{300\p@} % standard 345pt
1707 <11pt>\setlength\lxvchars{324\p@} % standard 360pt
1708 <12pt>\setlength\lxvchars{336\p@} % standard 390pt
1709 <14pt>\setlength\lxvchars{408\p@} %
1710 <17pt>\setlength\lxvchars{444\p@} %
1711 <20pt>\setlength\lxvchars{528\p@} %
1712 <25pt>\setlength\lxvchars{626\p@} %
1713 <30pt>\setlength\lxvchars{748\p@} %
1714 <36pt>\setlength\lxvchars{891\p@} %
1715 <48pt>\setlength\lxvchars{1177\p@} %
1716 <60pt>\setlength\lxvchars{1463\p@} %

```

\xlvchars The length `\xlvchars` is the approximate length of a normal double column text line containing 45 characters (a typesetters rule of thumb is that there should be about 40–50 characters per column line).

```

1717 <9pt>\setlength\xlvchars{192\p@} %
1718 <10pt>\setlength\xlvchars{204\p@} %
1719 <11pt>\setlength\xlvchars{216\p@} %
1720 <12pt>\setlength\xlvchars{240\p@} %
1721 <14pt>\setlength\xlvchars{288\p@} %
1722 <17pt>\setlength\xlvchars{312\p@} %
1723 <20pt>\setlength\xlvchars{365\p@} %
1724 <25pt>\setlength\xlvchars{438\p@} %
1725 <30pt>\setlength\xlvchars{518\p@} %
1726 <36pt>\setlength\xlvchars{617\p@} %
1727 <48pt>\setlength\xlvchars{815\p@} %
1728 <60pt>\setlength\xlvchars{1014\p@} %

```

\marginparsep `\marginparsep` is the horizontal space between the text block and marginal notes, while **\marginparpush** `\marginparpush` is the minimum vertical separation between the notes.

```

1729 \if@twocolumn
1730 <*9pt | 10pt | 11pt | 12pt | 14pt>
1731 \setlength\marginparsep{10\p@}
1732 </9pt | 10pt | 11pt | 12pt | 14pt>
1733 <*17pt | 20pt | 25pt | 30pt | 36pt | 48pt | 60pt>
1734 \setlength\marginparsep{1em}
1735 </17pt | 20pt | 25pt | 30pt | 36pt | 48pt | 60pt>
1736 \else
1737 <*9pt | 10pt | 11pt | 12pt | 14pt>
1738 \setlength\marginparsep{7\p@}
1739 </9pt | 10pt | 11pt | 12pt | 14pt>
1740 <*17pt | 20pt | 25pt | 30pt | 36pt | 48pt | 60pt>
1741 \setlength\marginparsep{0.7em}
1742 </17pt | 20pt | 25pt | 30pt | 36pt | 48pt | 60pt>
1743 \fi
1744 <9pt | 10pt | 11pt>\setlength{\marginparpush}{5\p@}

```

```

1745 <12pt | 14pt>\setlength{\marginparpush}{7\p@}
1746 <*17pt | 20pt | 25pt | 30pt | 36pt | 48pt | 60pt>
1747 \setlength{\marginparpush}{0.5em}
1748 </17pt | 20pt | 25pt | 30pt | 36pt | 48pt | 60pt>

1749 </9pt | 10pt | 11pt | 12pt | 14pt | 17pt | 20pt | 25pt | 30pt | 36pt | 48pt | 60pt>
1750 <*class>

```

8 Page Layout

8.1 The typeblock and margins

All margin dimensions are measured from a point one inch from the top and lefthand side of the page.

`\setlxvchars` These were suggested and supplied by Morten Høgholm (18 May 2003).
`\setxlvchars` `\setlxvchars[⟨fontspec⟩]` sets `\lxvchars` to the length of a line containing 65 characters in the `⟨fontspec⟩`.
Similarly `\setxlvchars[⟨fontspec⟩]` sets `\xlvchars` for 45 characters.

```

1751 \newcommand*{\setlxvchars}[1][\normalfont]{\begingroup
1752 #1
1753 \settowidth{\lxvchars}{abcdefghijklmnopqrstuvwxyz}%
1754 \setlength{\lxvchars}{2.042\lxvchars}%
1755 \addtolength{\lxvchars}{33.41pt}%
1756 \global\lxvchars=\lxvchars
1757 \endgroup}
1758 \newcommand*{\setxlvchars}[1][\normalfont]{\begingroup
1759 #1
1760 \settowidth{\xlvchars}{abcdefghijklmnopqrstuvwxyz}%
1761 \setlength{\xlvchars}{1.415\xlvchars}%
1762 \addtolength{\xlvchars}{23.03pt}%
1763 \global\xlvchars=\xlvchars
1764 \endgroup}
1765

```

`\setrectanglesize` The macro `\setrectanglesize{⟨H⟩}{⟨W⟩}{⟨r⟩}` calculates the height and width of a rectangle given any two out of the three arguments. An unvalued argument is denoted by `*`.
Table 2 shows the argument combinations and the result for each combination. The calculated height and width are stored in `\@tempdima` and `\@tempdimb` respectively. Both lengths are set to zero if there is an error.

```

1766 \newcommand*{\setrectanglesize}[3]{%
1767 \nametest{#1}{*}%
1768 \ifsamename % H = *
1769 \nametest{#2}{*}%
1770 \ifsamename % W = *
1771 \@memerror{%
1772 The combination of argument values is ambiguous.\MessageBreak

```

Table 2: Arguments and results for `\setrectanglesize`

H	W	r	Result
*	W	r	$H = rW$
*	W	*	$H = W$
*	*	r	ambiguous
*	*	*	ambiguous
H	W	r	H, W
H	W	*	H, W
H	*	r	$W = rH$
H	*	*	$W = H$

```

1773     The lengths will be set to zero}\@ehd}%
1774     \setlength{\@tempdima}{0pt}%
1775     \setlength{\@tempdimb}{0pt}%
1776 \else                                     % W
1777     \nametest{#3}{*}%
1778     \ifsamename                           % r = *
1779         \setlength{\@tempdimb}{#2}%
1780         \setlength{\@tempdima}{\@tempdimb}%
1781     \else                                  % r
1782         \setlength{\@tempdimb}{#2}%
1783         \setlength{\@tempdima}{#3\@tempdimb}%
1784     \fi
1785 \fi
1786 \else                                     % H
1787     \nametest{#2}{*}%
1788     \ifsamename                           % W = *
1789         \nametest{#3}{*}%
1790         \ifsamename                       % r = *
1791             \setlength{\@tempdima}{#1}%
1792             \setlength{\@tempdimb}{\@tempdima}%
1793         \else                             % r
1794             \setlength{\@tempdima}{#1}%
1795             \setlength{\@tempdimb}{#3\@tempdima}%
1796         \fi
1797     \else                                  % W
1798         \setlength{\@tempdima}{#1}%
1799         \setlength{\@tempdimb}{#2}%
1800     \fi
1801 \fi}
1802

```

`\setfillsize` Consider a set of 4 lengths, T , L , C , and R , such that $T = L + C + R$, where T is a fixed size and normally C is also fixed. Also L and R may be specified independently of each other or as a ratio (i.e., $L = rR$ or $R = rL$). The macro `\setfillsize{\langle T \rangle}{\langle C \rangle}{\langle L \rangle}{\langle R \rangle}{\langle r \rangle}` maintains these constraints among the variables, where an unvalued argument is denoted by $*$.

Table 3: Arguments and results for `\setfillsize`

C	L	R	r	Result
*	*	R	r	$L = rR,$ $C = T - L - R$
*	*	R	*	$L = R,$ $C = T - L - R$
*	*	*	r	ambiguous
*	*	*	*	ambiguous
*	L	R	r	$L, R,$ $C = T - L - R$
*	L	R	*	$L, R,$ $C = T - L - R$
*	L	*	r	$R = rL,$ $C = T - L - R$
*	L	*	*	$R = L,$ $C = T - L - R$
C	*	R	r	$L = T - C - R$ C
C	*	R	*	$L = T - C - R,$ C
C	*	*	r	$L + R = T - C, R = rL,$ C
C	*	*	*	$L + R = T - C, R = L,$ C
C	L	R	r	ambiguous C
C	L	R	*	ambiguous C
C	L	*	r	$R = T - C - L,$ C
C	L	*	*	$R = T - C - L,$ C

Table 3 shows the argument combinations and the result for each combination. The calculated values of C, L and R are stored `\@tempdimc`, `\@tempdima` and `\@tempdimb` respectively. If there is an error the lengths are set to zero.

```

1803 \newcommand*\setfillsize}[5]{%
1804   \nametest{#2}{*}%
1805   \ifsamename                               % C = *
1806     \nametest{#3}{*}%
1807     \ifsamename                               % L = *
1808       \nametest{#4}{*}%
1809       \ifsamename                             % R = *
1810         \@memerror{%
1811           The combination of argument values is ambiguous.\MessageBreak
1812           The lengths will be set to zero}{\@ehd}
1813         \setlength{\@tempdima}{0pt}%
1814         \setlength{\@tempdimb}{0pt}%
1815         \setlength{\@tempdimc}{0pt}%
1816       \else                                   % R
1817         \nametest{#5}{*}%
1818         \ifsamename                             % r = *
1819           \setlength{\@tempdimb}{#4}%
1820           \setlength{\@tempdima}{\@tempdimb}%
1821           \setlength{\@tempdimc}{#1}%
1822           \advance\@tempdimc -\@tempdima
1823           \advance\@tempdimc -\@tempdimb
1824         \else                                   % r
1825           \setlength{\@tempdimb}{#4}%
1826           \setlength{\@tempdima}{#5\@tempdimb}%

```

```

1827         \setlength{\@tempdimc}{#1}%
1828         \advance\@tempdimc -\@tempdima
1829         \advance\@tempdimc -\@tempdimb
1830     \fi
1831 \fi
1832 \else                                     % L
1833     \nametest{#4}{*}%
1834     \ifsamename                         % R = *
1835     \nametest{#5}{*}%
1836     \ifsamename                         % r = *
1837         \setlength{\@tempdima}{#3}%
1838         \setlength{\@tempdimb}{\@tempdima}
1839         \setlength{\@tempdimc}{#1}%
1840         \advance\@tempdimc -\@tempdima
1841         \advance\@tempdimc -\@tempdimb
1842     \else                                 % r
1843         \setlength{\@tempdima}{#3}%
1844         \setlength{\@tempdimb}{#5\@tempdima}
1845         \setlength{\@tempdimc}{#1}%
1846         \advance\@tempdimc -\@tempdima
1847         \advance\@tempdimc -\@tempdimb
1848     \fi
1849 \else                                     % R
1850     \setlength{\@tempdima}{#3}%
1851     \setlength{\@tempdimb}{#4}%
1852     \setlength{\@tempdimc}{#1}%
1853     \advance\@tempdimc -\@tempdima
1854     \advance\@tempdimc -\@tempdimb
1855 \fi
1856 \fi
1857 \else                                     % C is valued
1858     \nametest{#3}{*}%
1859     \ifsamename                         % L = *
1860     \nametest{#4}{*}%
1861     \ifsamename                         % R = *
1862     \nametest{#5}{*}%
1863     \ifsamename                         % r = *
1864         \setlength{\@tempdimc}{#2}%
1865         \setlength{\@tempdima}{#1}%
1866         \advance\@tempdima -\@tempdimc
1867         \@tempdima = 0.5\@tempdima
1868         \@tempdimb = \@tempdima
1869     \else                                 % r (CODE PERHAPS FIXED)
1870         \setlength{\@tempdimc}{#2}         % C
1871         \setlength{\@tempdimb}{#1}         % T
1872         \advance\@tempdimb -\@tempdimc % T - C
1873         \@tempdima = 1000sp
1874         \setlength{\@tempdima}{#5\@tempdima} % 1000r sp
1875         \advance\@tempdima by 1000sp % 1000(1+r)sp
1876         \@tempcnta = \@tempdima % 1000(1+r)

```

```

1877      \@tempdima = \@tempdimb          % T - C
1878      \divide\@tempdima by \@tempcnta  % (T-C)/1000(1+r) pts
1879      \@tempdima = 1000\@tempdima      % (T-C)/(1+r) pts = L
1880      \advance\@tempdimb by -\@tempdima % = R
1881      \fi
1882      \else                                % R
1883      \setlength{\@tempdimc}{#2}%
1884      \setlength{\@tempdimb}{#4}%
1885      \setlength{\@tempdima}{#1}%
1886      \advance\@tempdima -\@tempdimc
1887      \advance\@tempdima -\@tempdimb
1888      \fi
1889      \else                                % L
1890      \nametest{#4}{*}%
1891      \ifsamenamename                    % R = *
1892      \setlength{\@tempdimc}{#2}%
1893      \setlength{\@tempdima}{#3}%
1894      \setlength{\@tempdimb}{#1}%
1895      \advance\@tempdimb -\@tempdimc
1896      \advance\@tempdimb -\@tempdima
1897      \else                                % R
1898      \@memerror{%
1899      The combination of argument values is ambiguous.\MessageBreak
1900      The lengths will be set to zero}{\@ehd}%
1901      \setlength{\@tempdima}{Opt}%
1902      \setlength{\@tempdimb}{Opt}%
1903      \setlength{\@tempdimc}{#2}%
1904      \fi
1905      \fi
1906      \fi}
1907
\setstocksize \setstocksize{<height>}{<width>} sets the height and width of the stock
\settrims material and \settrims{<top>}{<edge>} sets the trim lengths for the top and
\settrimmedsize side (fore edge) of the stock. The macro
\settrimmedsize{<height>}{<width>}{<ratio>} sets the size for the trimmed
page, based on \setrectanglesize.
1908 \newcommand{\setstocksize}[2]{%
1909 \setlength{\stockheight}{#1}%
1910 \setlength{\stockwidth}{#2}}
1911 \newcommand{\settrims}[2]{%
1912 \setlength{\trimtop}{#1}%
1913 \setlength{\trimedge}{#2}}
1914 \newcommand{\settrimmedsize}[3]{%
1915 \setrectanglesize{#1}{#2}{#3}%
1916 \setlength{\paperheight}{\@tempdima}%
1917 \setlength{\paperwidth}{\@tempdimb}}
1918

```

\settypeblocksize \settypeblocksize{<height>}{<width>}{<ratio>} calculates the \textheight and

`\textwidth` from two out of the three arguments.

```
1919 \newcommand{\settypeblocksize}[3]{%
1920   \setrectanglesize{#1}{#2}{#3}%
1921   \setlength{\textheight}{\@tempdima}%
1922   \setlength{\textwidth}{\@tempdimb}}
1923
```

`\binding` The length `\binding` is an allowance on the spine margin for binding.
`\setbinding` `\setbinding{<length>}` sets the `\binding`.

```
1924 \newlength{\binding}
1925 \newcommand*{\setbinding}[1]{\setlength{\binding}{#1}}
1926 \setbinding{0pt}
1927
```

`\spinemargin` `\setlrmargins{<L>}{<R>}{<r>}` sets the Left (spine) and Right (fore edge)
`\foremargin` margins with constant typeblock.

```
\setlrmargins 1928 \newlength{\spinemargin}
1929 \newlength{\foremargin}
1930 \newcommand{\setlrmargins}[3]{%
1931   \advance\paperwidth -\binding
1932   \setfillsize{\paperwidth}{\textwidth}{#1}{#2}{#3}%
1933   \setlength{\textwidth}{\@tempdimc}%
1934   \setlength{\spinemargin}{\@tempdima}%
1935   \setlength{\foremargin}{\@tempdimb}%
1936   \advance\paperwidth \binding
1937   \advance\spinemargin \binding}
1938
```

`\setlrmarginsandblock` `\setlrmarginsandblock{<L>}{<R>}{<r>}` sets the Left (spine) and Right (fore edge) margins with variable typeblock.

```
1939 \newcommand{\setlrmarginsandblock}[3]{%
1940   \advance\paperwidth -\binding
1941   \setfillsize{\paperwidth}{*}{#1}{#2}{#3}%
1942   \setlength{\textwidth}{\@tempdimc}%
1943   \setlength{\spinemargin}{\@tempdima}%
1944   \setlength{\foremargin}{\@tempdimb}%
1945   \advance\paperwidth \binding
1946   \advance\spinemargin \binding}
1947
```

`\uppermargin` `\setulmargins{<L>}{<R>}{<r>}` sets the Left (upper) and Right (lower) margins
`\lowermargin` with constant typeblock.

```
\setulmargins 1948 \newlength{\uppermargin}
1949 \newlength{\lowermargin}
1950 \newcommand{\setulmargins}[3]{%
1951   \setfillsize{\paperheight}{\textheight}{#1}{#2}{#3}%
1952   \setlength{\textheight}{\@tempdimc}%
1953   \setlength{\uppermargin}{\@tempdima}%
1954   \setlength{\lowermargin}{\@tempdimb}}
```

1955

`\setulmarginsandblock` `\setulmarginsandblock{<L>}{<R>}{<r>}` sets the Left (upper) and Right (lower) margins with variable typeblock.

```
1956 \newcommand{\setulmarginsandblock}[3]{%
1957   \setfillsize{\paperheight}{*}{#1}{#2}{#3}%
1958   \setlength{\textheight}{\@tempdimc}%
1959   \setlength{\uppermargin}{\@tempdima}%
1960   \setlength{\lowermargin}{\@tempdimb}}
1961
```

`\headdrop` `\setheaderspaces{<L>}{<R>}{<r>}` sets the Left (head margin) and Right (headsep) spacing with constant headheight.

```
1962 \newlength{\headdrop}
1963 \newcommand{\setheaderspaces}[3]{%
1964   \setfillsize{\uppermargin}{\headheight}{#1}{#2}{#3}%
1965   \setlength{\headheight}{\@tempdimc}%
1966   \setlength{\headdrop}{\@tempdima}%
1967   \setlength{\headsep}{\@tempdimb}}
1968
```

`\setheadfoot` `\setheadfoot{<headheight>}{<footskip>}` sets the headheight and the footskip.

```
1969 \newcommand{\setheadfoot}[2]{%
1970   \setlength{\headheight}{#1}%
1971   \setlength{\footskip}{#2}}
1972
```

`\setcolsepandrul` `\setcolsepandrul{<colsep>}{<thickness>}` sets the column separation and the rule thickness.

```
1973 \newcommand{\setcolsepandrul}[2]{%
1974   \setlength{\columnsep}{#1}%
1975   \setlength{\columnseprule}{#2}}
1976
```

`\setmarginnotes` `\setmarginnotes{<sep>}{<width>}{<push>}` sets the marginpar parameters.

```
1977 \newcommand{\setmarginnotes}[3]{%
1978   \setlength{\marginparsep}{#1}%
1979   \setlength{\marginparwidth}{#2}%
1980   \setlength{\marginparpush}{#3}}
1981
```

Initialise the paper size and trimming to their default values.

```
1982 \settrimmedsize{\stockheight}{\stockwidth}{*}
1983 \settrims{\z0}{\z0}
1984
```

What now follows is the standard class's method for setting up the dimensions. Set `\@tempdimb` to size-dependent initial line length and set `\@tempdima` to the maximum textwidth for the paper width, an inch margin on either side. In the standard classes the initial line length is about 14% greater than `\lxvchars`.

```

1985 \setlength{\@tempdimb}{1.14\lxvchars}
1986 \setlength\@tempdima{\paperwidth}
1987 \addtolength\@tempdima{-2in}

```

\textwidth Now set the **\textwidth** depending on the number of columns. In twocolumn mode each column should be no wider than **\@tempdimb**.

```

1988 \if@twocolumn
1989 \ifdim\@tempdima>2\@tempdimb\relax
1990 \setlength\textwidth{2\@tempdimb}
1991 \else
1992 \setlength\textwidth{\@tempdima}
1993 \fi

```

In onecolumn the text should not be wider than the minimum of the paperwidth (less 2in for the margins) and the maximum length of the character line.

```

1994 \else
1995 \ifdim\@tempdima>\@tempdimb\relax
1996 \setlength\textwidth{\@tempdimb}
1997 \else
1998 \setlength\textwidth{\@tempdima}
1999 \fi
2000 \fi

```

Adjust the width to be a whole number of points.

```

2001 \@settopoint\textwidth
2002

```

\textheight The **\textheight** is the height of the text block, excluding headers and footers. This is set according to the **\paperheight**, to an integral number of lines, and allowing a 1in margin at the top and bottom and a further 1.5in for headers and footers.

```

2003 \setlength\@tempdima{\paperheight}
2004 \addtolength\@tempdima{-3.5in}

```

Divide this height by the **\baselineskip** to get the number of lines. Then (re)calculate the **\textheight** and finally add the **\topskip**.

```

2005 \divide\@tempdima\baselineskip
2006 \@tempcnta=\@tempdima
2007 \setlength\textheight{\@tempcnta\baselineskip}
2008 \addtolength\textheight{\topskip}
2009

```

The margins are calculated.

\oddsidemargin The margins depend on the paper size, also for two sided printing the inner margin is made smaller than the outer.

```

\evensidemargin 2010 \if@twoside
2011 \setlength\@tempdima {\paperwidth}
2012 \addtolength\@tempdima {-\textwidth}
2013 \setlength\oddsidemargin {.4\@tempdima}

```

```

2014 \addtolength\oddsidemargin {-1in}
2015 \setlength\marginparwidth {.6\@tempdima}
2016 \addtolength\marginparwidth{-\marginparsep}
2017 \addtolength\marginparwidth{-0.4in}
2018 \else
2019 \setlength\@tempdima {\paperwidth}
2020 \addtolength\@tempdima {-\textwidth}
2021 \setlength\oddsidemargin {.5\@tempdima}
2022 \addtolength\oddsidemargin {-1in}
2023 \setlength\marginparwidth {.5\@tempdima}
2024 \addtolength\marginparwidth{-\marginparsep}
2025 \addtolength\marginparwidth{-0.8in} % don't know why this isn't .4
2026 \fi
2027 \ifdim\marginparwidth>2in
2028 \setlength\marginparwidth{2in}%
2029 \fi

```

Set these values to integer numbers of points, and afterwards calculate the `\evensidemargin`. Jonathon Stickel (jjstickel@vcn.com) on 2008/05/30 noted that `\marginparwidth` had to be positive for the initial setting of the sidebar geometry through `\setsidebars`.

```

2030 \@settopoint\oddsidemargin
2031 \@settopoint\marginparwidth
2032 \ifdim\marginparwidth<1pt \setlength\marginparwidth{1pt}\fi
2033
2034 \setlength\evensidemargin {\paperwidth}
2035 \addtolength\evensidemargin{-2in}
2036 \addtolength\evensidemargin{-\textwidth}
2037 \addtolength\evensidemargin{-\oddsidemargin}
2038 \@settopoint\evensidemargin

```

`\topmargin` The `\topmargin` is the distance below the top of the printable area (1in below the top of the paper) and the top of the box containing the running head.

```

2039 \setlength\topmargin {\paperheight}
2040 \addtolength\topmargin{-2in}
2041 \addtolength\topmargin{-\headheight}
2042 \addtolength\topmargin{-\headsep}
2043 \addtolength\topmargin{-\textheight}
2044 \addtolength\topmargin{-\footskip}
2045 \addtolength\topmargin{-.5\topmargin}
2046 \@settopoint\topmargin
2047

```

That is the end of the classical algorithm. Now calculate the user-friendly dimensions. The calculations are simpler than in the general case as the `\paperwidth` and `\paperheight` is the same as the `\stockwidth` and `\stockheight`.

We can get the spine and edge margins from the `\oddsidemargin`.

```

2048 \setlength{\spinemargin}{\oddsidemargin}

```

```

2049 \addtolength{\spinemargin}{1in}
2050 \setlrmargins{\spinemargin}{*}{*}
2051

```

Similarly we can get the upper and lower margins from the `\topmargin`, `\headheight` and `\headskip`.

```

2052 \setlength{\uppermargin}{\topmargin}
2053 \addtolength{\uppermargin}{1in}
2054 \addtolength{\uppermargin}{\headheight}
2055 \addtolength{\uppermargin}{\headsep}
2056 \setulmargins{\uppermargin}{*}{*}
2057

```

`\@memznegtest` DA suggested this in a private email (2003/002/13) to make error checking and reporting a bit more (space) efficient. Use like `\@memznegtest{\marginparsep}` instead of

```

\ifdim\marginparsep>\z@else
  \@memerror{\protect\marginparsep\space is zero or negative}{\@ehd}%
\fi

```

If its length variable argument is zero or less it reports an error.

```

2058 \newcommand*{\@memznegtest}[1]{%
2059   \ifdim#1>\z@else
2060     %%%% \@memerror{\protect#1\space is zero or negative}{\@ehd}%
2061     \@memwarn{\protect#1\space is zero or negative}%
2062   \fi}

```

`\@memnegtest` Reports an error if its length variable argument is negative.

```

2063 \newcommand*{\@memnegtest}[1]{%
2064   \ifdim#1<\z@
2065     %%%% \@memerror{\protect#1\space is negative}{\@ehd}%
2066     \@memwarn{\protect#1\space is negative}%
2067   \fi}
2068

```

`\m@mclassicht` The classic adjustment of the `\textheight` to get an integral number of lines (given an integral number of baselineskips returns a height giving one more line in the block).

```

2069 \newcommand*{\m@mclassicht}{%
2070   \setlength{\@tempdima}{\textheight}%
2071   \divide\@tempdima \baselineskip
2072   \@tempcnta=\@tempdima
2073   \setlength{\textheight}{\@tempcnta\baselineskip}%
2074   \addtolength{\textheight}{\topskip}}
2075

```

`\m@mmlinesht` The adjustment of the `\textheight` to get an integral number of lines (given an integral number of baselineskips returns a height giving that number of lines).

```

2076 \newcommand*{\m@mmlinesht}{%
2077   \setlength{\@tempdima}{\textheight}%
2078   \advance\@tempdima -\baselineskip
2079   \divide\@tempdima \baselineskip
2080   \@tempcnta=\@tempdima
2081   \setlength{\textheight}{\@tempcnta\baselineskip}%
2082   \addtolength{\textheight}{\topskip}}
2083

```

`\m@mnearestht` The adjustment of the `\textheight` to get an integral number of lines with the calculated height being as the closest to the given height. Algorithm supplied by Lars Madsen and Morten Høgholm on 2006/07/27.

```

2084 \newcommand*{\m@mnearestht}{%
2085   \setlength{\@tempdima}{\textheight}%
2086   \advance\@tempdima -\topskip
2087   \advance\@tempdima 0.5\baselineskip
2088   \divide\@tempdima \baselineskip
2089   \@tempcnta=\@tempdima
2090   \setlength{\textheight}{\@tempcnta\baselineskip}%
2091   \addtolength{\textheight}{\topskip}}
2092

```

`\checkthelayout` `\checkthelayout[text]` is the user level macro for checking the layout. The *text* argument controls which algorithm should be used to calculate the `\textheight`.

```

2093 \newcommand*{\checkthelayout}[1][classic]{%

```

First check the dimensions are not (zero or) negative.

```

2094   \@memnegtest{\trimege}
2095   \@memnegtest{\trimtop}
2096   \@memznegtest{\stockwidth}
2097   \@memznegtest{\paperwidth}
2098   \@memznegtest{\textwidth}
2099   %%% \@memznegtest{\spinemargin}
2100   \@memnegtest{\spinemargin}
2101   %%% \@memznegtest{\foremargin}
2102   \@memnegtest{\foremargin}
2103   \@memznegtest{\marginparsep}
2104   \@memznegtest{\marginparwidth}
2105   \@memznegtest{\stockheight}
2106   \@memznegtest{\paperheight}
2107   \@memznegtest{\textheight}
2108   %%% \@memznegtest{\uppermargin}
2109   \@memnegtest{\uppermargin}
2110   %%% \@memznegtest{\lowermargin}
2111   \@memnegtest{\lowermargin}
2112   %%% \@memznegtest{\headheight}
2113   \@memnegtest{\headheight}
2114   %%% \@memznegtest{\headsep}
2115   \@memnegtest{\headsep}

```

```

2116 %%% \@memznegtest{\footskip}
2117 \@memnegtest{\footskip}

```

Carry on regardless. We may need to adjust the `\textheight` to get an integral number of lines.

```

2118 \nametest{#1}{classic}%
2119 \ifsamename
2120 \m@mclassicht
2121 \else
2122 \nametest{#1}{lines}%
2123 \ifsamename
2124 \m@mmlinesht
2125 \else
2126 \nametest{#1}{nearest}%
2127 \ifsamename
2128 \m@mnearestht
2129 \else
2130 \nametest{#1}{fixed}
2131 \ifsamename
2132 \else% not classic, lines, nearest, or fixed
2133 \@memerror{Optional argument is not one of:\MessageBreak
2134 classic, fixed, lines, or nearest. \MessageBreak
2135 I will assume the default}%
2136 {\@ehc}%
2137 \fi
2138 \fi
2139 \fi
2140 \fi
2141 \setulmargins{\uppermargin}{*}{*}

```

Check that all the sums add up correctly, or at least to within a small (`\@tempdimb`) error.

```

2142 \@tempdimb = -1pt
2143 \@tempdima=\stockwidth
2144 \advance\@tempdima -\trimedge
2145 \advance\@tempdima -\paperwidth
2146 \ifdim\@tempdima<\@tempdimb
2147 \@tempdima = -\@tempdima
2148 \@memerror{\protect\paperwidth\space (\the\paperwidth) and/or
2149 \protect\trimedge\space (\the\trimedge)
2150 are too large for \protect\stockwidth\space (\the\stockwidth)
2151 by \the\@tempdima}%
2152 {\@ehd}
2153 \fi
2154 \@tempdima = \paperwidth
2155 \advance\@tempdima -\foremargin
2156 \advance\@tempdima -\textwidth
2157 \advance\@tempdima -\spinewidth
2158 \ifdim\@tempdima<\@tempdimb
2159 \@tempdima = -\@tempdima

```

```

2160 \memerror{\protect\spinewidth\space (\the\spinewidth) and/or
2161 \protect\textwidth\space (\the\textwidth) and/or
2162 \protect\foremargin\space (\the\foremargin)
2163 are too large for \protect\paperwidth\space (\the\paperwidth)
2164 by \the\@tempdima}%
2165 {\@ehd}
2166 \fi
2167 \@tempdima = \stockheight
2168 \advance\@tempdima -\trimtop
2169 \advance\@tempdima -\paperheight
2170 \ifdim\@tempdima<\@tempdimb
2171 \@tempdima = -\@tempdima
2172 \memerror{\protect\paperheight\space (\the\paperheight) and/or
2173 \protect\trimtop\space (\the\trimtop)
2174 are too large for \protect\stockheight\space (\the\stockheight)
2175 by \the\@tempdima}%
2176 {\@ehd}
2177 \fi
2178 \@tempdima = \paperheight
2179 \advance\@tempdima -\uppermargin
2180 \advance\@tempdima -\textheight
2181 \advance\@tempdima -\lowermargin
2182 \ifdim\@tempdima<\@tempdimb
2183 \@tempdima = -\@tempdima
2184 \memerror{\protect\uppermargin\space (\the\uppermargin) and/or
2185 \protect\textheight\space (\the\textheight) and/or
2186 \protect\lowermargin\space (\the\lowermargin)
2187 are too large for \protect\paperheight\space (\the\paperheight)
2188 by \the\@tempdima}%
2189 {\@ehd}
2190 \fi
2191 \@tempdima = \uppermargin
2192 \advance\@tempdima -\headheight
2193 \advance\@tempdima -\headsep
2194 \ifdim\@tempdima<\@tempdimb
2195 \@tempdima = -\@tempdima
2196 \memerror{\protect\headheight\space (\the\headheight) and/or
2197 \protect\headsep\space (\the\headsep)
2198 are too large for \protect\uppermargin\space (\the\uppermargin)
2199 by \the\@tempdima}%
2200 {\@ehd}
2201 \fi
2202 \@tempdima = \lowermargin
2203 \advance\@tempdima -\footskip
2204 \ifdim\@tempdima<\z@
2205 \@tempdima = -\@tempdima
2206 \memerror{\protect\footskip\space (\the\footskip)
2207 is too large for \protect\lowermargin\space (\the\lowermargin)
2208 by \the\@tempdima}%
2209 {\@ehd}

```



```
2210 \fi}
2211
```

`\fixthelayout` Calculate the normal L^AT_EX page layout parameter values. We'll do the heights first as they are independent of the number of columns and the side printing.

```
2212 \newcommand*\fixthelayout{%
2213   \topmargin = \trimtop
2214   \advance\topmargin \uppermargin
2215   \advance\topmargin -\headsep
2216   \advance\topmargin -\headheight
2217   \advance\topmargin -1in\relax
```

Now the `\oddsidemargin`.

```
2218   \oddsidemargin = \stockwidth
2219   \advance\oddsidemargin -\trimedge
2220   \advance\oddsidemargin -\paperwidth
2221   \advance\oddsidemargin \spinemargin
2222   \advance\oddsidemargin -1in\relax
```

And the `\evensidemargin`.

```
2223   \evensidemargin = \trimedge
2224   \advance\evensidemargin \foremargin
2225   \advance\evensidemargin -1in\relax
```

Set the values to the nearest whole point.

```
2226   \@settopoint\textwidth
2227   \@settopoint\oddsidemargin
2228   \@settopoint\evensidemargin
```

Fix standard page layouts after possible change of `\textwidth`.

```
2229   \fixheaderwidths}
2230
```

`\typeoutlayout` Why not type out the calculated versions of the designed values?

```
2231 \newcommand*\typeoutlayout{%
2232   \typeout{}
2233   \typeout{*****}
2234   \typeout{Stock height and width:
2235             \the\stockheight\space by \the\stockwidth}
2236   \typeout{Top and edge trims:
2237             \the\trimtop\space and \the\trimedge}
2238   \typeout{Page height and width:
2239             \the\paperheight\space by \the\paperwidth}
2240   \typeout{Text height and width:
2241             \the\textheight\space by \the\textwidth}
2242   \typeout{Spine and edge margins:
2243             \the\spinemargin\space and \the\foremargin}
2244   \typeout{Upper and lower margins:
2245             \the\uppermargin\space and \the\lowermargin}
2246   \typeout{Headheight and headsep:
2247             \the\headheight\space and \the\headsep}
```

```

2248 \typeout{Footskip:
2249         \the\footskip}
2250 \typeout{Columnsep and columnseprule:
2251         \the\columnsep\space and \the\columnseprule}
2252 \typeout{Marginparsep and marginparwidth:
2253         \the\marginparsep\space and \the\marginparwidth}
2254 \typeout{Sidecapsep and sidecapwidth:
2255         \the\sidecapsep\space and \the\sidecapwidth}
2256 \typeout{Sidebarhsep and sidebarwidth:
2257         \the\sidebarhsep\space and \the\sidebarwidth}
2258 \typeout{Sidebarvsep and sidebartopsep:
2259         \the\sidebarvsep\space and \the\sidebartopsep}
2260 \typeout{Sidebarheight:
2261         \the\dimen\sideins}
2262 \typeout{Sidefoothsep and sidefootwidth:
2263         \the\sidefoothsep\space and \the\sidefootwidth}
2264 \typeout{Sidefootvsep and sidefootheight:
2265         \the\sidefootvsep\space and \the\sidefootheight}
2266 \typeout{*****}
2267 \typeout{}}
2268

```

`\checkandfixthelayout` This macro checks and fixes the layout, and reports the result. It takes the same optional argument as `\checkthelayout`.

```

2269 % \changes{v1.61803}{2008/01/30}{Changed \cs{checkandfixthelayout}
2270 %         for the extended \cs{checkthelayout} (mempatch v4.5)}
2271 % \begin{macrocode}
2272 \newcommand*{\checkandfixthelayout}[1][classic]{%
2273 \checkthelayout[#1]%
2274 \fixthelayout
2275 \typeoutlayout}
2276

```

`\fixpdflayout` Page layout with pdf \LaTeX seems a bit iffy. At the suggestion of Lars Madsen, `\fixdvipslayout` help with setting viewer (e.g., ghostview) window sizes for dvi/ps. `\fixdvipslayout` does for dvi output as `\fixpdflayout` does for pdf output.

```

2277 \newcommand*{\fixpdflayout}{%
2278 \pdfpageheight=\the\stockheight
2279 \pdfpagewidth=\the\stockwidth
2280 \ifxetex\else
2281 \ifdim\pdfvorigin=0pt\pdfvorigin=1in\fi
2282 \ifdim\pdfhorigin=0pt\pdfhorigin=1in\fi
2283 \fi}
2284 \newcommand*{\fixdvipslayout}{%
2285 \AtBeginDvi{\special{papersize=\the\stockwidth,\the\stockheight}}}
2286
2287 \AtBeginDocument{%
2288 \ifxetex
2289 \fixpdflayout

```

```

2290 \else
2291   \ifpdf
2292     \ifnum\pdfoutput<\@ne
2293       \fixdvipslayout
2294     \else
2295       \fixpdflayout
2296     \fi
2297   \else
2298     \fixdvipslayout
2299   \fi
2300 \fi}
2301

```

With a landscape document when going `latex -> dvips` the resulting `.ps` file may appear upside down in `ghostview`. If this happens, try putting the following in the document preamble:

```

\addtodef{\fixdvipslayout}{\{%
  \special{!TeXDict begin /landplus90{true}store end }}

```

See <http://www.radicaleye.com.dvips.html> (DVIPS Home Page) for an explanation.

Some other potential specials for PostScript printing may be (at least for an HP 5SiMx LaserJet duplex printer):

```

\special{!TeXDict begin <</Duplex true>>
  setpagedevice end} % duplex
\special{!TeXDict begin <</Tumble true>>
  setpagedevice end} % short side binding

```

`\typeoutstandardlayout` Types out the current values of the standard page layout parameters.

```

2302 \newcommand{\typeoutstandardlayout}{\%
2303   \typeout{}
2304   \typeout{*****}
2305   \typeout{Page height and width:
2306     \the\paperheight\space by \the\paperwidth}
2307   \typeout{Text height and width:
2308     \the\textheight\space by \the\textwidth}
2309   \typeout{Oddsides and evensides margins:
2310     \the\oddsidemargin\space and \the\evensidemargin}
2311   \typeout{Topmargin and footskip:
2312     \the\topmargin\space and \the\footskip}
2313   \typeout{Headheight and headsep:
2314     \the\headheight\space and \the\headsep}
2315   \typeout{Columnsep and columnseprule:
2316     \the\columnsep\space and \the\columnseprule}
2317   \typeout{Marginparsep and marginparwidth:
2318     \the\marginparsep\space and \the\marginparwidth}
2319   \typeout{*****}

```

```

2320 \typeout{}
2321 }
2322

```

8.2 Some predefined layouts

A few predefined layouts for the textblock are presented. The `\checkandfixthelayout` macro *must* be called afterwards.

\medievalpage This implements Jan Tschichold's reconstruction of the page and textblock layout used by medieval scribes and the early printers []. The spine, top, edge and bottom margins are in the ratios 2:3:4:6. `\medievalpage[⟨num⟩]` positions the typeblock on the page with the margins in the above ratios. The spine margin is $(\text{page width})/\langle num \rangle$ (default 9). This must be an integer.

```

2323 %%% s = w/#1, t = 1.5s, e = 2s, f = 3s
2324 \newcommand*\medievalpage{1}[9]{%
2325   \spinemargin=\paperwidth
2326   \divide\spinemargin #1\relax
2327   \uppermargin = 1.5\spinemargin
2328   \setlrmarginsandblock{\spinemargin}{*}{2}
2329   \setulmarginsandblock{\uppermargin}{*}{2}}
2330

```

\isopage An implementation of Bringhurst's layout for ISO proportioned pages. It works for any page though. The edge margin is twice the spine, and the bottom margin is twice the top. `\isopage[⟨num⟩]` positions and sizes the typeblock on the page according to the above ratios. The spine is $(\text{page width})/\langle num \rangle$ and the top margin is $(\text{page height})/\langle num \rangle$. `⟨num⟩` must be an integer.

```

2331 % s = w/#1, e = 2s, t = h/#1, f = 2h
2332 \newcommand*\isopage{1}[9]{%
2333   \spinemargin=\paperwidth
2334   \divide\spinemargin #1\relax
2335   \uppermargin=\paperheight
2336   \divide\uppermargin #1\relax
2337   \setlrmarginsandblock{\spinemargin}{*}{2}
2338   \setulmarginsandblock{\uppermargin}{*}{2}}
2339

```

\semiisopage An variation on Bringhurst's layout for ISO proportioned pages. It works for any page though. The top margin is the same as the spine, and the edge and bottom margins are twice the spine. `\semiisopage[⟨num⟩]` positions and sizes the typeblock on the page according to the above ratios. The spine is $(\text{page width})/\langle num \rangle$. `⟨num⟩` must be an integer.

```

2340 %%% s = w/#1, t = s, e = 2s, f = e
2341 \newcommand*\semiisopage{1}[9]{%
2342   \spinemargin=\paperwidth

```

```

2343 \divide\spinemargin #1\relax
2344 \uppermargin=\spinemargin
2345 \setlrmarginsandblock{\spinemargin}{*}{2}
2346 \setulmarginsandblock{\uppermargin}{*}{2}}
2347

```

`\setpagebl` `\setpagebl{<height>}{<width>}{<ratio>}` sets a page smaller than the stock size at the bottom left of the stock. The arguments are the the same as for `\setpagetl` `\settrimmedsize`; `<height>`, `<width>` and `<ratio>` of height and width (choose any two) of the desired page size. The trims are adjusted to suit. `\setpagetl` puts the page at the top left, `\setpageml` at the middle left, and `\setpagetm` at the top middle.

```

2348 \newcommand*\setpagebl}[3]{%
2349 \settrimmedsize{#1}{#2}{#3}%
2350 \trimtop=\stockheight \advance\trimtop -\paperheight
2351 \trimedge=\stockwidth \advance\trimedge -\paperwidth}
2352 \newcommand*\setpageml}[3]{%
2353 \settrimmedsize{#1}{#2}{#3}%
2354 \trimtop=\stockheight \advance\trimtop -\paperheight
2355 \advance\trimtop -0.5\trimtop
2356 \trimedge=\stockwidth \advance\trimedge -\paperwidth}
2357 \newcommand*\setpagetl}[3]{%
2358 \settrimmedsize{#1}{#2}{#3}%
2359 \trimtop=0pt
2360 \trimedge=\stockwidth \advance\trimedge -\paperwidth}
2361 \newcommand*\setpagetm}[3]{%
2362 \settrimmedsize{#1}{#2}{#3}%
2363 \trimtop=0pt
2364 \trimedge=\stockwidth \advance\trimedge -\paperwidth
2365 \advance\trimedge -0.5\trimedge}
2366

```

`\setpagetr` Similar to those above, these macros set the page on the stock at the top right, `\setpagemr` middle right, bottom right, bottom middle, and centered.

```

\setpagebr 2367 \newcommand*\setpagetr}[3]{%
\setpagebm 2368 \settrimmedsize{#1}{#2}{#3}%
\setpagecc 2369 \trimtop=0pt
2370 \trimedge=0pt}
2371 \newcommand*\setpagemr}[3]{%
2372 \settrimmedsize{#1}{#2}{#3}%
2373 \trimtop=\stockheight \advance\trimtop -\paperheight
2374 \advance\trimtop -0.5\trimtop
2375 \trimedge=0pt}
2376 \newcommand*\setpagebr}[3]{%
2377 \settrimmedsize{#1}{#2}{#3}%
2378 \trimtop=\stockheight \advance\trimtop -\paperheight
2379 \trimedge=0pt}
2380 \newcommand*\setpagebm}[3]{%
2381 \settrimmedsize{#1}{#2}{#3}%

```

```

2382 \trimtop=\stockheight \advance\trimtop -\paperheight
2383 \trimege=\stockwidth \advance\trimege -\paperwidth
2384 \advance\trimege -0.5\trimege}
2385 \newcommand*\setpagecc}[3]{%
2386 \settrimmedsize{#1}{#2}{#3}%
2387 \trimtop=\stockheight \advance\trimtop -\paperheight
2388 \advance\trimtop -0.5\trimtop
2389 \trimege=\stockwidth \advance\trimege -\paperwidth
2390 \advance\trimege -0.5\trimege}
2391

```

8.3 Float placement parameters

All float parameters are given default values in the L^AT_EX kernel. For this reason counters only need to be set with `\setcounter` and other parameters are set using `\renewcommand`.

Limits for the placement of floating objects The settings here make it easier to place floats than with the standard classes.

`\c@topnumber` The *topnumber* counter holds the maximum number of floats that can appear on the top of a text page (classically 2)

```
2392 \setcounter{topnumber}{3}
```

`\topfraction` This indicates the maximum part of a text page that can be occupied by floats at the top (classically 0.7).

```
2393 \renewcommand{\topfraction}{.85}
```

`\c@bottomnumber` The *bottomnumber* counter holds the maximum number of floats that can appear on the bottom of a text page (classically 1).

```
2394 \setcounter{bottomnumber}{2}
```

`\bottomfraction` This indicates the maximum part of a text page that can be occupied by floats at the bottom (classically 0.3).

```
2395 \renewcommand{\bottomfraction}{.5}
```

`\c@totalnumber` This indicates the maximum number of floats that can appear on any text page (classically 3).

```
2396 \setcounter{totalnumber}{4}
```

`\textfraction` This indicates the minimum part of a text page that has to be occupied by text (classically 0.2).

```
2397 \renewcommand{\textfraction}{.1}
```

`\floatpagefraction` This indicates the minimum part of a page that has to be occupied by floating objects before a ‘float page’ is produced (classically 0.5).

```
2398 \renewcommand{\floatpagefraction}{.7}
```

`\cdbltopnumber` The *dbltopnumber* counter holds the maximum number of two column floats that can appear on the top of a two column text page (classically 2).
2399 `\setcounter{dbltopnumber}{3}`

`\dbltopfraction` This indicates the maximum part of a two column text page that can be occupied by two column floats at the top (classically 0.7).
2400 `\renewcommand{\dbltopfraction}{.85}`

`\dblfloatpagefraction` This indicates the minimum part of a page that has to be occupied by two column wide floating objects before a ‘float page’ is produced (classically 0.5).
2401 `\renewcommand{\dblfloatpagefraction}{.7}`

9 Page Styles

The page style *foo* is defined by defining the command `\ps@foo`. This command should make only local definitions. There should be no stray spaces in the definition, since they could lead to mysterious extra spaces in the output.

`\@evenhead` The `\ps@...` command defines the macros `\@oddhead`, `\@oddfoot`, `\@evenhead`, `\@oddhead` and `\@evenfoot` to define the running heads and feet—e.g., `\@oddhead` is the macro to produce the contents of the heading box for odd-numbered pages. It is called inside an `\hbox` of width `\textwidth`.

9.1 Marking conventions

To make headings determined by the sectioning commands, the page style defines the commands `\chaptermark`, `\sectionmark`, ..., where `\chaptermark{TEXT}` is called by `\chapter` to set a mark, and so on. The `\...mark` commands and the `\...head` macros are defined with the help of the following macros. (All the `\...mark` commands should be initialized to no-ops.)

L^AT_EX extends T_EX’s `\mark` facility by producing two kinds of marks, a ‘left’ and a ‘right’ mark, using the following commands:

`\markboth{LEFT}{RIGHT}`: Adds both marks.
`\markright{RIGHT}`: Adds a ‘right’ mark.
`\leftmark`: Used in the `\@oddhead`, `\@oddfoot`, `\@evenhead` or `\@evenfoot` macros, it gets the current ‘left’ mark. `\leftmark` works like T_EX’s `\botmark` command.
`\rightmark`: Used in the `\@oddhead`, `\@oddfoot`, `\@evenhead` or `\@evenfoot` macros, it gets the current ‘right’ mark. `\rightmark` works like T_EX’s `\firstmark` command.

The marking commands work reasonably well for right marks ‘numbered within’ left marks—e.g., the left mark is changed by a `\chapter` command and the right mark is changed by a `\section` command. However, it does produce somewhat anomalous results if two `\markboth`’s occur on the same page.

Commands like `\tableofcontents` that should set the marks in some page styles use a `\mkboth` command, which is `\let` by the `pagestyle` command (`\ps@...`) to `\markboth` for setting the heading or to `\gobbletwo` to do nothing.

9.2 Defining the page styles

This class provides a set of commands for the user to define new pagestyles. Essentially defining a pagestyle consists of defining the macros `\@evenhead`, `\@oddhead`, `\@evenfoot`, and `\@oddfoot`. For this class, each header and footer is treated as three parts: a left, center, and right part. In this case, defining a pagestyle consists of specifying these 12 portions of the running headers and footers. The width of the headers/footers may also be specified, rules may be drawn below the headers and/or above the footers, and the complete header and/or footer may be offset with respect to the textblock when the width is not the same as the textwidth.

In the following *<style>* is the name of a pagestyle being defined (e.g., `ruled`).

```

\mem@set@ps@extra@info The class supports an aliasing feature, where a page style name can actually call
\mem@ps@find@real      the code from another. Quite handy. But There might be problems if a use try
\mem@ps@safe@change    to modify an alias page style. We care for this by storing some information
                       about each page style, and throwing an error if the user attempts to alter a page
                       style marked as an alias.
\mem@set@ps@extra@info store two things: the name of the style we are an alias
                       for (if we are not an alias it will be blank) and an indicator whether we are an
                       alias (00 if we are and 01 if we are not).
2402 \newcommand\mem@set@ps@extra@info[3]{%
2403   \@namedef{ps@#1@aliasfor}{#2}%
2404   \@namedef{ps@#1@isalias}{#3}}
2405
\mem@ps@find@real starts at a given page style, if it is marked as an alias, it
will recursively go down the chain of aliases and save the name of the first real
page style in \@tempa.
2406 \newcommand\mem@ps@find@real[1]{%
2407   \if\@nameuse{ps@#1@isalias}\relax
2408     \mem@ps@find@real{\@nameuse{ps@#1@aliasfor}}
2409   \else\def\@tempa{#1}\fi}
2410
\mem@ps@safe@change takes a page style name, checks to see if it is defined, if it
is, it checks to see if it is safe to change it.
2411 \newcommand\mem@ps@safe@change[1]{%
2412   \ifundefined{ps@#1}{%
2413     \@memerror{Undefined pagestyle '#1', so I cannot change it}{}}{}
2414   \if\@nameuse{ps@#1@isalias}\relax \mem@ps@find@real{#1}
2415   \@memerror{The pagestyle '#1' is marked as an alias page style.^^J
2416     Modifying an alias page style may give unexpected results.^^J The
2417     alias chain resolves to the real page style '\@tempa', so try

```



```

2418     issuing^^J \string\coppagestyle{#1}{\@tempa}^^J before before
2419     modifying '1'{} \fi }
2420

```

`\makeevenhead` The command `\makeevenhead{<style>}{<left>}{<center>}{<right>}` specifies that the left, center and right portions of the even header for pagestyle `<style>` are defined as the other three arguments, respectively. Internally it defines the commands `\styleeheadl`, `\styleeheadc` and `\styleeheadr` to be `<left>`, `<center>` and `<right>` respectively.

```

2421 \newcommand{\makeevenhead}[4]{%
2422   \mem@ps@safe@change{#1}
2423   \@namedef{#1eheadl}{#2}
2424   \@namedef{#1eheadc}{#3}
2425   \@namedef{#1eheadr}{#4}
2426 }

```

`\makeoddhead` These three macros are similar to `\makeevenhead` except that they are for the oddhead, evenfoot and oddfoot.

```

\makeevenfoot \makeoddfoot 2427 \newcommand{\makeoddhead}[4]{%
2428   \mem@ps@safe@change{#1}
2429   \@namedef{#1oheadl}{#2}
2430   \@namedef{#1oheadc}{#3}
2431   \@namedef{#1oheadr}{#4}
2432 }
2433 \newcommand{\makeevenfoot}[4]{%
2434   \mem@ps@safe@change{#1}
2435   \@namedef{#1efootl}{#2}
2436   \@namedef{#1efootc}{#3}
2437   \@namedef{#1efootr}{#4}
2438 }
2439 \newcommand{\makeoddfoot}[4]{%
2440   \mem@ps@safe@change{#1}
2441   \@namedef{#1ofootl}{#2}
2442   \@namedef{#1ofootc}{#3}
2443   \@namedef{#1ofootr}{#4}
2444 }
2445

```

`\makerunningwidth` The macro `\makerunningwidth{<style>}[<ftlength>]{<length>}` sets the width of the headers and footers of pagestyle `<style>` to be `<length>`, but if `<ftlength>` is present the footer width is set to `<ftlength>`. The lengths are stored as the macros `\m@mhstyle` `\styleheadrunwidth` and `\stylefootrunwidth`. The two widths can be set individually using `\makerunningheadwidth{<style>}{<length>}` and `\makerunningfootwidth{<style>}{<length>}`. .

```

2446 \newcommand*\makerunningwidth[1]{%
2447   \mem@ps@safe@change{#1}%
2448   \def\m@mhstyle{#1}%
2449   \m@mopthfwidth}
2450 \newcommand*\m@mopthfwidth[2][\@empty]{%

```

```

2451 % \setlength\@tempdima{#2}
2452 \@namedef{\m@mhfstyle headrunwidth}{#2}%
2453 % \expandafter\edef\csname \m@mhfstyle headrunwidth\endcsname{\the\@tempdima}
2454 \ifx\@empty #1
2455   \@namedef{\m@mhfstyle footrunwidth}{#2}%
2456 % \expandafter\edef\csname \m@mhfstyle footrunwidth\endcsname{\the\@tempdima}
2457 \else
2458   \@namedef{\m@mhfstyle footrunwidth}{#1}%
2459 % \setlength\@tempdima{#1}
2460 % \expandafter\edef\csname \m@mhfstyle footrunwidth\endcsname{\the\@tempdima}
2461 \fi}
2462 \newcommand*{\makerunningheadwidth}[2]{%
2463   \setlength\@tempdima{#2}%
2464   \expandafter\edef\csname \m@mhfstyle headrunwidth\endcsname{\the\@tempdima}%
2465   \mem@ps@safe@change{#1}%
2466   \@namedef{#1headrunwidth}{#2}%
2467 }
2468 \newcommand*{\makerunningfootwidth}[2]{%
2469   \setlength\@tempdima{#2}%
2470   \mem@ps@safe@change{#1}%
2471   \@namedef{#1footrunwidth}{#2}%
2472 % \expandafter\edef\csname \m@mhfstyle footrunwidth\endcsname{\the\@tempdima}%
2473 }
2474
\normalrulethickness \normalrulethickness is the thickness of a normal horizontal or vertical rule.
\footruleheight \footruleheight is the height of a normal rule above a footer (actually zero).
\footruleskip \footruleskip is a distance sufficient to ensure that a foot rule will appear
\makeheadrule between the bottom of the textblock and above any actual footer.
\makefootrule (There was a CTT thread ngerman, fancyhdr and \footrulewidth — bug? in
December 2002 that bears on the definitions below).
2475 \newlength{\normalrulethickness}
2476 \setlength{\normalrulethickness}{0.4pt}
2477 \newcommand{\footruleheight}{0pt}
2478 \newcommand{\footruleskip}{0.3\normalbaselineskip}
The macro \makeheadrule{<style>}{<width>}{<height>} specifies the width and
height of the header rule for <style>.
Similarly \makefootrule{<style>}{<width>}{<height>}{<skip>} specifies the
width, height and skip for the footrule.
2479 \newcommand{\makeheadrule}[3]{%
2480   \mem@ps@safe@change{#1}%
2481   \@namedef{#1headrule}{%
2482     \hrule\@width #2\@height #3 \vskip-#3}}
2483 \newcommand{\makefootrule}[4]{%
2484   \mem@ps@safe@change{#1}%
2485   \@namedef{#1footrule}{%
2486     \vskip-#4\vskip-#3
2487     \hrule\@width #2\@height #3 \vskip #4}}
2488

```

`\makeheadposition` `\makeheadposition{<style>}{<ehheadpos>}{<ohheadpos>}{<efootpos>}{<ofootpos>}`
 specifies the horizontal positioning of the even and odd headers and footers,
 respectively, for the pagestyle `<style>`. Each of the `<...pos>` arguments may be
 either `flushleft`, `center` or `flushright`, with the obvious meanings. An empty
 argument (or an unrecognised one) is equivalent to `center`.

```
2489 \newcommand{\makeheadposition}[5]{%
```

```
2490   \mem@ps@safe@change{#1}%
```

Do the even head position first.

```
2491   \nametest{flushleft}{#2}
```

```
2492   \ifsamename
```

```
2493     \@namedef{#1evenhpl}{\relax} \@namedef{#1evenhpr}{\hss}
```

```
2494   \else
```

```
2495     \nametest{flushright}{#2}
```

```
2496     \ifsamename
```

```
2497       \@namedef{#1evenhpl}{\hss} \@namedef{#1evenhpr}{\relax}
```

```
2498     \else
```

```
2499       \@namedef{#1evenhpl}{\hss} \@namedef{#1evenhpr}{\hss}
```

```
2500     \fi
```

```
2501   \fi
```

And similarly for the odd head and even & odd footers.

```
2502   \nametest{flushleft}{#3}
```

```
2503   \ifsamename
```

```
2504     \@namedef{#1oddhpl}{\relax} \@namedef{#1oddhpr}{\hss}
```

```
2505   \else
```

```
2506     \nametest{flushright}{#3}
```

```
2507     \ifsamename
```

```
2508       \@namedef{#1oddhpl}{\hss} \@namedef{#1oddhpr}{\relax}
```

```
2509     \else
```

```
2510       \@namedef{#1oddhpl}{\hss} \@namedef{#1oddhpr}{\hss}
```

```
2511     \fi
```

```
2512   \fi
```

```
2513   \nametest{flushleft}{#4}
```

```
2514   \ifsamename
```

```
2515     \@namedef{#1evenfpl}{\relax} \@namedef{#1evenfpr}{\hss}
```

```
2516   \else
```

```
2517     \nametest{flushright}{#4}
```

```
2518     \ifsamename
```

```
2519       \@namedef{#1evenfpl}{\hss} \@namedef{#1evenfpr}{\relax}
```

```
2520     \else
```

```
2521       \@namedef{#1evenfpl}{\hss} \@namedef{#1evenfpr}{\hss}
```

```
2522     \fi
```

```
2523   \fi
```

```
2524   \nametest{flushleft}{#5}
```

```
2525   \ifsamename
```

```
2526     \@namedef{#1oddfpl}{\relax} \@namedef{#1oddfpr}{\hss}
```

```
2527   \else
```

```
2528     \nametest{flushright}{#5}
```

```
2529   \ifsamename
```

```

2530      \@namedef{#1oddfpl}{\hss} \@namedef{#1oddfpr}{\relax}
2531      \else
2532        \@namedef{#1oddfpl}{\hss} \@namedef{#1oddfpr}{\hss}
2533      \fi
2534    \fi}
2535

```

`\makepsmarks` The macro `\makepsmarks{<style>}{<code>}` specifies that `<code>` is the definition of the hook for pagestyle `<style>`. `\makepshook` was a deprecated version of `\makepsmarks` and is now deleted.

```

2536 \newcommand{\makepsmarks}[2]{\mem@ps@safe@change{#1}\@namedef{#1pshook}{#2}}
2537

```

`\m@mhe@dreset` To cater for potential baselineskip changes, such as `\linespread{2}`, we have to ensure they don't percolate into the header/footer. (See CTT thread *memoir + linespread*, 2004/02/11)

```

2538 \newcommand*\m@mhe@dreset{\def\baselinestretch{1}\normalsize}
2539

```

`\makeheadfootvposition` The headers and footers are typeset inside `\parbox`'es, using `\makeheadfootvposition{<style>}{<headvpos>}{<footvpost>}` users can specify whether the alignment should be b,t or c. The default is b.

```

2540 \newcommand*\makeheadfootvposition[3]{%
2541   \mem@ps@safe@change{#1}%
2542   \@namedef{#1headvplacement}{#2}\@namedef{#1footvplacement}{#3}}
2543

```

`\makepagestyle` At last we can put everything together for defining a new pagestyle, via the macro `\makepagestyle{<style>}` which initially sets up a new pagestyle `<style>` corresponding to the `\LaTeX` empty pagestyle. The prior macros can then be used to make modifications to the style. We also make sure to specify that this is a 'real' page style, and thus sets the empty alias name and the alias test to 01.

```

2544 \newcommand{\makepagestyle}[1]{%
2545   \mem@set@ps@extra@info{#1}{01}%

```

First define the macro `\ps@style`, which in turn defines the macros `\@evenhead`, `\@oddhead`, `\@evenfoot` and `\@oddfoot`.

```

2546   \@namedef{ps@#1}{%
2547     \@namedef{#1@evenhead}{%

```

The code for the definition of `\@evenhead` and friends is based on code from Piet van Oostrum's `fancyhdr` package [Oos96]. The three parts of the header are put into parboxes, with fills between them, and the whole lot is put into a box the width of the header. Fillers are put before and after the main box which control the header position.

First the left filler which is either `\relax` or `\hss`, and then start the main box.

```

2548       \@nameuse{#1evenhpl}\hb@xt@{\@nameuse{#1headrunwidth}}{\m@mhe@dreset%
2549       \vbox{\hbox{%

```

The left part of the header.

```
2550      \rlap{\parbox[\@nameuse{#1headvplacement}]{\@nameuse{#1headrunwidth}}{%
2551          \raggedright\@nameuse{#1headl}\strut}}\hfill
```

The center part of the header.

```
2552          \parbox[\@nameuse{#1headvplacement}]{\@nameuse{#1headrunwidth}}{%
2553      \centering\@nameuse{#1headc}\strut}\hfill
```

The right part of the header.

```
2554      \llap{\parbox[\@nameuse{#1headvplacement}]{\@nameuse{#1headrunwidth}}{%
2555          \raggedleft\@nameuse{#1headr}\strut}}}%
```

Finally, the header rule and finish with the right filler, which is either `\relax` or `\hss`.

```
2556      \@nameuse{#1headrule}}}\@nameuse{#1evenhpr}}%
```

The code for the `\@oddhead`, `\@evenfoot` and `\@oddfoot` follows a similar pattern. Here is `\@oddhead`.

```
2557      \namedef{#1@oddhead}{%
2558          \@nameuse{#1oddhpl}\hb@xt@\@nameuse{#1headrunwidth}{\m@mhe@dreset%
2559          \vbox{\hbox{%
2560              \rlap{\parbox[\@nameuse{#1headvplacement}]{\@nameuse{#1headrunwidth}}{%
2561                  \raggedright\@nameuse{#1oheadl}\strut}}\hfill
2562              \parbox[\@nameuse{#1headvplacement}]{\@nameuse{#1headrunwidth}}{%
2563                  \centering\@nameuse{#1oheadc}\strut}\hfill
2564              \llap{\parbox[\@nameuse{#1headvplacement}]{\@nameuse{#1headrunwidth}}{%
2565                  \raggedleft\@nameuse{#1oheadr}\strut}}}%
2566              \@nameuse{#1headrule}}}\@nameuse{#1oddhpr}}%
```

And `\@evenfoot`. For the footers the rules come *before* any foot entries.

```
2567      \namedef{#1@evenfoot}{%
2568          \@nameuse{#1evenfpl}\hb@xt@\@nameuse{#1footrunwidth}{\m@mhe@dreset%
2569          \vbox{\@nameuse{#1footrule}\hbox{%
2570              \rlap{\parbox[\@nameuse{#1footvplacement}]{\@nameuse{#1footrunwidth}}{%
2571                  \raggedright\@nameuse{#1efootl}\strut}}\hfill
2572              \parbox[\@nameuse{#1footvplacement}]{\@nameuse{#1footrunwidth}}{%
2573                  \centering\@nameuse{#1efootc}\strut}\hfill
2574              \llap{\parbox[\@nameuse{#1footvplacement}]{\@nameuse{#1footrunwidth}}{%
2575                  \raggedleft\@nameuse{#1efootr}\strut}}}%
2576              }}\@nameuse{#1evenfpr}}%
```

Lastly the `\@oddfoot`.

```
2577      \namedef{#1@oddfoot}{%
2578          \@nameuse{#1oddfpl}\hb@xt@\@nameuse{#1footrunwidth}{\m@mhe@dreset%
2579          \vbox{\@nameuse{#1footrule}\hbox{%
2580              \rlap{\parbox[\@nameuse{#1footvplacement}]{\@nameuse{#1footrunwidth}}{%
2581                  \raggedright\@nameuse{#1ofootl}\strut}}\hfill
2582              \parbox[\@nameuse{#1footvplacement}]{\@nameuse{#1footrunwidth}}{%
2583                  \centering\@nameuse{#1ofootc}\strut}\hfill
2584              \llap{\parbox[\@nameuse{#1footvplacement}]{\@nameuse{#1footrunwidth}}{%
2585                  \raggedleft\@nameuse{#1ofootr}\strut}}}%
2586              }}\@nameuse{#1oddfpr}}%
```

Now we define \@evenhead etc., in terms of #1@evenhead.

```
2587 \def\@evenhead{\@nameuse{#1@evenhead}}%
2588 \def\@oddhead{\@nameuse{#1@oddhead}}%
2589 \def\@evenfoot{\@nameuse{#1@evenfoot}}%
2590 \def\@oddfoot{\@nameuse{#1@oddfoot}}%
```

To finish off the definition of \ps@style, add in a hook which can be defined so that it adds additional code, if required.

```
2591 \@nameuse{#1pshook}}%
```

The final part of setting up the new pagestyle is defining all the macros called by \ps@style, and giving them default values. Make the headers and footers empty.

```
2592 \makeevenhead{#1}{}{}{}%
2593 \makeoddhead{#1}{}{}{}%
2594 \makeevenfoot{#1}{}{}{}%
2595 \makeoddfoot{#1}{}{}{}%
```

Make the headers/footers the same width as the \textwidth, center them, and ensure that the rules have zero height so that they will be invisible.

```
2596 \makerunningwidth{#1}{\textwidth}%
2597 \makeheadposition{#1}{}{}{}%
2598 \makeheadrule{#1}{\textwidth}{0pt}%
2599 \makefootrule{#1}{\textwidth}{\footruleheight}{\footruleskip}%
```

Set the initial vertical header and footer positions.

```
2600 \makeheadfootvposition{#1}{b}{b}%
```

Finally, there is no additional code needed, so make the hook empty, and we are done.

```
2601 \makepsmarks{#1}{}%
2602
```

\aliaspagestyle The command \aliaspagestyle{<alias>}{<original>} defines the <alias> pagestyle to be an alias for the <original> pagestyle. We remember to set the stored alias name, and sets the alias test to true (00).

```
2603 \newcommand{\aliaspagestyle}[2]{%
2604 \mem@set@ps@extra@info{#1}{#2}{00}%
2605 \@namedef{ps@#1}{\@nameuse{ps@#2}}%
2606
```

\copypagestyle The command \copypagestyle{<new>}{<original>} defines the <new> pagestyle to be a copy of the <original> pagestyle.

It first makes the <new> (empty) pagestyle then defines the internals in terms of the <original> internals.

```
2607 \newcommand{\copypagestyle}[2]{%
2608 \makepagestyle{#1}%
```

Do the headers and footers.

```
2609 \makeevenhead{#1}{\@nameuse{#2eheadl}}%
2610 \makeevenhead{#1}{\@nameuse{#2eheadc}}{\@nameuse{#2eheadr}}%
2611 \makeoddhead{#1}{\@nameuse{#2oheadl}}%
```

```

2612          {\@nameuse{#2oheadc}}{\@nameuse{#2oheadr}}}%
2613 \makeevenfoot{#1}{\@nameuse{#2efootl}}}%
2614          {\@nameuse{#2efootc}}{\@nameuse{#2efootr}}}%
2615 \makeoddfoot{#1}{\@nameuse{#2ofootl}}}%
2616          {\@nameuse{#2ofootc}}{\@nameuse{#2ofootr}}}%

```

Set the width.

```

2617 \makerunningwidth{#1}[\@nameuse{#2footrunwidth}]{\@nameuse{#2headrunwidth}}%

```

Specify the `\headposition`.

```

2618 \namedef{#1evenhpl}{\@nameuse{#2evenhpl}}}%
2619 \namedef{#1oddhpl}{\@nameuse{#2oddhpl}}}%
2620 \namedef{#1evenhpr}{\@nameuse{#2evenhpr}}}%
2621 \namedef{#1oddhpr}{\@nameuse{#2oddhpr}}}%

```

Also vertically

```

2622 \makeheadfootvposition{#1}{\@nameuse{#2headvplacement}}{\@nameuse{#2footvplacement}}%

```

Specify the feet

```

2623 \namedef{#1evenfpl}{\@nameuse{#2evenfpl}}}%
2624 \namedef{#1oddfpl}{\@nameuse{#2oddfpl}}}%
2625 \namedef{#1evenfpr}{\@nameuse{#2evenfpr}}}%
2626 \namedef{#1oddfpr}{\@nameuse{#2oddfpr}}}%

```

Specify the head and foot rules.

```

2627 \namedef{#1headrule}{\@nameuse{#2headrule}}}%
2628 \namedef{#1footrule}{\@nameuse{#2footrule}}}%

```

And pick up the hook.

```

2629 \makepsmarks{#1}{\@nameuse{#2pshook}}}%
2630

```

`\ifonlyfloats` `\ifonlyfloats{<yes>}{<no>}` processes `<yes>` if the macro is called on a page consisting only of floats, otherwise `<no>` is processed. `\if@fcolmade` is specified in `ltoutput.dtx`.

```

2631 \newcommand{\ifonlyfloats}[2]{\if@fcolmade #1\else #2\fi}
2632

```

`\mergepagefloatstyle` `\mergepagefloatstyle{<style>}{<textstyle>}{<floatstyle>}` makes a new page style `<style>` that is `<textstyle>` on normal pages but uses `<floatstyle>` on float-only pages. Both `<textstyle>` and `<floatstyle>` must have been previously defined.

```

2633 \newcommand{\mergepagefloatstyle}[3]{%

```

Make sure that the two styles have been called, otherwise we get some undefined errors.

```

2634 \@nameuse{ps@#3}\@nameuse{ps@#2}}%

```

Specify the pagestyle's headers and footers.

```

2635 \namedef{ps@#1}{%
2636 \def\@evenhead{\ifonlyfloats{\@nameuse{#3@evenhead}}}%
2637 \def\@oddhead{\ifonlyfloats{\@nameuse{#3@oddhead}}}%
2638

```

```

2639          {\@nameuse{#2@oddhead}}}%
2640 \def\@evenfoot{\ifonlyfloats{\@nameuse{#3@evenfoot}}}%
2641          {\@nameuse{#2@evenfoot}}}%
2642 \def\@oddfoot{\ifonlyfloats{\@nameuse{#3@oddfoot}}}%
2643          {\@nameuse{#2@oddfoot}}}%

```

Set the hook to the `<textstyle>` on the assumption that that is more complex than required for a float page.

```

2644 \namedef{#1pshook}{\@nameuse{#2pshook}}%

```

That's it.

```

2645 }}
2646

```

The pagestyles *empty* and *plain* are defined in `latex.dtx`. However, I will redefine them here, just in case someone takes a fancy to modifying them.

`\ps@empty` The *empty* pagestyle is simple, it's just what we get when we call `\makepagestyle`.

```

2647 \makepagestyle{empty}
2648

```

`\ps@plain` The *plain* pagestyle is also simple, it just puts the page number at the bottom middle of the page. We call `\makepagestyle{plain}` and then adjust as required.

```

2649 \makepagestyle{plain}
2650 \makeevenfoot{plain}{\thepage}{}
2651 \makeoddfoot{plain}{\thepage}{}
2652

```

`\ps@simple` The *simple* page style simply puts the page number in the header at the outer margin.

```

2653 \makepagestyle{simple}
2654 \makeevenhead{simple}{\thepage}{}
2655 \makeoddhead{simple}{\thepage}{}
2656

```

`\nouppercaseheads` Spurred by Lars Madsen's `memexsupp v0.5` package here is a way of switching `\uppercaseheads` uppercasing in the headings pagestyle.

```

\memUChead 2657 \newcommand*\nouppercaseheads{\let\memUChead\relax}
2658 \newcommand*\uppercaseheads{\let\memUChead\MakeUppercase}
2659 \uppercaseheads
2660

```

The macros `\createplainmark`, `\createmark` and `\addtopsmarks` are modified versions of code supplied by Lars Madsen.

`\createplainmark` `\createplainmark{<type>}{<marks>}{<text>}` creates a `\typemark` (`<type>` is `toc`, `bib`, etc) with `<marks>` (left, both, right) whose contents are `<text>`. For example `\createplainmark{toc}{both}{\contentsname}`. The difference

between plain and regular marks, is that plain marks use a fixed text, whereas regular marks receive dynamic text and thus is given an argument.

```

2661 \newcommand*\createplainmark}[3]{%
2662   \nametest{#2}{left}%
2663   \ifsamename
2664     \@namedef{#1mark}{\markboth{\memUHead{#3}}{}}%
2665   \else
2666     \nametest{#2}{right}%
2667     \ifsamename
2668       \@namedef{#1mark}{\markright{\memUHead{#3}}}%
2669     \else
2670       \nametest{#2}{both}%
2671       \ifsamename\else
2672         \@memerror{%
2673           Unknown mark setting type ‘#2’ for #1mark}%
2674           I expected ‘left’, ‘both’ or ‘right’. \MessageBreak
2675           I will assume you meant ‘both’}%
2676       \fi
2677       \@namedef{#1mark}{\markboth{\memUHead{#3}}{\memUHead{#3}}}%
2678     \fi
2679   \fi}
2680

```

`\createmark` `\createmark{<division>}{<marks>}{<show>}{<prefix>}{<postfix>}` creates a `\divisionmark` with `<marks>` (= left, both or right) controlling which marks are set, `<show>` (= shownumber or nonnumber) controls the display of the division number in `\mainmatter`, `<prefix>` is text before the number and `<postfix>` is text after the number.

```

2681 \newcommand\createmark[5]{%
2682   Handle (show/no)number, fails to shownumber.
2683   \def\@tempa{00}
2684   \nametest{#3}{nonnumber}%
2685   \ifsamename
2686     \def\@tempa{01}%
2687   \else
2688     \nametest{#3}{shownumber}
2689     \ifsamename\else
2690       \@memerror{Unknown numbering value ‘#3’ for #1mark}%
2691       {I expected ‘shownumber’ or ‘nonnumber’. \MessageBreak
2692       I will assume you meant ‘shownumber’}%
2693     \fi
2694   \fi
2695   \expandafter\if\@tempa%           compares the two \@tempa digits
2696     \@namedef{#1marksn}##1{##1}%
2697   \else
2698     \@namedef{#1marksn}{\@gobble}%
2699   \fi

```

The three `<marks>` cases, left, both and right; fails to both.

```

2699 \nametest{#2}{left}%
2700 \ifsamename
2701   \@namedef{#1mark}##1{%
2702     \setclcnt{#1}{@memmarkcntra}%
2703     \advance\c@@memmarkcntra\m@ne
2704     \markboth{%
2705       \memUChead{%
2706         \ifnum \c@secnumdepth > \c@@memmarkcntra
2707           \if@mainmatter
2708             \@nameuse{#1marksn}{#4\@nameuse{the#1}#5}%
2709           \fi
2710         \fi
2711       ##1}}}%
2712 \else
2713   \nametest{#2}{right}
2714   \ifsamename
2715     \@namedef{#1mark}##1{%
2716       \setclcnt{#1}{@memmarkcntra}
2717       \advance\c@@memmarkcntra\m@ne
2718       \markright{%
2719         \memUChead{%
2720           \ifnum \c@secnumdepth > \c@@memmarkcntra
2721             \if@mainmatter%
2722               \@nameuse{#1marksn}{#4\@nameuse{the#1}#5}%
2723             \fi%
2724           \fi%
2725         ##1}}}%
2726 \else
2727   \nametest{#2}{both}%
2728   \ifsamename\else
2729     \@memerror{%
2730       Unknown mark setting type ‘#2’ for #1mark}%
2731     I expected ‘left’, ‘both’ or ‘right’. \MessageBreak
2732     I will assume you meant ‘both’}%
2733   \fi
2734   \@namedef{#1mark}##1{%
2735     \setclcnt{#1}{@memmarkcntra}
2736     \advance\c@@memmarkcntra\m@ne
2737     \markboth{%
2738       \memUChead{%
2739         \ifnum \c@secnumdepth > \c@@memmarkcntra
2740           \if@mainmatter
2741             \@nameuse{#1marksn}{#4\@nameuse{the#1}#5}%
2742           \fi
2743         \fi
2744       ##1}}{%
2745     \memUChead{%
2746       \ifnum \c@secnumdepth > \c@@memmarkcntra
2747         \if@mainmatter
2748           \@nameuse{#1marksn}{#4\@nameuse{the#1}#5}%

```

```

2749         \fi
2750         \fi
2751         ##1}}}%
2752     \fi
2753     \fi}
2754
\addtopsmarks   \addtopsmarks{<pagestyle>}{<prepend>}{<append>} inserts <prepend> and
                <append> at the start and end of the current definition of \pagestylepshook.
2755 \newcommand\addtopsmarks[3]{%
2756     \mem@ps@safe@change{#1}%
2757     \expandafter\addto\def\expandafter{\csname #1pshook\endcsname}{#2}{#3}}

\clearplainmark For some page styles it is handy to be able to make sure that no section or
\clearmark      otherwise add data to the \left- and \rightmark. The macros
                \clearplainmark and \clearmark will take one argument specifying which
                mark to clear, and then redefine this mark to do nothing or to gobble its given
                argument.
2758 \newcommand\clearplainmark[1]{%
2759     \@namedef{#1mark}{}}
2760 \newcommand\clearmark[1]{%
2761     \@namedef{#1mark}{\@gobble}}

\ps@headings    The headings pagestyle behaves differently for twosided and onesided printing.
                This is a rewrite of the standard style.
2762 \if@twoside
                The footer is empty and the header contains the page number and one of the
                marks.
2763     \makepagestyle{headings}
2764     \makepsmarks{headings}{%
2765         \def\chaptermark##1{%
2766             \markboth{\memUHead{%
2767                 \ifnum \c@secnumdepth > \m@ne
2768                     \if@mainmatter
2769                         \@chapapp\ thechapter. \ %
2770                     \fi
2771                 \fi
2772                 ##1}}}%
2773         \def\tocmark{\markboth{\memUHead{\contentsname}}{\memUHead{\contentsname}}}%
2774         \def\lofmark{\markboth{\memUHead{\listfigurename}}{\memUHead{\listfigurename}}}%
2775         \def\lotmark{\markboth{\memUHead{\listtablename}}{\memUHead{\listtablename}}}%
2776         \def\bibmark{\markboth{\memUHead{\bibname}}{\memUHead{\bibname}}}%
2777         \def\indexmark{\markboth{\memUHead{\indexname}}{\memUHead{\indexname}}}%
2778         \def\sectionmark##1{%
2779             \markright{\memUHead{%
2780                 \ifnum \c@secnumdepth > \z@
2781                     \thesection. \ %
2782                 \fi
2783                 ##1}}}%

```

And here's a version of the above `\makepsmarks` using `\createmark` and `\createplainmark`.

```

2784 \makepsmarks{headings}{%
2785 \createmark{chapter}{left}{shownumber}{\@chapapp\ }{. \ }
2786 \createmark{section}{right}{shownumber}{\ }{. \ }
2787 \createplainmark{toc}{both}{\contentsname}
2788 \createplainmark{lof}{both}{\listfigurename}
2789 \createplainmark{lot}{both}{\listtablename}
2790 \createplainmark{bib}{both}{\bibname}
2791 \createplainmark{index}{both}{\indexname}
2792 \createplainmark{glossary}{both}{\glossaryname}
2793 }
2794 \makeevenhead{headings}{\thepage}{\slshape\leftmark}
2795 \makeoddhead{headings}{\slshape\rightmark}{\thepage}
2796 \else

```

For one sided printing even and odd pages are treated the same, so no need to bother with the evenhead, and just the `\rightmark` is used.

```

2797 \makepagestyle{headings}
2798 \makepsmarks{headings}{%
2799 \def\chaptermark##1{%
2800 \markright{\memUHead{%
2801 \ifnum \c@secnumdepth >\m@ne
2802 \if@mainmatter
2803 \@chapapp\ thechapter. \ %
2804 \fi
2805 \fi
2806 ##1}}}%
2807 \def\tocmark{\markright{\memUHead{\contentsname}}}%
2808 \def\lofmark{\markright{\memUHead{\listfigurename}}}%
2809 \def\lotmark{\markright{\memUHead{\listtablename}}}%
2810 \def\bibmark{\markright{\memUHead{\bibname}}}%
2811 \def\indexmark{\markright{\memUHead{\indexname}}}%

```

And here's a version of the above `\makepsmarks` using `\createmark` and `\createplainmark`.

```

2812 \makepsmarks{headings}{%
2813 \createmark{chapter}{right}{shownumber}{\@chapapp\ }{. \ }
2814 \createplainmark{toc}{right}{\contentsname}
2815 \createplainmark{lof}{right}{\listfigurename}
2816 \createplainmark{lot}{right}{\listtablename}
2817 \createplainmark{bib}{right}{\bibname}
2818 \createplainmark{index}{right}{\indexname}
2819 \createplainmark{glossary}{right}{\glossaryname}
2820 }
2821 \makeoddhead{headings}{\slshape\rightmark}{\thepage}
2822 \fi
2823

```

`\ps@myheadings` The *myheadings* page style is simpler because the user has to specify the

contents using `\markboth` and `\markright` commands. This is the definition used in the standard classes.

```

}
\newcommand{\ps@myheadings}{%
  \let\@oddfoot\@empty\let\@evenfoot\@empty
  \def\@evenhead{\thepage\hfil{\slshape\leftmark}}%
  \def\@oddhead{\slshape\rightmark}\hfil\thepage}%
  \let\@mkboth\@gobbletwo
  \let\chaptermark\@gobble
  \let\sectionmark\@gobble
}

```

Translating that into our terms we get:

```

2824 \makepagestyle{myheadings}
2825   \makepsmarks{myheadings}{%
2826     \let\chaptermark\@gobble
2827     \let\sectionmark\@gobble
2828     \def\tocmark{}%
2829     \def\lofmark{}%
2830     \def\lotmark{}%
2831     \def\bibmark{}%
2832     \def\indexmark{}%
2833     \def\glossarymark{}%
2834     \makeevenhead{myheadings}{\thepage}{\slshape\leftmark}
2835     \makeoddhead{myheadings}{\slshape\rightmark}{\thepage}
2836

```

(Kai von Fintel (fintel@mit.edu) emailed me on 2003/02/24 saying that the original `\let\tocmark\@gobble` in the `myheadings` `pagestyle` did nasty things to the ToC, etc. Now using `\def\tocmark{}`.)

`\ps@chapter` The standard classes use the *plain* `pagestyle` for the first page of a chapter. This

`\ps@part` class uses the *chapter* instead, which is aliased to *plain*. Similarly for parts.

`\ps@cleared` Further, `\cleardoublepage` uses whatever `pagestyle` is in effect for the empty verso page. I find that this looks odd if the header contains a chapter name. This class uses the *cleared* `pagestyle` in this case. I have aliased this to *empty*; the *plain* would be another reasonable choice.

```

2837 \aliaspagestyle{chapter}{plain}
2838 \aliaspagestyle{part}{plain}
2839 \aliaspagestyle{cleared}{empty}
2840

```

`\cleardoublepage` A slight adjustment to the kernel definition to set a `pagestyle`.

```

2841 \def\cleardoublepage{\clearpage\if@twoside \ifodd\c@page\else
2842   \hbox{}\thispagestyle{cleared}%
2843   \newpage\if@twocolumn\hbox{}\newpage\fi\fi\fi}
2844

```

`\ps@ruled` Partly to show how it is done, the class provides a *ruled* pagestyle. In this style the headers and footers are the same width as the textblock, there is a rule under the header, page numbers are set in the footers at the outside of the page. Even page headers have the chapter number and title at the left, and odd page headers have the section title at the right. Start by making the (empty) *ruled* pagestyle.

```
2845 \makepagestyle{ruled}
```

There is no need to change the default width (which is the `\textwidth`), nor the default positions (centered), nor to make the footrule visible. We do, though, have to put the page numbers into the footers.

```
2846 \makeevenfoot{ruled}{\thepage}{\thepage}
```

```
2847 \makeoddfoot{ruled}{\thepage}{\thepage}
```

Make the header rule visible and equal to the `\textwidth`.

```
2848 \makeheadrule{ruled}{\textwidth}{\normalrulethickness}
```

`\@ruledmarks` We have to make sure that the `\chapter` and `\section` commands make the appropriate marks for use in the headers. We use the hook for this. Note that contrary to normal L^AT_EX practice, the titles are not automatically upper-cased. The marks for the tocbibinds also need adjusting.

```
2849 \newcommand{\@ruledmarks}{%
```

```
2850   \def\chaptermark##1{%
```

```
2851     \markboth{%
```

```
2852       \ifnum \c@secnumdepth > \m@ne
```

```
2853         \if@mainmatter
```

```
2854           \thechapter. \%
```

```
2855         \fi
```

```
2856       \fi
```

```
2857     ##1}{}}
```

```
2858   \def\sectionmark##1{\markright{##1}}
```

```
2859   \def\tocmark{\markboth{\contentsname}{}}
```

```
2860   \def\lofmark{\markboth{\listfigurename}{}}
```

```
2861   \def\lotmark{\markboth{\listtablename}{}}
```

```
2862   \def\bibmark{\markboth{\bibname}{}}
```

```
2863   \def\indexmark{\markboth{\indexname}{}}
```

```
2864   \def\glossarymark{\markboth{\glossaryname}{}}
```

```
2865 }
```

And here's a version using `\createmark` and friends.

```
2866 \renewcommand*{\@ruledmarks}{%
```

```
2867   \nouppercaseheads
```

```
2868   \createmark{chapter}{left}{\shownumber}{\space}
```

```
2869   \createmark{section}{right}{\shownumber}{\space}
```

```
2870   \createplainmark{toc}{both}{\contentsname}
```

```
2871   \createplainmark{lof}{both}{\listfigurename}
```

```
2872   \createplainmark{lot}{both}{\listtablename}
```

```
2873   \createplainmark{bib}{both}{\bibname}
```

```
2874   \createplainmark{index}{both}{\indexname}
```

```
2875   \createplainmark{glossary}{both}{\glossaryname}}
```

```
2876 \makepsmarks{ruled}{\@ruledmarks}
```

We can now define the even page header which is to have the chapter title at the left. As the chapter mark did no upper-casing we will print it using small caps, but just use the normal font for section title on the odd page header.

```
2877 \makeevenhead{ruled}{\scshape\leftmark}{\leftmark}
2878 \makeoddhead{ruled}{\rightmark}{\rightmark}
2879
```

This is all that we need to do for the *ruled* pagestyle.

\ps@Ruled Also define a *Ruled* pagestyle similar to *ruled* except that the headers and footers are 10% wider than the textblock, sticking out into the fore edge.

```
2880 \makepagestyle{Ruled}
2881 \makerunningwidth{Ruled}{1.1\textwidth}
2882 \makeheadposition{Ruled}{flushright}{flushleft}{flushright}{flushleft}
2883 \makeevenfoot{Ruled}{\thepage}{\thepage}
2884 \makeoddfoot{Ruled}{\thepage}{\thepage}
2885 \makeheadrule{Ruled}{1.1\textwidth}{\normalrulethickness}
2886 \makepsmarks{Ruled}{\@ruledmarks}
2887 \makeevenhead{Ruled}{\scshape\leftmark}{\leftmark}
2888 \makeoddhead{Ruled}{\rightmark}{\rightmark}
2889
```

\headwidth A *companion* pagestyle like the one in the *L^AT_EX Companion* series. We need the **\ps@companion** `\headwidth` length for this.

```
2890 \newlength{\headwidth}
2891
2892 \makepagestyle{companion}
2893 \setlength{\headwidth}{\textwidth}
2894 \addtolength{\headwidth}{\marginparsep}
2895 \addtolength{\headwidth}{\marginparwidth}
2896 \makerunningwidth{companion}{\headwidth}
2897 \makeheadrule{companion}{\headwidth}{\normalrulethickness}
2898 \makeheadposition{companion}{flushright}{flushleft}{\leftmark}{\rightmark}
2899 \makepsmarks{companion}{%
2900   \def\chaptermark##1{\markboth{##1}{##1}}      % left mark & right marks
2901   \def\sectionmark##1{\markright{##1}}
2902   \ifnum \c@secnumdepth>\z@
2903     \thesection. \ %
2904   \fi
2905   ##1}}
2906 \def\tocmark{\markboth{\contentsname}{\contentsname}}
2907 \def\lofmark{\markboth{\listfigurename}{\listfigurename}}
2908 \def\lotmark{\markboth{\listtablename}{\listtablename}}
2909 \def\bibmark{\markboth{\bibname}{\bibname}}
2910 \def\indexmark{\markboth{\indexname}{\indexname}}
```

And here's a version of the above `\makepsmarks` using `\createmark` and `\createplainmark`.

```

2911 \makepsmarks{companion}{%
2912   \nouppercaseheads
2913   \createmark{chapter}{both}{nonumber}{}{. \space}
2914   \createmark{section}{right}{shownumber}{}{. \space}
2915   \createplainmark{toc}{both}{\contentsname}
2916   \createplainmark{lof}{both}{\listfigurename}
2917   \createplainmark{lot}{both}{\listtablename}
2918   \createplainmark{bib}{both}{\bibname}
2919   \createplainmark{index}{both}{\indexname}
2920   \createplainmark{glossary}{both}{\glossaryname}}
2921 \makeevenhead{companion}{\normalfont\bfseries\thepage}{}%
2922   {\normalfont\bfseries\leftmark}
2923 \makeoddhead{companion}{\normalfont\bfseries\rightmark}{}%
2924   {\normalfont\bfseries\thepage}
2925

```

Another pagestyle called *showlocs* can be used to show the locations of the header, footer and textblock. I would expect that this would be mainly used as an example for authors to create their own similar styles.

```

\ifshowheadfootloc  Booleans controlling the appearance, or not, of the header/footer lines and text
\showheadfootloctrue  frame in showlocs.
\showheadfootlocfalse 2926 \newif\ifshowheadfootloc
\showheadfootlocon 2927   \showheadfootloctrue
\showheadfootlocoff 2928 \newcommand*{\showheadfootlocon}{\showheadfootloctrue}
\ifshowtextblockloc 2929 \newcommand*{\showheadfootlocoff}{\showheadfootlocfalse}
\showtextblockloctrue 2930 \newif\ifshowtextblockloc
\showtextblocklocfalse 2931   \showtextblockloctrue
\showtextblocklocon 2932 \newcommand*{\showtextblocklocon}{\showtextblockloctrue}
\showtextblocklocoff 2933 \newcommand*{\showtextblocklocoff}{\showtextblocklocfalse}
2934

```

\framepichead For producing a zero-sized picture of a line at the base of the header. It is meant to be used as the left part of the header for a pagestyle.

```

2935 \newcommand*{\framepichead}{%
2936 \ifshowheadfootloc
2937   \begin{picture}(0,0)
2938     \unitlength 1pt
2939     \put(0,0){\line(1,0){\strip@pt\textwidth}}
2940   \end{picture}%
2941 \fi}
2942

```

\framepictextfoot For producing a zero-sized picture of a line at the base of the footer and a frame around the text block. It is meant to be used as the left part of a footer for a pagestyle.

```

2943 \newcommand*{\framepictextfoot}{%
2944   \begin{picture}(0,0)
2945     \unitlength 1pt

```



```

2946 \ifshowheadfootloc
2947 \put(0,0){\line(1,0){\strip@pt\textwidth}}
2948 \fi
2949 \ifshowtextblockloc
2950 \put(0,\strip@pt\footskip)%
2951 {\framebox(\strip@pt\textwidth,\strip@pt\textheight){}}
2952 \fi
2953 \end{picture}}
2954

```

`\ps@showlocs` The *showlocs* pagestyle. This is more to show what can be done.

```

2955 \makepagestyle{showlocs}
2956 \makeevenhead{showlocs}{\framepichead\thepage}{\thepage}{\thepage}
2957 \makeoddhead{showlocs}{\framepichead\thepage}{\thepage}{\thepage}
2958 \makeevenfoot{showlocs}{\framepictextfoot\thepage}{\thepage}{\thepage}
2959 \makeoddfoot{showlocs}{\framepictextfoot\thepage}{\thepage}{\thepage}
2960

```

`\fixheaderwidths` The companion pagestyle, at least, needs adjusting (at `\fixthelayout` time) if the `\textwidth` has changed from its initial value.

```

2961 \newcommand*{\fixheaderwidths}{%
2962 % companion pagestyle
2963 \setlength{\headwidth}{\textwidth}
2964 \addtolength{\headwidth}{\marginparsep}
2965 \addtolength{\headwidth}{\marginparwidth}
2966 \makerunningwidth{companion}{\headwidth}
2967 \makeheadrule{companion}{\headwidth}{\normalrulethickness}
2968 \makefootrule{companion}{\textwidth}{\footruleheight}{\footruleskip}
2969 }
2970

```

9.3 Page numbering

The kernel includes the `\pagenumbering` command for setting the style (arabic, roman, etc.) of the page numbers, and at the same time it resets the page counter. I want a version that resets the style but not the number¹³.

`\pagenumbering` `\pagenumbering{style}` is the normal version whereas the starred version does not reset the counter.

```

2971 \renewcommand{\pagenumbering}{%
2972 \@ifstar{\@smempnum}{\@mempnum}}

```

`\@smempnum`

```

2973 \newcommand{\@smempnum}[1]{%
2974 \gdef\thepage{\csname @#1\endcsname \c@page}}

```

¹³Added to meet a request by Daniel Richard G. (skunk@mit.edu) September 2001.

```

\@mempnum
2975 \newcommand{\@mempnum}[1]{%
2976   \@smempnum{#1}\global\c@page \@ne}
2977

\c@storedpagenumber A counter to store the page number.
2978 \newcounter{storedpagenumber}
2979 \setcounter{storedpagenumber}{1}

\savepagenumber \savepagenumber saves the current page number and \restorepagenumber sets
\restoregenumber the page number to the stored value.
2980 \newcommand{\savepagenumber}{\global\c@storedpagenumber \c@page}
2981 \newcommand{\restorepagenumber}{\global\c@page \c@storedpagenumber}
2982

```

10 Non-traditional spacing

10.1 Double spacing

This is an embedding and extension of the code from the `setspace` package, with names changed.

```

\setSpacing \setSpacing{<num>} effectively increases the \baselineskip to
<num>*\baselineskip. (In package was \setstretch).
2983 \newcommand*\setSpacing[1]{%
2984   \def\baselinestretch{#1}%
2985   \@currsizet}
2986

\setSingleSpace \setSingleSpace{<num>} effectively increases the \baselineskip for single
spacing to <num>*\baselineskip (<num> should be close to 1.0). (In package
was \SetSinglespace).
2987 \newcommand*\setSingleSpace[1]{%
2988   \def\m@m@singlespace{#1}}
2989 \setSingleSpace{1}
2990

\SingleSpacing \SingleSpacing starts single spacing. (In package was \singlespacing).
2991 \newcommand*\SingleSpacing{%
2992   \setSpacing{\m@m@singlespace}%
2993   \vskip\baselineskip% correction for coming into single spacing
2994 }
2995 \SingleSpacing
2996

\OnehalfSpacing \OnehalfSpacing starts ‘one and a half’ spacing, which to most thesis nitpickers
will look like double spacing. (In package was \onehalfspacing).

```

```

2997 \newcommand*{\OnehalfSpacing}{
2998   \setSpacing{1.25}% default (10pt)
2999   \ifcase \@ptsize \relax   % 10pt
3000     \setSpacing{1.25}%
3001   \or% 11pt
3002     \setSpacing{1.213}%
3003   \or% 12pt
3004     \setSpacing{1.241}%
3005   \or\or% 14pt
3006     \setSpacing{1.20}%
3007   \or\or\or% 17pt
3008     \setSpacing{1.16}%
3009   \or\or% 9pt
3010     \setSpacing{1.35}%
3011   \else% the extended sizes
3012     \setSpacing{1.16}%
3013   \fi}
3014

```

`\DoubleSpacing` `\DoubleSpacing` starts double spacing, which to most thesis nitpickers will look far too spaced out. (In package was `\doublespacing`).

```

3015 \newcommand*{\DoubleSpacing}{
3016   \setSpacing{1.667}% default (10pt)
3017   \ifcase \@ptsize \relax   % 10pt
3018     \setSpacing{1.667}%
3019   \or% 11pt
3020     \setSpacing{1.618}%
3021   \or% 12pt
3022     \setSpacing{1.655}%
3023   \or\or% 14pt
3024     \setSpacing{1.60}%
3025   \or\or\or% 17pt
3026     \setSpacing{1.545}%
3027   \or\or% 9pt
3028     \setSpacing{1.8}%
3029   \else%      larger sizes
3030     \setSpacing{1.5}%
3031   \fi}
3032

```

`\@setsize` Modify the kernel command. check what this does!!!!!!

```

3033 \renewcommand*{\@setsize}[4]{%
3034   \@nomath#1%
3035   \let\@currsiz#1%
3036   \baselineskip #2%
3037   \baselineskip \baselinestretch\baselineskip
3038   \parskip \baselinestretch\parskip
3039   \setbox\strutbox \hbox{%
3040     \vrule height.7\baselineskip
3041     depth .3\baselineskip

```

```

3042         width \z@}%
3043 \skip\footins \baselinestretch\skip\footins
3044 \normalbaselineskip\baselineskip#3#4}
3045

```

SingleSpace Environment form of `\SingleSpacing`. (In package was `singlespace`).

```

3046 \newenvironment{SingleSpace}{%
3047 \vskip\baselineskip
3048 \setSpacing{\m@m@singlespace}%
3049 \vskip -\baselineskip
3050 }\par}
3051

```

SingleSpace* Don't use this; it's only here to match the `setspace` package. (In package was `singlespace*`).

```

3052 \newenvironment{SingleSpace*}{%
3053 %% \vskip\baselineskip
3054 \setSpacing{\m@m@singlespace}%
3055 \vskip 0.5\baselineskip
3056 }\vskip -0.5\baselineskip}
3057

```

`\m@mrestore@spacing`

```

3058 \newcommand*{\m@mrestore@spacing}{%
3059 \par
3060 \vskip \parskip
3061 \vskip \baselineskip
3062 \endgroup
3063 \vskip -\parskip
3064 \vskip -\baselineskip}
3065

```

Spacing `\begin{Spacing}{num}` increases the `\baselineskip` to `num*\baselineskip`. (In package was `spacing`).

```

3066 \newenvironment{Spacing}[1]{%
3067 \par
3068 \begingroup
3069 \setSpacing{#1}}{\m@mrestore@spacing}
3070

```

OnehalfSpace Environment form of `\OnehalfSpacing`. (In package was `onehalfspace`).

```

3071 \newenvironment{OnehalfSpace}{%
3072 \begingroup
3073 \OnehalfSpacing}{\m@mrestore@spacing}
3074

```

DoubleSpace Environment form of `\DoubleSpacing`. (In package was `doublespace`).

```

3075 \newenvironment{DoubleSpace}{%
3076 \begingroup

```

```

3077 \DoubleSpacing}{\m@mrestore@spacing}
3078

```

Deal with spacing around displays.

```

\memdskipstretch \setDisplayskipStretch{<num>} changes space around displays by the factor
\setDisplayskipStretch (1+<num>). \noDisplaydkipStretch keeps the regular spacing around displays.
\noDisplayskipStretch (In package were \setdisplayskipstretch and the nodisplayskipstretch
option).

```

```

3079 \newcommand*{\memdskipstretch}{0.0}
3080 \newcommand*{\setDisplayskipStretch}[1]{%
3081 \renewcommand*{\memdskipstretch}{#1}}
3082 \newcommand*{\noDisplayskipStretch}{\setDisplayskipStretch{0.0}}
3083

```

`\memdskips` Macro added to the kernel hook `\everydisplay`, changing the settings of `displayskip`s

```

3084 \newcommand*{\memdskips}{%
3085 \advance\abovedisplayskip \memdskipstretch\abovedisplayskip
3086 \advance\belowdisplayskip \memdskipstretch\belowdisplayskip
3087 \advance\abovedisplayshortskip \memdskipstretch\abovedisplayshortskip
3088 \advance\belowdisplayshortskip \memdskipstretch\belowdisplayshortskip}

```

`\everydisplay`

```

3089 \everydisplay\expandafter{%
3090 \the\everydisplay
3091 \memdskips}
3092

```

`\@xfloat` Reset `\baselinestretch` in floats.

```

3093 \let\m@m@xfloat\@xfloat
3094 \def\@xfloat #1[#2]{%
3095 \m@m@xfloat #1[#2]%
3096 \def\baselinestretch{\m@m@singlespace}%
3097 \normalsize}
3098

```

The extra spacing does not add space before and after a minipage. The solution, hinted at by Donald Arseneau, is to create a new environment. I came up with a solution which DA then much improved upon (CTT *setspace and minipages*, 2006/11/28). This is like minipage from an author's view, except that it is always top positioned and acts like a paragraph.

`\memPD`

```

3099 \newdimen\memPD

```

`vminipage` The user view, just as for `minipage`, but ensures the `t` position and as a new paragraph. It is like `minipage` but with better fore and aft spacing.

```

3100 \newenvironment{vminipage}{%
3101   \par
3102   \@ifnextchar[%]
3103     \@ivminipage
3104     {\@iiiminipage t\relax[s]}
3105 }{%
3106   \par\global\memPD=\prevdepth
3107   \endminipage
3108   \par
3109   \kern-\memPD%      no pagebreak allowed here
3110   \hbox{\vrule depth \memPD width \z@}}
3111

```

\@ivminipage Deal with first optional argument to vminipage.

```

3112 \def\@ivminipage[#1]{%
3113   \@ifnextchar[%]
3114     {\@iiiminipage{t}}{\@iiiminipage{t}\relax[s]}

```

10.2 Abnormal parskips

Non-zero, positive \parskip, which is not to be encouraged. Code based on Robin Fairbairns parskip package and the NTG classes.

```

\ifm@mznzpskip \ifm@mznzpskip is TRUE if \parskip set to anything but 0pt.
\m@mznzpskiptrue 3115 \newif\ifm@mznzpskip
\m@mznzpskipfalse
\traditionalparskip \traditionalparskip sets \parskip to 0pt. \abnormalparskip{<length>} sets
\m@mabparskip \parskip to <length>. \nonzeroparskip sets \parskip to a non-zero value that
\abnormalparskip might be not too bad (any non-zero \parskip is not good).
\nonzeroparskip 3116 \newcommand*{\traditionalparskip}{%
3117   \setlength\parskip{0\p@ \@plus \p@}
3118   \m@mznzpskipfalse}
3119 \newskip\m@mabparskip
3120 \newcommand*{\abnormalparskip}[1]{%
3121   \setlength{\parskip}{#1}\m@mabparskip=#1\relax
3122   \m@mznzpskiptrue}
3123 \newcommand*{\nonzeroparskip}{\abnormalparskip{%
3124   0.5\baselineskip
3125   \@plus .1\baselineskip \@minus .1\baselineskip% NTG
3126   %% 0.5/baselineskip \@plus 2pt% RF
3127 }}
3128 \traditionalparskip
3129

```

\@minpagerestore This needs extending: here's what NTG does, but I've put the appropriate code later.

```

\providecommand*{\@minpagerestore}{%
  \parskip=.5\baselineskip \@plus .1\baselineskip \@minus .1\baselineskip}

```

11 Titles

For books the title is usually designed specifically for the particular work, so originally the class did not provide a `\maketitle` command or any of the `\title` and other commands that normally go along with this. After some thought I decided to add in the code from the titling package [Wil01g]. However, in this case life is a bit simpler as there is no pre-existing class code.

```

\prettitle  To provide some flexibility in the titling style of the document, user level
\@bsprettitle  commands are provided that can be changed to reconfigure the appearance
\posttitle  resulting from \maketitle.
\@bsposttitle 3130 \newcommand{\prettitle}[1]{\def\@bsprettitle{#1}}
\preauthor 3131 \newcommand{\posttitle}[1]{\def\@bsposttitle{#1}}
\@bspreauthor 3132 \newcommand{\preauthor}[1]{\def\@bspreauthor{#1}}
\postauthor 3133 \newcommand{\postauthor}[1]{\def\@bspostauthor{#1}}
\@bspostauthor 3134 \newcommand{\predate}[1]{\def\@bspredate{#1}}
\predate 3135 \newcommand{\postdate}[1]{\def\@bspostdate{#1}}
\@bspredate 3136
\postdate  These are defined initially to mimic the normal LATEX style.
\@bspostdate
3137 \prettitle{\begin{center}\LARGE}
3138 \posttitle{\par\end{center}\vskip 0.5em}
3139 \preauthor{\begin{center}
3140 \large \lineskip .5em%
3141 \begin{tabular}[t]{c}}
3142 \postauthor{\end{tabular}\par\end{center}}
3143 \predate{\begin{center}\large}
3144 \postdate{\par\end{center}}
3145

\maketitlehooka  The four hooks which will be called by \maketitle. These are initially vacuous.
\maketitlehookb 3146 \newcommand{\maketitlehooka}{}
\maketitlehookc 3147 \newcommand{\maketitlehookb}{}
\maketitlehookd 3148 \newcommand{\maketitlehookc}{}
3149 \newcommand{\maketitlehookd}{}
3150

\thanksmarkseries  These are for specifying the kind of series for thanks markers.
\@bsmarkseries 3151 \newcommand{\thanksmarkseries}[1]{%
\symbolthanksmark 3152 \def\@bsmarkseries{\renewcommand{\thefootnote}%
3153 {\@nameuse{#1}{footnote}}}}
3154 \newcommand{\symbolthanksmark}{\thanksmarkseries{\fnsymbol}}

\continuousmarks  These are for (non) zeroing of the footnote counter.
\@bscontmark 3155 \newcommand{\@bscontmark}{\setcounter{footnote}{0}}
3156 \newcommand{\continuousmarks}{\def\@bscontmark{}}

\thanksheadextra  These are for inserting stuff before and after a mark in the titling.
\@bsthanksheadpre
\@bsthanksheadpost

```

```

3157 \newcommand{\thanksheadextra}[2]{%
3158   \def\@bsthanksheadpre{#1}%
3159   \def\@bsthanksheadpost{#2}}

\thanksmark This adds a thanks mark. The \footnotemark could have been used directly but
             it is fragile in a moving argument.
3160 \DeclareRobustCommand{\thanksmark}[1]{\footnotemark[#1]}

\thanksgap This specifies some horizontal space.
3161 \newcommand{\thanksgap}[1]{\hspace{#1}}

\tamark This stores the current definition of \@thefnmark. For some reason using
        \@thefnmark directly only gave the last value.
3162 \newcommand{\tamark}{\@thefnmark}
3163

\thanksmarkwidth A length determining the size of the box for typesetting a thanks marker.
3164 \newlength{\thanksmarkwidth}

\thanksmarksep A length determining the inset of thanks footnotes.
3165 \newlength{\thanksmarksep}

\thanksmarkstyle \thanksscript is a wrapper round the actual mark stuff to be typeset. The user
\thanksscript can define this via \thanksmarkstyle{<code>}. The default is a superscript
mark.
3166 \newcommand{\thanksmarkstyle}[1]{\def\thanksscript#1{#1}}
3167 \thanksmarkstyle{\textsuperscript{#1}}

\makethanksmarkhook A vacuous macro used as a hook into \makethanksmark.
3168 \newcommand{\makethanksmarkhook}{}
3169

\thanksfootmark This typesets the thanks footnote mark.
3170 \newcommand{\thanksfootmark}{%
3171   \ifdim\thanksmarkwidth < \z@
      Negative width, mark is in the margin.
3172     \llap{\hb@xt@ -\thanksmarkwidth{%
3173       \hss\normalfont\thanksscript{\tamark}}}%
3174     \hspace*{-\thanksmarkwidth}%
3175   \else
3176     \ifdim\thanksmarkwidth = \z@
      Zero width, mark is at (inside) the margin.
3177       {\normalfont\thanksscript{\tamark}}%
3178     \else

```


Positive width.

```

3179      \hb@xt@\thanksmarkwidth{\hss\normalfont\thanksscript{\tmark}}}%
3180      \fi
3181    \fi}
3182

```

`\makethanksmark` This sets the general indentations for the thanks footnote, and typesets the mark. The code is a simplified version of that for typesetting ToC entries.

```

3183 \newcommand{\makethanksmark}{%
3184   \leavevmode%
3185   \parindent 1em\noindent
3186   %% \leftskip\thanksmarksep\relax
3187   \memRTLleftskip\thanksmarksep\relax
3188   %% \advance\leftskip\thanksmarkwidth
3189   \advance\memRTLleftskip\thanksmarkwidth
3190   %% \null\nobreak\hskip-\leftskip\relax
3191   \null\nobreak\hskip-\memRTLleftskip\relax
3192   \makethanksmarkhook\relax
3193   \thanksfootmark}
3194

```

`\usethanksrule` Simple macros that let `\footnoterule` to another rule definition.

```

\cancelthanksrule 3195 \newcommand{\usethanksrule}{\let\footnoterule\thanksrule}
3196 \newcommand{\cancelthanksrule}{\let\footnoterule\@bsfootnoterule}
3197

```

Now set up the rest of the thanks defaults, styling having been done earlier.

```

3198 \thanksmarkseries{fnsymbol} % symbols
3199 \thanksheadextra{}{}
3200 \setlength{\thanksmarkwidth}{1.8em}
3201 \setlength{\thanksmarksep}{-\thanksmarkwidth}
3202

```

`\thanksrule` These are saved versions of the `\footnoterule` definition as it is at the end of `\@bsfootnoterule` the preamble.

```

3203 \AtBeginDocument{%
3204   \let\thanksrule\footnoterule
3205   \let\@bsfootnoterule\footnoterule
3206 }
3207

```

`\droptitle` A titling block has `\droptitle` amount of additional vertical space above it (normally zero).

```

3208 \newlength{\droptitle}
3209 \setlength{\droptitle}{0pt}
3210

```

`\maketitle` The following is a modification of `\maketitle` as in the article, report, and book classes. It sets the pagestyle to *title*.

```

3211 \newcommand{\maketitle}{\par
3212   \begin{group}
3213     \@bsmarkseries
3214     \def\@makefnmark{\@textsuperscript{%
3215       \normalfont\@bsthanksheadpre \tamark \@bsthanksheadpost}}%
3216     \long\def\@makefntext##1{\makethanksmark ##1}
3217     \if@twocolumn
3218       \ifnum \col@number=\@one
3219         \@maketitle
3220       \else
3221         \twocolumn[\@maketitle]%
3222       \fi
3223     \else
3224       \ifdim\pagetotal>\z@
3225         \newpage
3226       \fi
3227       \global\@topnum\z@
3228       \@maketitle
3229     \fi
3230     \thispagestyle{title}\@thanks
3231   \end{group}
3232   \@bscontmark % \setcounter{footnote}{0}%
3233 }
3234 \aliaspagestyle{title}{plain}
3235
\@mem@titlefootkill \@mem@titlefootkill{\note} Warn about footnotes in titles.
3236 \newcommand*{\@mem@titlefootkill}[1]{%
3237   \@memwarn{Do not use \string\footnote\space in
3238     \string\maketitle.\MessageBreak
3239     Use \protect\thanks\space instead}}
3240
\maketitle Our version of \@maketitle. Footnotes are killed in the title; see the thread
  ‘\title, \author and \footnote feature in memoir class’, February 2003.
3241 \newcommand{\@maketitle}{%
3242   \let\footnote\@mem@titlefootkill
3243   \ifdim\pagetotal>\z@
3244     \newpage
3245   \fi
3246   \null
3247   \vskip 2em%
3248   \vspace*{\droptitle}
3249   \maketitlehooka
3250   {\@bsprettitle \@title \@bsposttitle}
3251   \maketitlehookb
3252   {\@bspreauthor \@author \@bspostauthor}
3253   \maketitlehookc
3254   {\@bspredate \@date \@bspostdate}
3255   \maketitlehookd

```

```

3256 \par
3257 \vskip 1.5em}
3258

```

titlingpage The `titlingpage` environment sets the `pagestyle` to be *titlingpage*, disables the footnote rule and ensures that the page is single column. At the end it switches back to `twocolumn` if necessary, and then starts a new page as number 1.

```

3259 \newenvironment{titlingpage}%
3260 {\let\footnoterule\relax
3261 \let\footnotesize\small
3262 \if@twocolumn
3263 \if@restonecoltrue\onecolumn
3264 \else
3265 \if@restonecolfalse
3266 \fi
3267 \thispagestyle{titlingpage}%
3268 \setcounter{page}{\@ne}%
3269 }{%
3270 \thispagestyle{titlingpage}%
3271 \if@restonecol \twocolumn \fi
3272 \if@twoside \cleardoublepage \else \clearpage \fi
3273 \setcounter{page}{\@ne}}
3274 \aliaspagestyle{titlingpage}{empty}
3275

```

\emptythanks This macro discards all prior `\thanks` texts.

```

3276 \newcommand{\emptythanks}{\global\let\@thanks\@empty}
3277

```

\andnext The kernel `\and` macro puts space between author's names. The `\andnext` macro puts a newline between the names.

```

3278 \newcommand*{\andnext}{%
3279 \end{tabular}}\ \begin{tabular}[t]{c}}
3280

```

\@bsmtitleempty `\@bsmtitleempty` is a helper macro to save some macro space. It empties some elements of `\maketitle`.

```

3281 \newcommand{\@bsmtitleempty}{%
3282 \global\let\maketitle\relax
3283 \global\let\@maketitle\relax
3284 \global\let\title\relax
3285 \global\let\author\relax
3286 \global\let\date\relax
3287 \global\let\thanksmarkseries\relax
3288 \global\let\thanksheadextra\relax
3289 \global\let\thanksfootextra\relax
3290 \global\let\thanksmark\relax
3291 \global\let\thanksgap\relax}
3292

```

`\keepthetitle` This macro undefines all the titling commands except for `\thetitle`, `\theauthor` and `\thedata`.

```

3293 \newcommand{\keepthetitle}{%
3294   \@bsmtitleempty
3295   \global\let\thanks\relax
3296   \global\let\and\relax
3297   \global\let\andnext\relax
3298   \global\let\@thanks\@empty
3299   \global\let\@title\@empty
3300   \global\let\@author\@empty
3301   \global\let\@date\@empty}
3302

```

`\killtitle` `\killtitle` undefines the remaining macros of `\maketitle`.

```

3303 \newcommand{\killtitle}{%
3304   \keepthetitle
3305   \global\let\thetitle\relax
3306   \global\let\theauthor\relax
3307   \global\let\thedata\relax}
3308

```

`\thetitle` In order to make the `\title`, etc., values available for printing their definitions
`\theauthor` (in `ltsect.dtx`) need extending to save their arguments. We have to make sure
`\thedata` that extraneous material, like `\thanks`, is excluded from the saved texts.

```

3309 \addtoargdef{\title}{%
3310   \begingroup\let\footnote\@gobble}{%
3311   \begingroup
3312     \renewcommand{\thanks}[1]{}
3313     \renewcommand{\thanksmark}[1]{}
3314     \renewcommand{\thanksgap}[1]{}
3315     \protected@xdef\thetitle{#1}
3316   \endgroup\endgroup}
3317 \addtoargdef{\author}{%
3318   \begingroup\let\footnote\@gobble}{%
3319   \begingroup
3320     \renewcommand{\thanks}[1]{}
3321     \renewcommand{\and}{\unskip, }
3322     \renewcommand{\andnext}{\unskip, }
3323     \renewcommand{\thanksmark}[1]{}
3324     \renewcommand{\thanksgap}[1]{}
3325     \protected@xdef\theauthor{#1}
3326   \endgroup\endgroup}
3327 \addtoargdef{\date}{%
3328   \begingroup\let\footnote\@gobble}{%
3329   \begingroup
3330     \renewcommand{\thanks}[1]{}
3331     \renewcommand{\thanksmark}[1]{}
3332     \renewcommand{\thanksgap}[1]{}
3333     \protected@xdef\thedata{#1}

```

```
3334 \endgroup\endgroup}
3335
```

12 Parts, chapters and other divisions

12.1 Building blocks

The definitions in this part of a class file usually make use of two internal macros, `\@startsection` and `\secdef`. To understand what is going on here, we describe their syntax.

`\@startsection` The macro `\@startsection` has 6 required arguments, optionally followed by a *, an optional argument and a required argument:
`\@startsection<name><level><indent><before skip><after skip><style>` optional *

`[<altheading>]<heading>`

It is a generic command to start a section, the arguments have the following meaning:

`<name>` The name of the user level command, e.g., ‘section’.

`<level>` A number, denoting the depth of the section — e.g., chapter = 0, section = 1, etc. A section number will be printed if and only if `<level>` ≤ the value of the `secnumdepth` counter.

`<indent>` The indentation of the heading from the left margin

`<before skip>` The absolute value of this argument gives the skip to leave above the heading. If it is negative, then the paragraph indent of the text following the heading is suppressed.

`<after skip>` If positive, this gives the skip to leave below the heading, else it gives the skip to leave to the right of a run-in heading.

`<style>` Commands to set the style of the heading.

* When this is missing the heading is numbered and the corresponding counter is incremented.

`<altheading>` Gives an alternative heading to use in the table of contents and in the running heads. This should be present when the * form is used.

`<heading>` The heading of the new section.

A sectioning command is normally defined to `\@startsection` and its first six arguments.

`\secdef` The macro `\secdef` can be used when a sectioning command is defined without using `\@startsection`. It has two arguments:
`\secdef<unstarcmds><starcmds>`

`<unstarcmds>` Used for the normal form of a sectioning command.

`<starcmds>` Used for the *-form of a sectioning command.

You can use `\secdef` as follows:

```
\def\chapter { ... \secdef \CMDA \CMDB }
\def\CMDA    [#1]#2{ ... } % Command to define
                        % \chapter[...]{...}
\def\CMDB    #1{ ... }    % Command to define
                        % \chapter*{...}
```

`\@hangfrom` Internally the `\@startsection` macro uses `\@hangfrom{NUM}`, where NUM is the sectional number, to produce a hanging paragraph. That is, the second and later lines of a multiline title are indented from the left margin by the width of the number. The definition of `\@hangfrom`, from `ltsect.dtx`, is:

```
\def\@hangfrom#1{\setbox\@tempboxa\hbox{#{#1}}%
  \hangindent \wd\@tempboxa\noindent\box\@tempboxa}
```

To get a normal paragraphed title you can do:

```
\renewcommand{\@hangfrom}[1]{#1}
```

or as a block paragraph:

```
\renewcommand{\@hangfrom}[1]{\noindent #1}
```

`\@secCNTformat` The `\@startsection` macro also uses `\@secCNTformat{NUM}` to format the section number, including the space after it. Its definition, from `ltsect.dtx`, is

```
\def\@secCNTformat#1{\csname the#1\endcsname\quad}
```

This is the command to change if you need different number formatting. For example the combination

```
\renewcommand{\@secCNTformat}[1]{\llap{\csname the#1\endcsname\quad}}
\renewcommand{\@hangfrom}[1]{\noindent #1}
```

will hang the section numbers in the margin.

The values used for the document division levels are the same as defined by the standard L^AT_EX classes, with the exception of the book division which is a memoir addition, and are given in Table 4.

12.2 Mark commands

`\bookpagemark` The default initialisations of the `\...mark` commands for use in the pagestyles.

`\partmark` Most are already defined in the kernel but they are all noted here.

```
\chaptermark 3336 \newcommand*{\bookpagemark}[1]{}
```

```
\sectionmark 3337 \newcommand*{\partmark}[1]{}
```

```
\subsectionmark 3338 \newcommand*{\chaptermark}[1]{}
```

```
\subsubsectionmark
```

```
\paragraphmark
```

```
\subparagraphmark
```

Table 4: Document division levels

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

```

3339 % \newcommand*{\sectionmark}[1]{
3340 % \newcommand*{\subsectionmark}[1]{
3341 % \newcommand*{\subsubsectionmark}[1]{
3342 % \newcommand*{\paragraphmark}[1]{
3343 % \newcommand*{\subparagraphmark}[1]{
3344

```

`\bibmark` Marks for the bibliography, which may be filled with `\bibname`.

```
3345 \newcommand*{\bibmark}{}

```

`\indexmark` Marks for the index, which may be filled with `\indexname`.

```
3346 \newcommand*{\indexmark}{}

```

`\glossarymark` Marks for the glossary, which may be filled with `\glossaryname`.

```
3347 \newcommand*{\glossarymark}{}

```

```
3348

```

12.3 Define Counters

`\c@secnumdepth` The value of the counter *secnumdepth* gives the depth of the highest-level sectioning command that is to produce section numbers.

```
3349 \setcounter{secnumdepth}{2}

```

`\c@book` These counters are used for the sectioning numbers. The macro

```
\c@part \newcounter{<newctr>}[<oldctr>]
```

`\c@chapter` defines *<newctr>* to be a counter, which is reset to zero when counter *<oldctr>* is stepped. Counter *<oldctr>* must already be defined.

`\c@section` Book, part and chapter are the top level document divisions.

`\c@subsection` Book, part and chapter are the top level document divisions.

```
\c@subsubsection 3350 \newcounter{book} \setcounter{book}{0}

```

```
\c@paragraph 3351 \newcounter{part} \setcounter{part}{0}

```

```
\c@subparagraph 3352 \newcounter{chapter} \setcounter{chapter}{0}

```

The lower level divisions get reset by higher level divisions.

```
3353 \newcounter{section}[chapter]

```

```
3354 \newcounter{subsection}[section]

```

```

3355 \newcounter{subsubsection}[subsection]
3356 \newcounter{paragraph}[subsubsection]
3357 \newcounter{subparagraph}[paragraph]

\thebook For any counter CTR, \theCTR is a macro that defines the printed version of
\thepart counter CTR. It is defined in terms of the following macros:
\thechapter \arabic{<COUNTER>} prints the value of <COUNTER> as an arabic numeral.
\thesection \roman{<COUNTER>} prints the value of <COUNTER> as a lowercase roman
\thesubsection numeral.
\thesubsubsection \Roman{<COUNTER>} prints the value of <COUNTER> as an uppercase roman
\theparagraph numeral.
\thesubparagraph \alph{<COUNTER>} prints the value of <COUNTER> as a lowercase letter:
1 = a, 2 = b, etc.
\Alph{<COUNTER>} prints the value of <COUNTER> as an uppercase letter:
1 = A, 2 = B, etc.

3358 \renewcommand*{\thebook}{\@Roman\c@book}
3359 \renewcommand*{\thepart}{\@Roman\c@part}
3360 \renewcommand*{\thechapter}{\@arabic\c@chapter}
3361 \renewcommand*{\thesection}{\thechapter.\@arabic\c@section}
3362 \renewcommand*{\thesubsection}{%
3363     \thesection.\@arabic\c@subsection}
3364 \renewcommand*{\thesubsubsection}{%
3365     \thesubsection.\@arabic\c@subsubsection}
3366 \renewcommand*{\theparagraph}{%
3367     \thesubsubsection.\@arabic\c@paragraph}
3368 \renewcommand*{\thesubparagraph}{%
3369     \theparagraph.\@arabic\c@subparagraph}

\@chapapp \@chapapp is initially defined as \chaptername. The \appendix command
redfines it as \appendixname.

3370 \newcommand{\@chapapp}{\chaptername}
3371

```

12.4 Front, main and back matter

These are the three main logical divisions in a book. As noted earlier, the boolean `\if@mainmatter` is TRUE iff the main matter is being processed. Chapters will be unnumbered when `\if@mainmatter` is FALSE.

```

\frontmatter The \frontmatter command starts roman numbering and turns off chapter
\frontmatter* numbering. It ensures that lower level divisions will not have chapter numbers,
nor will figures or tables. It also ensures that the next page will be recto. The
starred version makes no changes to the page numbering14.

3372 \newcommand{\frontmatter}{%
3373     \@ifstar{\@smemfront}{\@memfront}}

```

¹⁴The starred versions were added to meet a request by Daniel Richard G. (skunk@mit.edu) in September 2001.

`\@memfront@floats` acting out the float counter domination, in case someone wants something added or changed

```
3374 \newcommand\@memfront@floats{%
3375   \counterwithout{figure}{chapter}
3376   \counterwithout{table}{chapter}
3377 }
```

`\@smemfront`

```
3378 \newcommand{\@smemfront}{%
3379   \cleardoublepage
3380   \@mainmatterfalse
3381   \setcounter{secnumdepth}{-10}
3382   \@memfront@floats
3383 }
```

`\@memfront`

```
3384 \newcommand{\@memfront}{%
3385   \@smemfront\pagenumbering{roman}}
3386
```

`\mainmatter` The `\mainmatter` command starts arabic numbering and turns on chapter numbering. It sets numbering to the normal state. It also ensures that the next page will be recto. The starred version does not change the page numbering. Romano Giannetti suggested that `\mainmatter` (and `\backmatter`) should be almost no-ops with the article option.

```
3387 \newcommand{\mainmatter}{%
3388   \@ifstar{\@smemmain}{\@memmain}}
```

`\@memmain@floats` acting out the float counter domination, in case someone wants something added or changed

```
3389 \newcommand\@memmain@floats{%
3390   \counterwithin{figure}{chapter}
3391   \counterwithin{table}{chapter}
3392 }
```

`\@smemmain` If `\frontmatter` and `\mainmatter` are used with the article option, then a page numbering / margin problem can occur if the frontmatter ends on an odd page — the first mainmatter next page is odd as well which throws puts two odd-margined pages together.

One fix from Morten is to always `\cleardoublepage` for twosided articles.

One fix from me is for the ...matter macros to just switch on/off section numbering.

Morten's is simpler to implement.

```
3393 \newcommand*{\@smemmain}{%
3394   \@mainmattertrue
3395   \setcounter{secnumdepth}{\value{maxsecnumdepth}}
3396   \ifartopt
```

```

3397     \if@twoside
3398         \cleardoublepage
3399     \else
3400         \clearpage
3401     \fi
3402 \else
3403     \cleardoublepage
3404     \@memmain@floats
3405 \fi}

\@memmain
3406 \newcommand{\@memmain}{%
3407     \@smemmain\pagenumbering{arabic}}
3408

\@memback@floats  acting out the float counter domination, in case someone wants something
                   added or changed
3409 \newcommand{\@memback@floats}{%
3410     \counterwithout{figure}{chapter}
3411     \counterwithout{table}{chapter}
3412     \setcounter{figure}{0}
3413     \setcounter{table}{0}
3414 }

\backmatter  This command turns off chapter numbering but leaves the page numbering
              alone. It twiddles the numbering caption numbering. The back matter may start
              on any new page.
3415 \newcommand{\backmatter}{%
3416     \ifartopt
3417         \clearpage
3418     \else
3419         \if@openright
3420             \cleardoublepage
3421         \else
3422             \clearpage
3423         \fi
3424     \fi
3425     \@mainmatterfalse
3426     \setcounter{secnumdepth}{-10}
3427     \ifartopt\else
3428         \@memback@floats
3429     \fi}
3430

```

12.5 Book

This was added at the request of Frederic Connes who said that in Frency typography there was often a division above the `\part` level.

`\theHbook` These are needed if the `hyperref` is used.

```

\toclevel@book 3431 \newcommand*{\theHbook}{\arabic{book}}
                 3432 \newcommand*{\toclevel@book}{-2}
                 3433

```

`\book` `\book{<title>}` starts a new book division called *<title>*. The actual typesetting of `\book*` the title is done by `\@book` or `\@sbook`.

```

                 3434 \newcommand*{\book}{%
                 3435   \@setupbook
                 3436   \secdef\@book\@sbook}

```

`\beforebookskip` These three macros are the skips before, in the middle, and after the Book heading.

```

\midbookskip
\afterbookskip 3437 \newcommand*{\beforebookskip}{\null\vfil}
                 3438 \newcommand*{\midbookskip}{\par\vskip 2\onelineskip}
                 3439 \newcommand*{\afterbookskip}{\vfil\newpage}
                 3440

```

`\@setupbook` This macro does the work of setting up for the `\book` command. A single column page, normally recto, with a *book* pagestyle is started.

```

                 3441 \newcommand{\@setupbook}{%
                 3442   \if@openright
                 3443     \cleardoublepage
                 3444   \else
                 3445     \clearpage
                 3446   \fi
                 3447   \thispagestyle{book}%
                 3448   \if@twocolumn
                 3449     \onecolumn
                 3450   \@tempwatrue
                 3451   \else
                 3452     \@tempswafalse
                 3453   \fi
                 3454   \beforebookskip}
                 3455

```

`\booknamefont` These three macros specify the fonts for the book name and number, and for the `\booknumfont` book title, respectively.

```

\booktitlefont 3456 \newcommand*{\booknamefont}{\normalfont\huge\bfseries}
                 3457 \newcommand*{\booknumfont}{\normalfont\huge\bfseries}
                 3458 \newcommand*{\booktitlefont}{\normalfont\Huge\bfseries}
                 3459

```

`\printbookname` Macros to print the various books of a Book heading.

```

\booknamenenum 3460 \newcommand*{\printbookname}{\booknamefont \bookname}
\printbooknum  3461 \newcommand*{\booknamenenum}{\space}
\printbooktitle 3462 \newcommand*{\printbooknum}{\booknumfont \thebook}
                 3463 \newcommand*{\printbooktitle}[1]{\booktitlefont #1}
                 3464

```

```

\membookinfo
\membookstarinfo 3465 \newcommand{\membookinfo}[3]{}
                  3466 \newcommand{\membookstarinfo}[1]{}
                  3467

\@book \book[short]{long} typesets the title of a \book book. There is a number
      if \secnumdepth is greater than -3.
3468 \long\def\@book#1#2{%
3469   \M@gettitle{#1}%
3470   \phantomsection
3471   \ifnum \c@secnumdepth >-3\relax
3472     \refstepcounter{book}%
3473     \addcontentsline{toc}{book}%
3474       {\protect\booknumberline{\thebook}#1}%
3475     \membookinfo{\thebook}{#1}{#2}%
3476   \else
3477     \addcontentsline{toc}{book}{#1}%
3478     \membookinfo{}{#1}{#2}%
3479   \fi

      Empty the marks, center the title on the page, and set the normal font.
3480   \bookpagemark{#1}%
3481   {\centering
3482     \interlinepenalty \@M
3483     \normalfont

      Print the number, if there is one, then the title below. Macro \@endbook tidies
      everything up at the end.
3484     \ifnum \c@secnumdepth >-3\relax
3485       \printbookname \booknamenum \printbooknum
3486       \midbookskip
3487     \fi
3488     \printbooktitle{#2}\par}%
3489   \@endbook}
3490

\@sbook \@sbook{long} formats the title of a \book* book. It is simpler than \@book
      because there is no number to print.
3491 \long\def\@sbook#1{%
3492   \M@gettitle{#1}%
3493   \phantomsection
3494   \membookstarinfo{#1}%
3495   {\centering
3496     \interlinepenalty \@M
3497     \normalfont
3498     \printbooktitle{#1}\par}%
3499   \@endbook}
3500

```

```

\ifm@mbooknewpage The declaration \nobookblankpage prevents \book from outputting a further
\m@mbooknewpagetrue (blank) page when it finishes. The default is set by \bookblankpage.
\m@mbooknewpagefalse 3501 \newif\ifm@mbooknewpage
\bookblankpage 3502 \m@mbooknewpagefalse
\nobookblankpage 3503 \newcommand*{\bookblankpage}{\m@mbooknewpagefalse}
3504 \newcommand*{\nobookblankpage}{\m@mbooknewpagetrue}
3505

\bookpageend This finishes off both \@book and \@sbook. Following a \nobookblankpage
\@endbook nothing is done, otherwise (after \bookblankpage) the current page is flushed,
and an extra blank page is created if both twoside and openright are in effect. At
the end, if necessary, two column mode is switched back on.
3506 \newcommand*{\bookpageend}{\afterbookskip
3507 \ifm@mbooknewpage
3508 \else
3509 \if@twoside
3510 \if@openright
3511 \null
3512 \thispagestyle{afterbook}%
3513 \newpage
3514 \fi
3515 \fi
3516 \fi
3517 \if@tempwa
3518 \twocolumn
3519 \fi}
3520 \def\@endbook{\bookpageend}
3521

\ps@book Pagestyles for the book page and the (blank) page after a book page.
\ps@afterbook 3522 \aliaspagestyle{book}{empty}
3523 \aliaspagestyle{afterbook}{empty}
3524

```

12.6 Part

```

\part \part{<title>} starts a new Part called <title>. The actual typesetting of the title
is done by \@part or \@spart.
3525 \newcommand{\part}{%
3526 \@setuppart
3527 \secdef\@part\@spart}

\beforepartskip These three macros are the skips before, in the middle, and after the Part
\midpartskip heading.
\afterpartskip 3528 \newcommand{\beforepartskip}{\null\vfil}
3529 \newcommand{\midpartskip}{\par\vskip 2\onelineskip}
3530 \newcommand{\afterpartskip}{\vfil\newpage}
3531

```

`\setuppart` This macro does the work of setting up for the `\part` command. A single column page, normally recto, with a *part* pagestyle is started.

```

3532 \newcommand{\setuppart}{%
3533   \if@openright
3534     \cleardoublepage
3535   \else
3536     \clearpage
3537   \fi
3538   \thispagestyle{part}%
3539   \if@twocolumn
3540     \onecolumn
3541     \@tempwattrue
3542   \else
3543     \@tempwafalse
3544   \fi
3545   \beforepartskip}
3546
```

`\partnamefont` These three macros specify the fonts for the part name and number, and for the
`\partnumfont` part title, respectively.

```

\parttitlefont 3547 \newcommand{\partnamefont}{\normalfont\huge\bfseries}
3548 \newcommand{\partnumfont}{\normalfont\huge\bfseries}
3549 \newcommand{\parttitlefont}{\normalfont\Huge\bfseries}
3550
```

`\printpartname` Macros to print the various parts of a Part heading.

```

\partnamenumb 3551 \newcommand{\printpartname}{\partnamefont \partname}
\printpartnum 3552 \newcommand{\partnamenumb}{\space}
\printparttitle 3553 \newcommand{\printpartnum}{\partnumfont \thepart}
3554 \newcommand{\printparttitle}[1]{\parttitlefont #1}
3555
```

`\mempartinfo` `\mempartinfo{\thepart}{short}{long}`
`\mempartstarinfo` `\mempartstarinfo{long}`

```

3556 \newcommand{\mempartinfo}[3]{}
3557 \newcommand{\mempartstarinfo}[1]{}
3558
```

`\@part` `\@part[short][long]` typesets the title of a `\part` part. There is a number if
`\secnumdepth` is greater than -2.

```

3559 \long\def\@part[#1]#2{%
3560   \M@gettitle{#1}%
3561   \phantomsection
3562   \ifnum \c@secnumdepth >-2\relax
3563     \refstepcounter{part}%
3564     \addcontentsline{toc}{part}%
3565       {\protect\partnumberline{\thepart}#1}%
3566     \mempartinfo{\thepart}{#1}{#2}%

```

```

3567 \else
3568   \addcontentsline{toc}{part}{#1}%
3569   \mempartinfo{#1}{#2}%
3570 \fi

```

Empty the marks, center the title on the page, and set the normal font.

```

3571 \partmark{#1}%
3572 {\centering
3573  \interlinepenalty \@M
3574  \normalfont

```

Print the number, if there is one, then the title below. Macro `\@endpart` tidies everything up at the end.

```

3575   \ifnum \@secnumdepth >-2\relax
3576     \printpartname \partnamenum \printpartnum
3577     \midpartskip
3578   \fi
3579   \printparttitle{#2}\par}%
3580 \@endpart}
3581

```

`\@spart` `\@spart{<long>}` formats the title of a `\part*` part. It is simpler than `\@part` because there is no number to print.

```

3582 \long\def\@spart#1{%
3583   \M@gettitle{#1}%
3584   \phantomsection
3585   \mempartstarinfo{#1}%
3586   {\centering
3587    \interlinepenalty \@M
3588    \normalfont
3589    \printparttitle{#1}\par}%
3590   \@endpart}
3591

```

`\ifm@mnopartnewpage` The declaration `\nopartblankpage` prevents `\part` from outputting a blank page when it finishes. The default is set by `\partblankpage`. This code was originally supplied by Frederic Connes.

```

\partblankpage 3592 \newif\ifm@mnopartnewpage
\nopartblankpage 3593 \m@mnopartnewpagefalse
3594 \newcommand*{\partblankpage}{\m@mnopartnewpagefalse}
3595 \newcommand*{\nopartblankpage}{\m@mnopartnewpagetrue}
3596

```

`\partpageend` This finishes off both `\@part` and `\@spart`. Following a `\nopartblankpage` nothing is done, otherwise (after `\partblankpage`) the current page is flushed. In the standard book class if two-sided mode is on a blank page is then produced. I think that this looks odd when the `openany` option is in force, so here it only produces an extra blank page if both `twoside` and `openright` are in effect. At the end, if necessary, two column mode is switched back on.

```

3597 \newcommand*{\partpageend}{\afterpartskip
3598   \ifm@mnopartnewpage
3599   \else
3600     \if@twoside
3601       \if@openright
3602         \null
3603         \thispagestyle{afterpart}%
3604         \newpage
3605       \fi
3606     \fi
3607   \fi
3608   \if@tempwa
3609     \twoocolumn
3610   \fi}
3611 \def\endpart{\partpageend}
3612

```

`\ps@afterpart` Pagestyle for the (blank) page after a part page.

```

3613 \aliaspagestyle{afterpart}{empty}
3614

```

12.7 Chapter

`\chapter` The command to start a new chapter. Chapters always start on a new page with a *chapter* pagestyle. Floats are not allowed at the top of the page. The typesetting is done by either `\@chapter` or `\@schapter`.

```

3615 \newcommand\chapter{%
3616   \ifartopt\par\else
3617     \clearforchapter
3618     \thispagestyle{chapter}
3619     \global\@topnum\z@
3620   \fi
3621   \@afterindentfalse
3622   \@ifstar{\@m@mschapter}{\@m@mchapter}}
3623

```

`\@m@mchapter` Intermediate and support macros for the extra optional argument to `\chapter`.

`\ch@pt@c` Have to do this long windedly otherwise dear old `hyperref` barfs.

`\m@m@empty` The code for two optional arguments is based on a posting to CTT by Robin Fairbairns (1997/04/12 *Re: Several optional arguments for macro?*).

```

3624 \newcommand{\@m@mchapter}[1][{}]{%
3625   \def\ch@pt@c{#1}% capture first optional arg
3626   \@ifnextchar[{\@chapter}{\@chapter[]}%
3627 }
3628 \def\m@m@empty{\@empty}
3629

```

`\memchapinfo` `\memchapinfo{num}{toc}{head}{full}`

`\memchapstarinfo`

`\memappchapinfo`

`\memappchapstarinfo`


```

\memchapstarinfo{short}{full}
3630 \newcommand{\memchapinfo}[4]{}
3631 \newcommand{\memchapstarinfo}[2]{}
3632 \newcommand{\memappchapinfo}[4]{}
3633 \newcommand{\memappchapstarinfo}[2]{}
3634

```

```

\ifm@pn@new@chap These are for supporting the [age/end notes, flagging that a \chapter(*) has
\m@pn@new@chapfalse been called.
\ifm@pn@new@chaptrue 3635 \newif\ifm@pn@new@chap
\ifm@pn@new@schap 3636 \m@pn@new@chapfalse
\m@pn@new@schapfalse 3637 \newif\ifm@pn@new@schap
\m@pn@new@schaptrue 3638 \m@pn@new@schapfalse
3639

```

`\@chapter` `\@chapter[<tocmark>]{<title>}` typesets the title of a `\chapter`. There is a `\f@rtoc` number if `\secnumdepth` is greater than -1 and `\@mainmatter` is TRUE.

```

\f@rhdr 3640 \def\@chapter[#1]#2{%
\f@rbody 3641 \m@pn@new@chaptrue%
3642 \m@pn@new@schapfalse%
3643 \def\f@rbody{#2}%
3644 \ifx\ch@pt@c@empty % no optional args
3645 \def\f@rtoc{#2}%
3646 \def\f@rhdr{#2}%
3647 \else % at least one opt arg
3648 \let\f@rtoc\ch@pt@c
3649 \ifx@empty#1@empty
3650 \let\f@rhdr\ch@pt@c
3651 \else
3652 \def\f@rhdr{#1}%
3653 \fi
3654 \fi
3655 \m@Andfalse
3656 \ifnum \c@secnumdepth >\m@ne
3657 \if@mainmatter
3658 \m@Andtrue
3659 \fi
3660 \fi
3661 \ifm@And
3662 \refstepcounter{chapter}%
3663 \fi

```

Deal with the article option.

```

3664 \ifartopt
3665 \@makechapterhead{#2}%
3666 \@afterheading
3667 \chaptermark{\f@rhdr}%
3668 \else

```

Store the (short) title via `\chaptermark`, and add some whitespace to the LoF and LoT. Then fiddle when we are using two columns, calling `\@makechapterhead` to do the typesetting.

```

3669 \chaptermark{\f@rhdr}
3670 \insertchapterspace
3671 \if@twocolumn
3672 \topnewpage[\@makechapterhead{#2}]%
3673 \else
3674 \makechapterhead{#2}%
3675 \fi
3676 \@afterheading
3677 \fi

```

Vittorio De Martino (demartino.vittoria@grtn.it) on 31 March 2003 reported that for article chapters, which do not do any `\clearpage`, `\addcontents` had to come after the title.

```

3678 \ifm@m@And
3679 \ifanappendix
3680 \addcontentsline{toc}{appendix}{%
3681 \protect\chapternumberline{\thechapter}\f@rtoc}%
3682 \memappchapinfo{\thechapter}{\f@rtoc}{\f@rhdr}{#2}%
3683 \else
3684 \addcontentsline{toc}{chapter}{%
3685 \protect\chapternumberline{\thechapter}\f@rtoc}%
3686 \memchapinfo{\thechapter}{\f@rtoc}{\f@rhdr}{#2}%
3687 \fi
3688 \else
3689 \addcontentsline{toc}{chapter}{\f@rtoc}%
3690 \ifanappendix
3691 \memappchapinfo{}{\f@rtoc}{\f@rhdr}{#2}%
3692 \else
3693 \memchapinfo{}{\f@rtoc}{\f@rhdr}{#2}%
3694 \fi
3695 \fi

```

Add hook for title referencing.

```

3696 \ifheadnameref\M@getttitle{\f@rhdr}\else\M@getttitle{\f@rtoc}\fi
3697

```

`\@makechapterhead` This *really* typesets a `\chapter`. Leave some whitespace and prepare to set `\raggedright`.

```

3698 \def\@makechapterhead#1{%
3699 \chapterheadstart% \vspace*{50\p@}%
3700 {%\parindent \z@ \raggedright \normalfont
3701 \parindent \z@ \memRTLraggedright \normalfont

```

If there is a number, typeset it, otherwise call `\printchapternonum`.

```

3702 \ifm@m@And
3703 \printchaptername \chapternamenum \printchapternum
3704 \afterchapternum % \par\nobreak \vskip 20\p@

```

```

3705 \else
3706     \printchapternonum
3707 \fi

Typeset the title.

3708 \interlinepenalty\@M
3709 \printchaptertertitle{#1} % \Huge \bfseries #1
3710 \afterchaptertertitle % \par\nobreak \vskip 40\p@
3711 }
3712

```

`\insertchapterspace` By default, a `\chapter` inserts some vertical space into the LoF and LoT. The macro `\insertchapterspace` performs the insertion.

```

3713 \newcommand*\insertchapterspace{%
3714 \addtocontents{lof}{\protect\addvspace{10pt}}%
3715 \addtocontents{lot}{\protect\addvspace{10pt}}%
3716

```

`\beforechapskip` Lengths separating the various parts of a chapter heading.

```

\midchapskip 3717 \newlength{\beforechapskip}\setlength{\beforechapskip}{50pt}
\afterchapskip 3718 \newlength{\midchapskip}\setlength{\midchapskip}{20pt}
3719 \newlength{\afterchapskip}\setlength{\afterchapskip}{40pt}
3720

```

`\@chs@def@ult` This sets up all the definitions and defaults used in `\@makechapterhead` and `\@makeschapterhead`.

```

\chapterheadstart
\printchaptername 3721 \newcommand{\@chs@def@ult}{%
    \chapternamenumber 3722 \def\chapterheadstart{\vspace*{\beforechapskip}}%
    \printchapternum 3723 \def\printchaptername{\chapnamefont \@chapapp}%
    \afterchapternum 3724 \def\chapternamenumber{\space}%
    \printchapternum 3725 \def\printchapternum{\chapnumfont \thechapter}%
\printchapternonum 3726 \def\afterchapternum{\par\nobreak\vskip \midchapskip}%
\printchaptertertitle 3727 \def\printchapternonum{}%
\afterchaptertertitle 3728 \def\printchaptertertitle##1{\chapttitlefont ##1}%
3729 \def\afterchaptertertitle{\par\nobreak\vskip \afterchapskip}%

```

`\chapnamefont` Fonts for setting chapter name, number, and title.

```

\chapnumfont 3730 \def\chapnamefont{\normalfont\huge\bfseries}%
\chapttitlefont 3731 \def\chapnumfont{\normalfont\huge\bfseries}%
3732 \def\chapttitlefont{\normalfont\Huge\bfseries}%
3733 \setlength{\beforechapskip}{50pt}%
3734 \setlength{\midchapskip}{20pt}%
3735 \setlength{\afterchapskip}{40pt}%
3736

```

`\@m@mschapter` This deals with the new optional argument for starred chapters.

```

3737 \newcommand{\@m@mschapter}[2][\@empty]{%
3738 \schapter{#2}%

```

```

3739 \ifx \@empty#1
3740   \def\frhdr{#2}%
3741 \else%                               opt arg
3742   \def\frhdr{#1}%
3743   \setcounter{secnumdepth}{-10}%
3744   \chaptermark{#1}%
3745   \setcounter{secnumdepth}{\value{maxsecnumdepth}}%
3746 \fi
3747 \ifanappendix
3748   \memappchapstarinfo{\frhdr}{#2}%
3749 \else
3750   \memchapstarinfo{\frhdr}{#2}%
3751 \fi
3752 \ifheadnameref\M@getttitle{\frhdr}\else\M@getttitle{#2}\fi}
3753

```

`\schapter` `\schapter{<long>}` typesets the title of a `\chapter*`. It is easier than the `\frbdy` `\chapter` as there is no number to worry about.

```

3754 \newcommand{\schapter}[1]{%
3755   \m@mn@new@schaptrue%
3756   \m@mn@new@chapfalse%
3757   \def\frbdy{#1}%
3758   \ifartopt
3759     \@makeschapterhead{#1}%
3760   \else
3761     \if@twocolumn
3762       \@topnewpage[\@makeschapterhead{#1}]%
3763     \else
3764       \@makeschapterhead{#1}%
3765     \fi
3766   \fi
3767   \@afterheading}
3768

```

`\@makeschapterhead` This *really* typesets a `\chapter*`, and is similar to `\@makechapterhead`.

```

3769 \def\@makeschapterhead#1{%
3770   \chapterheadstart
3771   {%\parindent \z@ \raggedright \normalfont
3772     \parindent \z@ \memRTLraggedright \normalfont
3773     \printchapternonum
3774     \interlinepenalty\M
3775     \printchaptertitle{#1}%
3776     \afterchaptertitle}%
3777 }
3778

```

12.7.1 Chapter styling

`\makechapterstyle` `\chapterstyle{<style>}` is like `\pagestyle`, except it's for chapters.
`\chapterstyle`

`\makechapterstyle{<style>}{<text>}` creates or overrides the *<style>* chapter style defining it as *<text>*. So, `\makechapterstyle{fred}{code}` specifies that the chapter style *fred* is defined as the macro `\chs@fred{code}`, and `\chapterstyle{fred}` calls the macro `\chs@fred`.

```
3779 \newcommand{\makechapterstyle}[2]{\@namedef{chs@#1}{\@chs@def@ult #2}}
3780 \newcommand{\chapterstyle}[1]{\@nameuse{chs@#1}}
3781
```

Set the *default* chapter style.

```
3782 \makechapterstyle{default}{%
3783 %% \setlength{\beforechapskip}{50pt}
3784 %% \def\chapterheadstart{\vspace*{\beforechapskip}}
3785 %% \def\chapnamefont{\normalfont\huge\bfseries}
3786 %% \def\printchaptername{\chapnamefont \@chapapp}
3787 %% \def\chapternamenum{\space}
3788 %% \def\chapnumfont{\normalfont\huge\bfseries}
3789 %% \def\printchapternum{\chapnumfont \thechapter}
3790 %% \setlength{\midchapskip}{20pt}
3791 %% \def\afterchapternum{\par\nobreak\vskip \midchapskip}
3792 %% \def\printchapternonum{}
3793 %% \def\chapttitlefont{\normalfont\Huge\bfseries}
3794 %% \def\printchaptertitle#1{\chapttitlefont #1}
3795 %% \setlength{\afterchapskip}{40pt}
3796 %% \def\afterchaptertitle{\par\nobreak\vskip \afterchapskip}}
3797 \chapterstyle{default}
3798
```

`\chs@section` The *section* chapter style. It prints the heading as though it were a section.

```
3799 \makechapterstyle{section}{%
3800 \chapterstyle{default}
3801 \renewcommand{\printchaptername}{}
3802 \renewcommand{\chapternamenum}{}
3803 \renewcommand{\chapnumfont}{\normalfont\Huge\bfseries}
3804 \renewcommand{\printchapternum}{\chapnumfont \thechapter\space}
3805 \renewcommand{\afterchapternum}{}
3806
```

`\chs@article` The *article* chapter style. It prints the heading as though it were a section in an article class document.

```
3807 \makechapterstyle{article}{%
3808 \chapterstyle{default}
3809 \setlength{\beforechapskip}{3.5ex \@plus 1ex \@minus .2ex}
3810 \renewcommand*{\chapterheadstart}{\vspace{\beforechapskip}}
3811 \setlength{\afterchapskip}{2.3ex \@plus .2ex}
3812 \renewcommand{\printchaptername}{}
3813 \renewcommand{\chapternamenum}{}
3814 \renewcommand{\chapttitlefont}{\normalfont\Large\bfseries}
3815 \renewcommand{\chapnumfont}{\chapttitlefont}
3816 \renewcommand{\printchapternum}{\chapnumfont \thechapter\quad}
```

```

3817 \renewcommand{\afterchapternum}{}
3818

```

`\chs@reparticle` The *reparticle* chapter style. It replicates the appearance of a `\section` in an article class document.

```

3819 \makechapterstyle{reparticle}{%
3820   \chapterstyle{default}
3821   \setlength{\beforechapskip}{3.5ex \@plus 1ex \@minus .2ex}
3822   \renewcommand*{\chapterheadstart}{\vspace{\beforechapskip}}
3823   \setlength{\afterchapskip}{2.3ex \@plus .2ex}
3824   \renewcommand*{\printchaptername}{}
3825   \renewcommand*{\chapternamenum}{}
3826   \renewcommand*{\chaptitelfont}{\normalfont\Large\bfseries}
3827   \renewcommand*{\chapnumfont}{\chaptitelfont}
3828   \renewcommand*{\printchapternum}{\@hangfrom{\chapnumfont \thechapter\quad}}%
3829   \renewcommand*{\afterchapternum}{}
3830

```

`\reparticle` Call this to get division heads to replicate the article class. For example:
`\ifartopt \reparticle \fi`
 The `\chapter` commands must still be used, though.

```

3831 \newcommand*{\reparticle}{%
3832   \chapterstyle{reparticle}
3833   %%% \setseheadstyle{\normalfont\large\bfseries\raggedright}
3834   \setseheadstyle{\normalfont\large\bfseries\memRTLraggedright}%
3835   %%% \setsubseheadstyle{\normalfont\bfseries\raggedright}
3836   \setsubseheadstyle{\normalfont\bfseries\memRTLraggedright}%
3837 }
3838

```

`\chs@hangnum` The *hangnum* style puts the chapter number in the margin.

```

3839 \makechapterstyle{hangnum}{%
3840   \chapterstyle{default}
3841   \renewcommand*{\chapnumfont}{\chaptitelfont}
3842   \settowidth{\chapindent}{\chapnumfont 999}
3843   \renewcommand*{\printchaptername}{}
3844   \renewcommand*{\chapternamenum}{}
3845   \renewcommand*{\printchapternum}{%
3846     \noindent\llap{\makebox[\chapindent][l]{\chapnumfont \thechapter}}}
3847   \renewcommand*{\afterchapternum}{}
3848

```

`\chapindent` The *companion* style is like that in the *L^AT_EX Companion* series. It requires the `\chs@companion` length. The user needs to be careful to have a wide enough spine margin on verso pages if the title may appear on a verso page.

```

3849 \newlength{\chapindent}
3850
3851 \makechapterstyle{companion}{%
3852   \chapterstyle{default}

```

```

3853 \renewcommand*{\chapnamefont}{\normalfont\LARGE\scshape}
3854 \renewcommand*{\printchaptername}{\raggedleft\chapnamefont \@chapapp}
3855 \renewcommand*{\chapnumfont}{\normalfont\Huge}
3856 \setlength{\chapindent}{\marginparsep}
3857 \addtolength{\chapindent}{\marginparwidth}
3858 \renewcommand*{\printchaptertertitle}[1]{%
3859   \begin{adjustwidth}{-}\chapindent}
3860   \raggedleft \chaptitelfont ##1\par\nobreak
3861   \end{adjustwidth}}
3862

```

\chs@demo An exotic chapter style for demonstration purposes.

```

3863 \makechapterstyle{demo}{%
3864   \chapterstyle{default}
3865   \renewcommand*{\printchaptername}{\centering}
3866   \renewcommand*{\printchapternum}{\chapnumfont \numtoName{\c@chapter}}
3867   \renewcommand*{\chaptitelfont}{\normalfont\Huge\sffamily}
3868   \renewcommand*{\printchaptertertitle}[1]{%
3869     \hrule\vskip\onelineskip \raggedleft \chaptitelfont ##1}
3870   \renewcommand*{\afterchaptertertitle}%
3871     {\vskip\onelineskip \hrule\vskip \afterchapskip}}
3872

```

\chs@bianchi Posted to CTT on 2003/12/09, *New chapter style: chapter vs chapter** by Stefano Bianchi.

```

3873 \makechapterstyle{bianchi}{%
3874   \chapterstyle{default}
3875   \renewcommand*{\chapnamefont}{\normalfont\Large\sffamily\itshape}
3876   \renewcommand*{\chapnumfont}{\normalfont\huge}
3877   \renewcommand*{\printchaptername}{%
3878     \chapnamefont\centering\@chapapp}
3879   \renewcommand*{\printchapternum}{\chapnumfont \textit{\thechapter}}
3880   \renewcommand*{\chaptitelfont}{\normalfont\Huge\sffamily}
3881   \renewcommand*{\printchaptertertitle}[1]{%
3882     \hrule\vskip\onelineskip \centering \chaptitelfont\textbf{##1}\par}
3883   \renewcommand*{\afterchaptertertitle}{\vskip\onelineskip \hrule\vskip
3884     \afterchapskip}
3885   \renewcommand*{\printchapternonum}{%
3886     \vphantom{\chapnumfont \textit{9}}\afterchapternum}}
3887

```

\chs@bringhurst My bringhurst style.

```

3888 \makechapterstyle{bringhurst}{%
3889   \chapterstyle{default}
3890   \renewcommand*{\chapterheadstart}{}
3891   \renewcommand*{\printchaptername}{}
3892   \renewcommand*{\chapternamenum}{}
3893   \renewcommand*{\printchapternum}{}
3894   \renewcommand*{\afterchapternum}{}

```

```

3895 \renewcommand*{\printchaptertitle}[1]{%
3896   %% \raggedright\Large\scshape\MakeLowercase{##1}}
3897   \memRTLraggedright\Large\scshape\MakeLowercase{##1}}
3898 \renewcommand*{\afterchaptertitle}{%
3899 \vskip\onelineskip \hrule\vskip\onelineskip}}
3900

```

\chs@brotherton An extremely simple chapterstyle created by William Adams for Mike Brotherton's science fiction novel *Star Dragon*, posted to CTT 2006/12/09, *An example of a novel?*

```

3901 \makechapterstyle{brotherton}{%
3902   \chapterstyle{default}
3903   \renewcommand*{\printchapternum}{\chapnumfont
3904     \ifanappendix \thechapter \else \numtoName{\c@chapter}\fi}}
3905

```

\chs@chappell My Chappell style

```

3906 \makechapterstyle{chappell}{%
3907   \chapterstyle{default}
3908   \setlength{\beforechapskip}{0pt}
3909   \renewcommand*{\chapnamefont}{\large\centering}
3910   \renewcommand*{\chapnumfont}{\large}
3911   \renewcommand*{\printchapternonum}{%
3912     \vphantom{\printchaptername \chapnumfont 1}
3913     \afterchapternum
3914     \vskip \onelineskip \vskip -\topskip}
3915   \renewcommand*{\chapttitlefont}{\Large\itshape}
3916   \renewcommand*{\printchaptertitle}[1]{%
3917     \hrule\vskip\onelineskip \centering\chapttitlefont ##1}}
3918

```

\chs@culver By me in an answer to Christopher Culver on CTT, *"Biblical" formatting, how?* on 2004/03/29, where I called it the 'biblical' style.

```

3919 \makechapterstyle{culver}{%
3920   \chapterstyle{default}
3921   \chapterstyle{article}%
3922   \renewcommand*{\thechapter}{\Roman{chapter}}
3923   \renewcommand*{\printchapternum}{% center number/title
3924     \centering\chapnumfont \thechapter\space\space}%
3925   \renewcommand*{\printchapternonum}{\centering}
3926   \renewcommand*{\clearforchapter}{}% no new page
3927   \aliaspagestyle{chapter}{headings}}% no special pagestyle
3928

```

\chs@dash A very simple style of mine but I couldn't think of a good name for it.

```

3929 \makechapterstyle{dash}{%
3930   \chapterstyle{default}
3931   \setlength{\beforechapskip}{5\onelineskip}
3932   \renewcommand*{\printchaptername}{%

```



```

3933 \renewcommand*{\chapternamenum}{%
3934 \renewcommand*{\chapnumfont}{\normalfont\large}
3935 \settoheight{\midchapskip}{\chapnumfont 1}
3936 \renewcommand*{\printchapternum}{\centering \chapnumfont
3937 \rule[0.5\midchapskip]{1em}{0.4pt} \thechapter\
3938 \rule[0.5\midchapskip]{1em}{0.4pt}}
3939 \renewcommand*{\afterchapternum}{\par\nobreak\vskip 0.5\onelineskip}
3940 \renewcommand*{\printchapternonum}{\centering
3941 \vphantom{\chapnumfont 1}\afterchapternum}
3942 \renewcommand*{\chaptitelfont}{\normalfont\Large}
3943 \renewcommand*{\printchaptertitle}[1]{\centering \chaptitelfont ##1}
3944 \setlength{\afterchapskip}{2.5\onelineskip}}
3945

```

\chs@demo2 My second version of the demo chapterstyle.

```

3946 \makechapterstyle{demo2}{%
3947 \chapterstyle{default}
3948 \renewcommand*{\printchaptername}{\centering}
3949 \renewcommand*{\printchapternum}{\chapnumfont
3950 \ifanappendix \thechapter \else \numtoName{\c@chapter}\fi}
3951 \renewcommand*{\chaptitelfont}{\normalfont\Huge\sffamily}
3952 \renewcommand*{\printchaptertitle}[1]{%
3953 \hrule\vskip\onelineskip \raggedleft \chaptitelfont ##1}
3954 \renewcommand*{\afterchaptertitle}{%
3955 \vskip\onelineskip \hrule\vskip \afterchapskip}
3956 \setlength{\beforechapskip}{3\baselineskip}
3957 \renewcommand*{\printchapternonum}{%
3958 \vphantom{\chapnumfont One}
3959 \afterchapternum%
3960 \vskip\topskip}
3961 \setlength{\beforechapskip}{2\onelineskip}}
3962

```

\chs@demo3 My third version of the demo chapterstyle.

```

3963 \makechapterstyle{demo3}{%
3964 \chapterstyle{default}
3965 \renewcommand*{\printchaptername}{\centering}
3966 \renewcommand*{\chapnumfont}{\normalfont\HUGE\itshape}
3967 \renewcommand*{\printchapternum}{\chapnumfont
3968 \ifanappendix \thechapter \else \numtoName{\c@chapter}\fi}
3969 \renewcommand*{\chaptitelfont}{\normalfont\Huge\sffamily}
3970 \renewcommand*{\printchaptertitle}[1]{%
3971 \hrule\vskip\onelineskip \raggedleft \chaptitelfont ##1}
3972 \renewcommand*{\afterchaptertitle}{%
3973 \vskip\onelineskip \hrule\vskip \afterchapskip}
3974 \setlength{\beforechapskip}{0pt}
3975 \setlength{\midchapskip}{2\onelineskip}
3976 \setlength{\afterchapskip}{2\onelineskip}
3977 \renewcommand*{\printchapternonum}{%
3978 \vphantom{\chapnumfont One}

```

```

3979 \afterchapternum%
3980 \vskip\topskip}}
3981

```

\chs@ell Another of my styles which I'll probably use for my next book.

```

3982 \makechapterstyle{ell}{%
3983 \chapterstyle{default}
3984 \renewcommand*{\chapnumfont}{\normalfont\HUGE\sffamily}
3985 \renewcommand*{\chapttitlefont}{\normalfont\huge\sffamily}
3986 \settowidth{\chapindent}{\chapnumfont 111}
3987 \renewcommand*{\chapterheadstart}{\begingroup
3988 \vspace*{\beforechapskip}%
3989 \begin{adjustwidth}{-}\chapindent}%
3990 \hrulefill
3991 \smash{\rule{0.4pt}{15mm}}
3992 \end{adjustwidth}\endgroup}
3993 \renewcommand*{\printchaptername}{%
3994 \renewcommand*{\chapternamenum}{%
3995 \renewcommand*{\printchapternum}{%
3996 \begin{adjustwidth}{-}\chapindent}
3997 \hfill
3998 \raisebox{10mm}[0pt][0pt]{\chapnumfont \thechapter}%
3999 \hspace*{1em}
4000 \end{adjustwidth}\vspace*{-3.0\onelineskip}}
4001 \renewcommand*{\printchaptertitle}[1]{%
4002 \vskip\onelineskip
4003 \raggedleft {\chapttitlefont ##1}\par\nobreak}}
4004

```

\chs@ger Posted to CTT on 2002/04/12 *Fancy Headings, Chapter Headings* by Gerardo Garcia.

```

4005 \makechapterstyle{ger}{%
4006 \chapterstyle{default}
4007 \renewcommand*{\chapterheadstart}{\vspace*{\beforechapskip}
4008 \mbox{\rule[0pt]{\textwidth}{0.4pt}\par}
4009 \setlength{\midchapskip}{20pt}
4010 \renewcommand*{\printchaptertitle}[1]{\chapttitlefont ##1
4011 \rule[5pt]{\textwidth}{0.4pt}}}
4012

```

\chs@lyhne Posted to CTT 2006/02/09 *Glossary*, by Anders Lyhne. It requires the `graphicx` package. I have modified it to remove the new length and adjusted the unnumbered appearance.

```

4013 \makechapterstyle{lyhne}{% needs graphicx package
4014 \chapterstyle{default}
4015 \setlength{\beforechapskip}{1.5cm}
4016 \setlength{\afterchapskip}{1cm}
4017 \setlength{\midchapskip}{2cm}
4018 \renewcommand*{\chapnamefont}{\normalfont\normalsize\scshape\raggedleft}

```

```

4019 \renewcommand*{\chaptitlefont}{\normalfont\normalsize\bfseries\sffamily\raggedleft}
4020 \renewcommand*{\chapternamenum}{%
4021 \renewcommand*{\printchapternum}{\makebox[0pt][l]{\hspace{0.2em}%
4022 \resizebox{!}{2ex}{\chapnamefont\bfseries\sffamily\thechapter}}}%
4023 \renewcommand*{\afterchapternum}{\par\hspace{1.5cm}\hrule\vspace{0.2cm}}
4024 \renewcommand*{\printchapternonum}{\vphantom{\chapnamefont 1}\afterchapternum}
4025 \renewcommand*{\afterchaptertitle}{\vskip 0.2cm
4026 \hrule\vskip\afterchapskip}}
4027

```

`\chs@madsen` Posted to CTT on 2003/12/09, *New chapter style: chapter vs chapter** by Lars Madsen. This requires the `graphicx` package,

```

4028 \makechapterstyle{madsen}{% requires graphicx package
4029 \chapterstyle{default}
4030 \renewcommand*{\chapnamefont}{%
4031 \normalfont\Large\scshape\raggedleft}
4032 \renewcommand*{\chaptitlefont}{%
4033 \normalfont\Huge\bfseries\sffamily\raggedleft}
4034 \renewcommand*{\chapternamenum}{%
4035 \renewcommand*{\printchapternum}{%
4036 \makebox[0pt][l]{\hspace{0.4em}
4037 \resizebox{!}{4ex}{%
4038 \chapnamefont\bfseries\sffamily\thechapter}
4039 }%
4040 }%
4041 \renewcommand*{\printchapternonum}{%
4042 \chapnamefont \phantom{\printchaptername \chapternamenum \printchapternum}
4043 \afterchapternum %
4044 }%
4045 \renewcommand*{\afterchapternum}{%
4046 \par\hspace{1.5cm}\hrule\vskip\midchapskip}}
4047

```

`\colorchapnum` Posted to CTT 2006/01/31, *Chapter style*, by Troels Pedersen. It requires the `graphicx` package and possibly the `color` package. I have used a `\sidebar` instead of the original `\marginpar`, and eliminated a new length. The original color specification, the same for the title and number, can now be set via `\colorchapnum` and `\colorchaptitle`.

```

4048 \newcommand*{\colorchapnum}{%
4049 \newcommand*{\colorchaptitle}{%
4050 \makechapterstyle{pedersen}{%
4051 \chapterstyle{default}
4052 \setlength{\beforechapskip}{-20pt}
4053 \setlength{\afterchapskip}{10pt}
4054 \renewcommand*{\chapnamefont}{\normalfont\Large\itshape}
4055 \renewcommand*{\chapnumfont}{\normalfont\HUGE\itshape\colorchapnum}
4056 \renewcommand*{\chaptitlefont}{\normalfont\huge\itshape\colorchaptitle}
4057 \renewcommand*{\afterchapternum}{%
4058 \renewcommand*{\printchaptername}{%

```

```

4059 \setlength{\midchapskip}{20mm}% was \numberheight
4060 \renewcommand*{\chapternamenum}{%
4061 \renewcommand*{\printchapternum}{%
4062 \sidebar{\raisebox{0pt}[0pt][0pt]{\makebox[0pt][l]{%
4063 \resizebox{!}{\midchapskip}{\chapnumfont\thechapter}}}}
4064 \renewcommand*{\printchaptertitle}[1]{\chaptitelfont ##1}
4065

```

\chs@southall Style provided by Thomas Dye. I have modified the original to eliminate the use of new lengths.

```

4066 %% Thomas Dye's southall chapter style
4067 \makechapterstyle{southall}{%
4068 \chapterstyle{default}
4069 \setlength{\afterchapskip}{5\baselineskip}
4070 \setlength{\beforechapskip}{36pt}% \headindent
4071 \setlength{\midchapskip}{\textwidth}% \rightblock
4072 \addtolength{\midchapskip}{-\beforechapskip}
4073 \renewcommand*{\chapterheadstart}{\vspace*{2\baselineskip}}
4074 %% \renewcommand*{\chaptitelfont}{\huge\rmfamily\raggedright}
4075 \renewcommand*{\chaptitelfont}{\huge\rmfamily\memRTLraggedright}
4076 \renewcommand*{\chapnumfont}{\chaptitelfont}
4077 \renewcommand*{\printchaptername}{%
4078 \renewcommand*{\chapternamenum}{%
4079 \renewcommand*{\afterchapternum}{%
4080 \renewcommand*{\printchapternum}{%
4081 \begin{minipage}[t][\baselineskip][b]{\beforechapskip}
4082 {\vspace{0pt}\chapnumfont%%\figureversion{lining}
4083 \thechapter}
4084 \end{minipage}}
4085 \renewcommand*{\printchaptertitle}[1]{%
4086 \hfill\begin{minipage}[t]{\midchapskip}
4087 {\vspace{0pt}\chaptitelfont ##1\par}\end{minipage}}
4088 \renewcommand*{\afterchaptertitle}{%
4089 \par\vspace{\baselineskip}%
4090 \hrulefill \par\nobreak\noindent \vskip \afterchapskip}}
4091

```

\chs@thatcher Original posted to CTT on 2006/01/18 by Scott Thatcher, *memoir: chapter headings capitalize math symbols*. I have modified it to cater for multiline titles, appendices, and unnumbered chapters.

```

4092 \makechapterstyle{thatcher}{%
4093 \chapterstyle{default}
4094 \renewcommand*{\chapterheadstart}{%
4095 \renewcommand*{\printchaptername}{%
4096 \centerline{\chapnumfont{\@chapapp\ \thechapter}}}
4097 \renewcommand*{\chapternamenum}{%
4098 \renewcommand*{\chapnumfont}{\normalfont\scshape\MakeLowercase}
4099 \renewcommand*{\printchapternum}{%
4100 \renewcommand*{\afterchapternum}{%

```

```

4101 \par\centerline{\parbox{0.5in}{\hrulefill}}\par}
4102 \renewcommand*{\printchapternonum}{%
4103 \vphantom{\chapnumfont \@chapapp 1}\par
4104 \parbox{0.5in}{}\par}
4105 \renewcommand*{\chapttitlefont}{\normalfont\large}
4106 \renewcommand*{\printchaptertitle}[1]{%
4107 \centering \chapttitlefont\MakeUppercase{##1}}
4108

```

`\chs@veelo` This, from Baastian Veelo, has been noted in the documentation for quite a time. I have modified the original to eliminate the need for extra lengths. It well suits trimmed documents. We are scaling the chapter number, which most DVI viewers will not display accurately. It requires the `graphicx` package.

```

4109 \makechapterstyle{veelo}{%
4110 % \chapterstyle{default}
4111 \setlength{\afterchapskip}{40pt}
4112 \renewcommand*{\chapterheadstart}{\vspace*{40pt}}
4113 \renewcommand*{\afterchapternum}{\par\nobreak\vskip 25pt}
4114 \renewcommand*{\chapnamefont}{\normalfont\LARGE\flushright}
4115 \renewcommand*{\chapnumfont}{\normalfont\HUGE}
4116 \renewcommand*{\chapttitlefont}{\normalfont\HUGE\bfseries\flushright}
4117 \renewcommand*{\printchaptername}{%
4118 \chapnamefont\MakeUppercase{\@chapapp}}
4119 \renewcommand*{\chapternamenum}{%
4120 % \setlength{\numberheight}{18mm}
4121 % \setlength{\barlength}{\paperwidth}
4122 % \addtolength{\barlength}{-\textwidth}
4123 % \addtolength{\barlength}{-\spinewidth}
4124 \setlength{\beforechapskip}{18mm}% \numberheight
4125 \setlength{\midchapskip}{\paperwidth}% \barlength
4126 \addtolength{\midchapskip}{-\textwidth}
4127 \addtolength{\midchapskip}{-\spinewidth}
4128 \renewcommand*{\printchapternum}{%
4129 \makebox[0pt][l]{%
4130 \hspace{.8em}%
4131 \resizebox{!}{\beforechapskip}{\chapnumfont \thechapter}%
4132 \hspace{.8em}%
4133 \rule{\midchapskip}{\beforechapskip}%
4134 }%
4135 }%
4136 \makeoddfoot{plain}{}{\thechapter}}
4137

```

`\chs@verville` I posted the original to CTT on 2005/01/18, *Headers and special formatting of sections*, in answer to a question by Guy Verville. This version caters for unnumbered chapters.

```

4138 \makechapterstyle{verville}{%
4139 % \chapterstyle{default}
4140 \setlength{\beforechapskip}{0pt}

```

```

4141 \renewcommand*{\printchaptername}{}
4142 \renewcommand*{\printchapternum}{%
4143   \hrule \vskip 0.5\onelineskip
4144   \Huge \centering \thechapter.\ }
4145 \renewcommand*{\printchapternonum}{%
4146   \hrule \vskip 0.5\onelineskip
4147   \Huge \centering}
4148 \renewcommand*{\afterchapternum}{}
4149 \setlength{\midchapskip}{0pt}
4150 \renewcommand*{\printchaptertitle}[1]{%
4151   #1 \par
4152   \vskip 0.5\onelineskip
4153   \hrule}}
4154

```

\chs@crosshead A centered chapterstyle (from Bringhurst)

```

4155 \makechapterstyle{crosshead}{%
4156   \setlength{\beforechapskip}{2\onelineskip}%
4157   \renewcommand*{\chapterheadstart}{\vspace{\beforechapskip}}%
4158   \setlength{\afterchapskip}{2\onelineskip \@plus .2\onelineskip
4159     \@minus 0.2\onelineskip}%
4160   \renewcommand*{\printchaptername}{}%
4161   \renewcommand*{\chapternamenum}{}%
4162   \renewcommand*{\chapnumfont}{\normalfont\LARGE\bfseries}%
4163   \renewcommand*{\chapttitlefont}{\chapnumfont}%
4164   \renewcommand*{\printchapternum}{%
4165     \centering\chapnumfont \thechapter\quad}%
4166   \renewcommand*{\afterchapternum}{}%
4167   \renewcommand*{\printchapternonum}{\centering}}
4168

```

\chs@dowding A centered chapterstyle (from Dowding's *Finer Points*)

```

4169 \makechapterstyle{dowding}{%
4170   \setlength{\beforechapskip}{2\onelineskip}%
4171   \setlength{\afterchapskip}{1.5\onelineskip \@plus .1\onelineskip
4172     \@minus 0.167\onelineskip}%
4173   \renewcommand*{\chapnamefont}{\normalfont}%
4174   \renewcommand*{\chapnumfont}{\chapnamefont}%
4175   \renewcommand*{\printchapternum}{\centering\chapnumfont \ifanappendix \thechapter
4176     \else \numtoName{c@chapter}\fi}%
4177   \renewcommand*{\chapttitlefont}{\normalfont\itshape\huge\centering}%
4178   \renewcommand*{\printchapternonum}{%
4179     \vphantom{\printchaptername}\vskip\midchapskip}}
4180

```

\chs@komalike A chapterstyle approximating the KOMA script style (scrbook.cls)

```

4181 \makechapterstyle{komalike}{%
4182   \setlength{\beforechapskip}{2\onelineskip}%
4183   \setlength{\afterchapskip}{1.5\onelineskip \@plus .1\onelineskip}

```

```

4184 \@minus 0.167\onelineskip}%
4185 \renewcommand*{\printchaptername}{}%
4186 \renewcommand*{\chapternamenum}{}%
4187 \renewcommand*{\chapnumfont}{\normalfont\LARGE\sffamily\bfseries}%
4188 \renewcommand*{\printchapternum}{\chapnumfont \thechapter\space}%
4189 \renewcommand*{\afterchapternum}{}%
4190 \renewcommand*{\chaptitelfont}{\normalfont\LARGE\sffamily\bfseries}}
4191

```

`\chs@ntglike` A chapterstyle approximating the NTG style (boek.cls)

```

4192 \makechapterstyle{ntglike}{%
4193 \setlength{\beforechapskip}{50pt \@plus 20pt}%
4194 \renewcommand*{\chapnamefont}{\normalfont\Large\bfseries}%
4195 \renewcommand*{\chapnumfont}{\normalfont\Large\bfseries}%
4196 \renewcommand*{\chaptitelfont}{\normalfont\Large\bfseries}}
4197

```

`\chs@tandh` A chapterstyle based on Thames & Hudson *Typography*.

```

4198 \makechapterstyle{tandh}{%
4199 \setlength{\beforechapskip}{1\onelineskip}%
4200 \setlength{\afterchapskip}{2\onelineskip \@plus .1\onelineskip
4201 \@minus 0.167\onelineskip}%
4202 \renewcommand*{\printchaptername}{}%
4203 \renewcommand*{\chapternamenum}{}%
4204 \renewcommand*{\chapnumfont}{\normalfont\huge\bfseries}%
4205 \renewcommand*{\printchapternum}{\chapnumfont \thechapter\quad}%
4206 \renewcommand*{\afterchapternum}{}%
4207 %%% \renewcommand*{\chaptitelfont}{\chapnumfont\raggedright}}
4208 \renewcommand*{\chaptitelfont}{\chapnumfont\memRTLraggedright}}
4209

```

`\chs@wilsondob` A chapterstyle based on Adrian Wilson's *Design of books*.

```

4210 \makechapterstyle{wilsondob}{%
4211 \setlength{\beforechapskip}{2\onelineskip}%
4212 \setlength{\afterchapskip}{4\onelineskip \@plus .1\onelineskip
4213 \@minus 0.167\onelineskip}%
4214 \renewcommand*{\printchaptername}{}%
4215 \renewcommand*{\chapternamenum}{}%
4216 \renewcommand*{\chapnumfont}{\normalfont\Huge\itshape}%
4217 \renewcommand*{\printchapternum}{\raggedleft\chapnumfont \thechapter\quad}%
4218 \renewcommand*{\afterchapternum}{}%
4219 \renewcommand*{\chaptitelfont}{\chapnumfont}%
4220 \renewcommand*{\printchapternonum}{\raggedleft}}
4221

```

12.8 Lower level headings

These commands all make use of `\@startsection`. However, for the purposes of the class the kernel version needs modification to support:

- make short pages (where a section heading is moved to the top of the next page) ragged bottom;
- provide headings with two optional arguments.

We will tweak `\@startsection` so that a short page (where a section heading is moved from the bottom of the page to the top of the next) can be set `\raggedbottom`.

```
\ifraggedbottomsection \raggedbottomsectiontrue for ragged short pages.
\raggedbottomsectiontrue 4222 \newif\ifraggedbottomsection
\raggedbottomsectionfalse 4223 \raggedbottomsectionfalse
\raggedbottomsection 4224 \newcommand*\raggedbottomsection{\raggedbottomsectiontrue}
\normalbottomsection 4225 \newcommand*\normalbottomsection{\raggedbottomsectionfalse}
4226
```

`\bottomsectionsip` Decreasing this length increases short page bottom flushness.

```
4227 \newlength{\bottomsectionsip}
4228 \setlength{\bottomsectionsip}{10mm}
4229
```

We will add a second optional argument for `\section`, etc. See the CTT thread ‘*Long headers*’ 15 Jan 2003.

The kernel `\@dblarg` is used in situations where the default value for an optional argument is the same as the required argument. Schematically,

`\@dblarg -> Main, Main`

The macros `\@trplargomm` and `\@trplargoom` are extensions to this idea for two optional arguments.

```
\@trplargomm \@trplargomm -> Opt, Main, Main
\@xtrplargomm 4230 \newcommand{\@trplargomm}[1]{%
\@xxtrplarg 4231 \@ifnextchar[{\@xtrplargomm{#1}}%
4232 {\@xxtrplarg{#1}}}
4233 \long\def\@xtrplargomm#1[#2]{\@dblarg{#1[#2]}}
4234 \newcommand{\@xxtrplarg}[2]{#1[#2][#2][#2]}

\@trplargoom \@trplargoom -> Opt, Opt, Main
\@xtrplargoom 4235 \newcommand{\@trplargoom}[1]{%
4236 \@ifnextchar[{\@xtrplargoom{#1}}%
4237 {\@xxtrplarg{#1}}}
4238 \long\def\@xtrplargoom#1[#2]{%
4239 \@ifnextchar[#{#1[#2]}%
4240 {#1[#2][#2][#2]}
4241

\memsecinfo \memsecinfo{secname}{\thenum}{toc}{head}{title}
\memsecstarinfo \memsecstarinfo{secname}{title}

4242 \newcommand{\memsecinfo}[5]{%
4243 \newcommand{\memsecstarinfo}[2]{%
4244
```


`\@startsection` Change kernel `\@startsection` to:
`\m@msecn@me`

- Make short pages raggedbottom. This is based on the thread *Can \flushbottom and \section be made to live together?* on CTT in September 2002.

- add an extra optional argument.

The original is in `ltsect.dtx`.

```
\@startsection{<name>}{<level>}{<indent>}{<beforeskip>}{<afterskip>}{<style>}
```

```
4245 \renewcommand{\@startsection}[6]{%
```

Do raggedbottom stuff.

```
4246 \ifraggedbottomsection\if@nobreak\else
4247   \vskip\z@\@plus\bottomsectionskip
4248   \penalty\z@
4249   \vskip\z@\@plus -\bottomsectionskip
4250 \fi\fi
```

Save the section name.

```
4251 \def\m@msecn@me{#1}%
```

The original code.

```
4252 \if@noskipsec \leavevmode \fi
4253 \par
4254 \@tempskipa #4\relax
4255 \@afterindenttrue
4256 \ifdim \@tempskipa <\z@
4257   \@tempskipa -\@tempskipa \@afterindentfalse
4258 \fi
4259 \if@nobreak
4260   \everypar{}%
4261 \else
4262   \addpenalty\@secpenalty\addvspace\@tempskipa
4263 \fi
```

For the extra optional argument, change the original `\@dblarg{\@sect{...}`
below to `\@trplargoom{\M@sect{...}`

```
4264 \@ifstar
4265   {\@ssect{#3}{#4}{#5}{#6}}%
4266   {\@trplargoom{\M@sect{#1}{#2}{#3}{#4}{#5}{#6}}}%
4267
```

`\M@sect` At least the `hyperref` and `nameref` packages, and possibly other packages, modify the kernel `\@sect` assuming 8 args. That's why I've called my version, which has 9 args, something else (i.e., `\M@sect`). This version also has the support for title referencing.

```
\M@sect{<name>}{<level>}{<indent>}{<beforeskip>}{<afterskip>}{<style>}{<toc>}[<head>][<title>]}
```

```

4268 \def\M@ssect#1#2#3#4#5#6[#7][#8]#9{%
4269   \ifheadnameref\M@getttitle{#8}\else\M@getttitle{#7}\fi
4270   \ifnum #2>\c@secnumdepth
4271     \let\@svsec\@empty
4272     \memsecinfo{#1}{#7}{#8}{#9}%
4273   \else
4274     \refstepcounter{#1}%
4275     \protected@edef\@svsec{\@secntformat{#1}\relax}%
4276     \memsecinfo{#1}{\@nameuse{the#1}}{#7}{#8}{#9}%
4277   \fi
4278   \@tempskipa #5\relax
4279   \ifdim \@tempskipa>\z@
4280     \begingroup
4281       #6{%
4282         \@hangfrom{\hskip #3\relax\@svsec}%
4283         \interlinepenalty \@M #9\@par}%
4284     \endgroup
4285     \csname #1mark\endcsname{#8}%
4286     \addcontentsline{toc}{#1}{%
4287       \ifnum #2>\c@secnumdepth \else
4288         \protect\numberline{\csname the#1\endcsname}%
4289       \fi
4290       #7}%
4291   \else
4292     \def\@svsechd{%
4293       #6{\hskip #3\relax
4294         \@svsec #9}%
4295     \csname #1mark\endcsname{#8}%
4296     \addcontentsline{toc}{#1}{%
4297       \ifnum #2>\c@secnumdepth \else
4298         \protect\numberline{\csname the#1\endcsname}%
4299       \fi
4300       #7}}%
4301   \fi
4302   \@xsect{#5}}
4303

```

`\@ssect` Add hook for name reference to `\section*` etc.

`\@ssect{<indent>}{<beforeskip>}{<afterskip>}{<style>}{<title>}`

```

4304 \let\@mem@old@ssect\@ssect
4305 \def\@ssect#1#2#3#4#5{%
4306   \M@getttitle{#5}%
4307   \memsecstarinfo{\m@msecn@me}{#5}%
4308   \@mem@old@ssect{#1}{#2}{#3}{#4}{#5}}
4309

```

`\section` A normal heading with white space above and below and no indentation of the first paragraph. By default the heading is set in a `\Large\bfseries` font.

```

4310 \newcommand{\section}{%

```

```

4311 \sechook%
4312 \@startsection{section}{1}% level 1
4313     {\secindent}% heading indent
4314     {\beforesecskip}% skip before the heading
4315     {\aftersecskip}% skip after the heading
4316     {\normalfont\seheadstyle}} % font

```

`\sechook` `\sechook` is called at the start of a `\section` and `\setsechook{<code>}`
`\setsechook` redefines it.

```

4317 \newcommand{\sechook}{}
4318 \newcommand{\setsechook}[1]{\renewcommand{\sechook}{#1}}

```

`\secindent` `\secindent` is the indentation of the section heading from the left margin,
`\beforesecskip` `\beforesecskip` and `\aftersecskip` specify the white space before and after
`\aftersecskip` the heading, and `\seheadstyle` specifies the heading style. These are set to the
`\seheadstyle` default values for the book class, except that the heading will be `\raggedright`,
`\setsecindent` thus preventing hyphenation. The `\set...` commands are for the user to change
`\setbeforesecskip` the values.

```

\setaftersecskip 4319 \newlength{\secindent}
\setseheadstyle 4320 \newcommand{\setsecindent}[1]{\setlength{\secindent}{#1}}
4321     \setsecindent{\z@}
4322 \newskip\beforesecskip
4323 \newcommand{\setbeforesecskip}[1]{\setlength{\beforesecskip}{#1}}
4324     \setbeforesecskip{-3.5ex \@plus -1ex \@minus -.2ex}
4325 \newskip\aftersecskip
4326 \newcommand{\setaftersecskip}[1]{\setlength{\aftersecskip}{#1}}
4327     \setaftersecskip{2.3ex \@plus .2ex}
4328 \newcommand{\seheadstyle}{}
4329 \newcommand{\setseheadstyle}[1]{\renewcommand{\seheadstyle}{#1}}
4330 %%% \setseheadstyle{\Large\bfseries\raggedright}
4331     \setseheadstyle{\Large\bfseries\memRTLraggedright}
4332

```

`\subsection` A normal heading with white space above and below and no indentation of the
first paragraph. By default the heading is set in a `\large\bfseries` font.

```

4333 \newcommand{\subsection}{%
4334     \subsechook%
4335     \@startsection{subsection}{2}% level 2
4336     {\subsecindent}% heading indent
4337     {\beforesubsecskip}% skip before the heading
4338     {\aftersubsecskip}% skip after the heading
4339     {\normalfont\subseheadstyle}} % font

```

`\subsechook` `\subsechook` is called at the start of a `\subsection` and
`\setsubsechook` `\setsubsechook{<code>}` redefines it.

```

4340 \newcommand{\subsechook}{}
4341 \newcommand{\setsubsechook}[1]{\renewcommand{\subsechook}{#1}}

```

`\subsecindent` The macros for controlling `\subsection` headings.

```

\beforesubsecskip 4342 \newlength{\subsecindent}
\aftersubsecskip 4343 \newcommand{\setsubsecindent}[1]{\setlength{\subsecindent}{#1}}
\subsecheadstyle 4344 \setsubsecindent{\z@}
\setsubsecindent 4345 \newskip\beforesubsecskip
\setbeforesubsecskip 4346 \newcommand{\setbeforesubsecskip}[1]{\setlength{\beforesubsecskip}{#1}}
\setaftersubsecskip 4347 \setbeforesubsecskip{-3.25ex \@plus -1ex \@minus -.2ex}
\setsubsecheadstyle 4348 \newskip\aftersubsecskip
\setsubsecheadstyle 4349 \newcommand{\setaftersubsecskip}[1]{\setlength{\aftersubsecskip}{#1}}
4350 \setaftersubsecskip{1.5ex \@plus .2ex}
4351 \newcommand{\subsecheadstyle}{}
4352 \newcommand{\setsubsecheadstyle}[1]{\renewcommand{\subsecheadstyle}{#1}}
4353 %%% \setsubsecheadstyle{\large\bfseries\raggedright}
4354 \setsubsecheadstyle{\large\bfseries\memRTLraggedright}
4355
\subsubsection A normal heading with white space above and below and no indentation of the
first paragraph. By default the heading is set in a \normalsize\bfseries font.
4356 \newcommand{\subsubsection}{%
4357 \subsubsechook%
4358 \@startsection{subsubsection}{3}% level 3
4359 {\subsubsecindent}% heading indent
4360 {\beforesubsubsecskip}% skip before the heading
4361 {\aftersubsubsecskip}% skip after the heading
4362 {\normalfont\subsubsecheadstyle}} % font

\subsubsechook \subsubsechook is called at the start of a \subsubsection and
\setsubsubsechook \setsubsubsechook{<code>} redefines it.
4363 \newcommand{\subsubsechook}{}
4364 \newcommand{\setsubsubsechook}[1]{\renewcommand{\subsubsechook}{#1}}

\subsubsecindent The macros for controlling \subsubsection headings.
\beforesubsubsecskip 4365 \newlength{\subsubsecindent}
\aftersubsubsecskip 4366 \newcommand{\setsubsubsecindent}[1]{%
\subsubsecheadstyle 4367 \setlength{\subsubsecindent}{#1}}
\setsubsubsecindent 4368 \setsubsubsecindent{\z@}
\setbeforesubsubsecskip 4369 \newskip\beforesubsubsecskip
\setaftersubsubsecskip 4370 \newcommand{\setbeforesubsubsecskip}[1]{%
\setsubsubsecheadstyle 4371 \setlength{\beforesubsubsecskip}{#1}}
\setsubsubsecheadstyle 4372 \setbeforesubsubsecskip{-3.25ex \@plus -1ex \@minus -.2ex}
4373 \newskip\aftersubsubsecskip
4374 \newcommand{\setaftersubsubsecskip}[1]{%
4375 \setlength{\aftersubsubsecskip}{#1}}
4376 \setaftersubsubsecskip{1.5ex \@plus .2ex}
4377 \newcommand{\subsubsecheadstyle}{}
4378 \newcommand{\setsubsubsecheadstyle}[1]{%
4379 \renewcommand{\subsubsecheadstyle}{#1}}
4380 %%% \setsubsubsecheadstyle{\normalsize\bfseries\raggedright}
4381 \setsubsubsecheadstyle{\normalsize\bfseries\memRTLraggedright}
4382

```

`\paragraph` A runin heading with white space above and to the right of the heading. By default the heading is set in a `\normalsize\bfseries` font.

```
4383 \newcommand{\paragraph}{%
4384   \parahook%
4385   \@startsection{paragraph}{4}% level 4
4386       {\paraindent}% heading indent
4387       {\beforeparaskip}% skip before the heading
4388       {\afterparaskip}% skip after the heading
4389       {\normalfont\paraheadstyle}} % font
```

`\parahook` `\parahook` is called at the start of a `\paragraph` and `\setparahook{<code>}` redefines it.

```
4390 \newcommand{\parahook}{}
4391 \newcommand{\setparahook}[1]{\renewcommand{\parahook}{#1}}
```

`\paraindent` The macros for controlling `\paragraph` headings.

```
\beforeparaskip 4392 \newlength{\paraindent}
\afterparaskip 4393 \newcommand{\setparaindent}[1]{\setlength{\paraindent}{#1}}
\paraheadstyle 4394 \setparaindent{\z@}
\setparaindent 4395 \newskip\beforeparaskip
\setbeforeparaskip 4396 \newcommand{\setbeforeparaskip}[1]{\setlength{\beforeparaskip}{#1}}
\setafterparaskip 4397 \setbeforeparaskip{3.25ex \@plus 1ex \@minus .2ex}
\setparaheadstyle 4398 \newskip\afterparaskip
4399 \newcommand{\setafterparaskip}[1]{\setlength{\afterparaskip}{#1}}
4400 \setafterparaskip{-1em}
4401 \newcommand{\paraheadstyle}{}
4402 \newcommand{\setparaheadstyle}[1]{\renewcommand{\paraheadstyle}{#1}}
4403 \setparaheadstyle{\normalsize\bfseries}
4404
```

`\subparagraph` A runin heading with white space above and to the right of the heading. By default the heading is set in a `\normalsize\bfseries` font.

```
4405 \newcommand{\subparagraph}{%
4406   \subparahook%
4407   \@startsection{subparagraph}{5}% level 5
4408       {\subparaindent}% heading indent
4409       {\beforesubparaskip}% skip before the heading
4410       {\aftersubparaskip}% skip after the heading
4411       {\normalfont\subparaheadstyle}} % font
```

`\subparahook` `\subparahook` is called at the start of a `\subparagraph` and `\setsubparahook{<code>}` redefines it.

```
4412 \newcommand{\subparahook}{}
4413 \newcommand{\setsubparahook}[1]{\renewcommand{\subparahook}{#1}}
```

`\subparaindent` The macros for controlling `\subparagraph` headings.

```
\beforesubparaskip 4414 \newlength{\subparaindent}
\aftersubparaskip 4415 \newcommand{\setsubparaindent}[1]{%
\subparaheadstyle 4416   \setlength{\subparaindent}{#1}}
\setsubparaindent
\setbeforesubparaskip
\setaftersubparaskip
\setsubparaheadstyle
```

```

4417 \setsubparaindent{\parindent}
4418 \newskip\beforesubparaskip
4419 \newcommand{\setbeforesubparaskip}[1]{%
4420     \setlength{\beforesubparaskip}{#1}}
4421 \setbeforesubparaskip{3.25ex \@plus 1ex \@minus .2ex}
4422 \newskip\aftersubparaskip
4423 \newcommand{\setaftersubparaskip}[1]{%
4424     \setlength{\aftersubparaskip}{#1}}
4425 \setaftersubparaskip{-1em}
4426 \newcommand{\subparaheadstyle}{}
4427 \newcommand{\setsubparaheadstyle}[1]{%
4428     \renewcommand{\subparaheadstyle}{#1}}
4429 \setsubparaheadstyle{\normalsize\bfseries}
4430

```

`\sethangfrom` The macro `\sethangfrom{<code>}` is a user-level command for changing the definition of `\@hangfrom`. Use #1 in `<code>` for the argument to `\@hangfrom` (or ##1 if used inside another macro).

```

4431 \newcommand{\sethangfrom}[1]{\renewcommand{\@hangfrom}[1]{#1}}

```

`\setsecnumformat` The macro `\setsecnumformat{<code>}` is a user-level command for changing the definition of `\@secntformat`. Use #1 in `<code>` for the argument to `\@secntformat` (or ##1 if used inside another macro).

```

4432 \newcommand{\setsecnumformat}[1]{\renewcommand{\@secntformat}[1]{#1}}

```

`\hangsecnum` These are declarations for putting sectional numbers in the margin, or the
`\defaultsecnum` default sectional number formatting.

```

4433 \newcommand{\hangsecnum}{%
4434     \def\@secntformat##1{\llap{\csname the##1\endcsname\quad}}}
4435 \newcommand{\defaultsecnum}{%
4436     \def\@secntformat##1{\csname the##1\endcsname\quad}}
4437

```

12.8.1 Anonymous headings

`\plainbreak` `\plainbreak{<num>}` generates `<num>` blanks lines and suppresses the indentation of a following paragraph. The starred version, `\plainbreak*`, does not suppress paragraph indentation.

```

4438 \newcommand{\plainbreak}{\@ifstar{\@spbreak}{\@pbreak}}

```

`\@pbreak` These are the internal forms for the two versions of `\plainbreak`. The code for
`\@spbreak` `\@pbreak` is almost a straight copy of code posted to CTT by Donald Arseneau on 2001/03/26.

```

4439 \newcommand*{\@pbreak}[1]{\par
4440     \penalty -100
4441     \vskip #1\onelineskip \@plus 2\onelineskip
4442     \penalty -20
4443     \vskip \z@ \@plus -2\onelineskip

```

```

4444 \afterindentfalse
4445 \afterheading}
4446 \newcommand*{\@spbreak}[1]{\par
4447 \penalty -100
4448 \vskip #1\onelineskip \@plus 2\onelineskip
4449 \penalty -20
4450 \vskip \z@ \@plus -2\onelineskip
4451 \afterindenttrue
4452 \afterheading}
4453

```

`\fancybreak` `\fancybreak{<text>}` typesets `<text>` centered. For example, `\fancybreak{*}\{ * * *\}\{ * }`. It suppresses indentation of the following paragraph. The starred version leaves indentation as is.

```

4454 \newcommand{\fancybreak}{\@ifstar{\@sfbreak}{\@fbreak}}

```

`\@fbreak` These are the internal forms for the two versions of `\fancybreak`.

```

\@sfbreak 4455 \newcommand{\@fbreak}[1]{\par
4456 \penalty -100
4457 \noindent\parbox{\linewidth}{\centering #1}%\null
4458 \par
4459 %% \penalty -20
4460 %% \vskip -\onelineskip
4461 \afterindentfalse
4462 \afterheading}
4463 \newcommand{\@sfbreak}[1]{\par
4464 \penalty -100
4465 \noindent\parbox{\linewidth}{\centering #1}%\null
4466 \par
4467 %% \penalty -20
4468 %% \vskip -\onelineskip
4469 \afterindenttrue
4470 \afterheading}
4471

```

`\plainfancybreak` The `\plainfancybreak{<space>}{<num>}{<text>}` acts like `\plainbreak{<num>}` in the middle of a page and like `\fancybreak{<text>}` at the bottom of a page. The `<space>` argument is the vertical space required for the `<num>` blank lines and some additional lines of text. From experiments, it seems that `<space>` should be at least $(3 + \langle num \rangle)$ lines (`\onelineskip`). There is also a starred version, `\plainfancybreak*`, to match the other starred break commands.

```

4472 \newcommand{\plainfancybreak}{\@ifstar{\@spfbreak}{\@pfbreak}}

```

`\@pfbreak` These are the internal macros for the unstarred and starred versions of `\plainfancybreak`. They essentially do the same thing, except they call, respectively, the unstarred and starred internal versions of `\plainbreak` and `\fancybreak`. The code for checking the amount of space left on the page is from an early, and abandoned, version of the code for `\needspace`.

```

4473 \newcommand{\@pfbreak}[3]{\par
4474   \@tempdimc\pagegoal \advance\@tempdimc-\pagetotal
4475   \ifdim #1>\@tempdimc \@fbreak{#3}\else \@pbreak{#2}\fi}
4476 \newcommand{\@spfbreak}[3]{\par
4477   \@tempdimc\pagegoal \advance\@tempdimc-\pagetotal
4478   \ifdim #1>\@tempdimc \@sfbreak{#3}\else \@spbreak{#2}\fi}
4479

```

From the thread *Customizing section hooks in memoir.cls* on CTT in October 2002 it appeared that `\plainfancybreak` may be fragile. Donald Arseneau said that `\plainfancybreak` guessed at the amount of space available and gave code based on a modified output routine to make it more robust. The following code is based on that thread.

`\pen@ltyabovepfbreak` Penalties for communication with the output routine.

```

\pen@ltybelowpfbreak 4480 \newcommand*{\pen@ltyabovepfbreak}{2}
4481 \newcommand*{\pen@ltybelowpfbreak}{-4}
4482

```

`\pfbreakskip` The vertical space taken by the plain and fancy breaks.

```

4483 \newlength{\pfbreakskip}
4484 \setlength{\pfbreakskip}{2\baselineskip}

```

`\pfbreakdisplay` This is the fancybreak display, which must fit into `\pfbreakskip` vertical space.

```

4485 \newcommand{\pfbreakdisplay}{*\quad*\quad*}
4486

```

`\pfbre@kdispl@y` Typesets `\pfbreakdisplay` vertically and horizontally centered.

```

4487 \def\pfbre@kdispl@y{\vbox to 1\pfbreakskip{\vss
4488   \hb@xt@ \columnwidth{\hss \pfbreakdisplay \hss}%
4489   \vss}}
4490

```

`\nopfbreakOutput` Save the current output routine.

```

4491 \edef\nopfbreakOutput{\the\output}

```

`\pfbreakOutput` Special output to handle the `\pfbreak`.

```

4492 \def\pfbreakOutput{%
4493   \ifnum\outputpenalty=\pen@ltyabovepfbreak
4494     \nopfbreakOutput
4495     \pfbre@kdispl@y
4496     \nobreak
4497     \vskip-\pfbreakskip
4498   \else\ifnum\outputpenalty=\pen@ltybelowpfbreak
4499     \unvbox 255\relax
4500     \nobreak
4501     \vskip-\pfbreakskip
4502     \pfbre@kdispl@y
4503     \break

```



```

4504 \else
4505 \nopfbreakOutput
4506 \fi
4507 \fi}

```

`\output` Use the new `\pfbreak` output routine.

```

4508 \output={\pfbreakOutput}
4509

```

`\pfbreak` Typesets a plain break in the middle of the page, otherwise a fancybreak at either the bottom or top of the page.

```

4510 \newcommand{\pfbreak}{\@ifstar{\@spfbreakgap}{\@pfbreakgap}}

```

`\@pfbreakgap` Unstarred version of `\pfbreak`.

```

4511 \newcommand{\@pfbreakgap}{%
4512 \par {%
4513 \skip@\lastskip
4514 \nobreak
4515 \vskip -\ifdim\prevdepth>\maxdepth \maxdepth
4516 \else\ifdim\prevdepth>-1000pt\prevdepth
4517 \else\ifinner Opt
4518 \else \pagedepth
4519 \fi \fi \fi
4520 \vskip -\skip@
4521 \ifdim\skip@<\pfbreakskip
4522 \advance\skip@ -1\skip@ \advance\skip@ 1\pfbreakskip
4523 \fi
4524 \penalty\pen@ltyabovepfbreak
4525 \vskip\skip@
4526 \penalty\pen@ltybelowpfbreak
4527 }
4528 \@afterindentfalse
4529 \@afterheading
4530 }

```

`\@spfbreakgap` Starred version of `\pfbreak`.

```

4531 \newcommand{\@spfbreakgap}{%
4532 \par {%
4533 \skip@\lastskip
4534 \nobreak
4535 \vskip -\ifdim\prevdepth>\maxdepth \maxdepth
4536 \else\ifdim\prevdepth>-1000pt\prevdepth
4537 \else\ifinner Opt
4538 \else \pagedepth
4539 \fi \fi \fi
4540 \vskip -\skip@
4541 \ifdim\skip@<\pfbreakskip
4542 \advance\skip@ -1\skip@ \advance\skip@ 1\pfbreakskip
4543 \fi

```

```

4544 \penalty\penaltyabovepfbreak
4545 \vskip\skip@
4546 \penalty\penaltybelowpfbreak
4547 }
4548 \@afterindenttrue
4549 \@afterheading
4550 }
4551

```

While on the subject of breaks...

`\noprelistbreak` Putting this immediately before a list (e.g., `itemize`) should prevent a page break at that point.

```

4552 \newcommand*\noprelistbreak{\@nobreaktrue\nopagebreak}
4553

```

12.9 Division head styles

The styles of the division heads should go together. As an aid the class supplies some ready made collections.

`\makeheadstyles` `\makeheadstyles{<name>}{<code>}` creates the `<name>` collection of division head styles, defined by `<code>`. `\headstyles{<name>}` makes the `<name>` division head styles the current styles.

```

\hds@default 4554 \newcommand{\makeheadstyles}[2]{%
4555   \@namedef{hds@#1}{\@hds@def@ult #2}}
4556 \newcommand*\headstyles[1]{\@nameuse{hds@#1}}

```

`\@hds@def@ult` is the default set of division head styles.

```

4557 \newcommand*\@hds@def@ult{%
4558 % Default \cs{book} style
4559 %   \begin{macrocode}
4560 % book
4561 \renewcommand*\beforebookskip{\null\vfil}%
4562 \renewcommand*\midbookskip{\par\vskip 20pt}%
4563 \renewcommand*\afterbookskip{\vfil\newpage}%
4564 \renewcommand*\booknamefont{\normalfont\huge\bfseries}%
4565 \renewcommand*\booknumfont{\normalfont\huge\bfseries}%
4566 \renewcommand*\booktitlefont{\normalfont\Huge\bfseries}%
4567 \renewcommand*\printbookname{\booknamefont \bookname}%
4568 \renewcommand*\booknamenenum{\space}%
4569 \renewcommand*\printbooknum{\booknumfont \thebook}%
4570 \renewcommand*\printbooktitle[1]{\booktitlefont{##1}}%

```

Default `\part` style

```

4571 % part
4572 \renewcommand*\beforepartskip{\null\vfil}%
4573 \renewcommand*\midpartskip{\par\vskip 20pt}%
4574 \renewcommand*\afterpartskip{\vfil\newpage}%
4575 \renewcommand*\partnamefont{\normalfont\huge\bfseries}%

```

```

4576 \renewcommand*{\partnumfont}{\normalfont\huge\bfseries}%
4577 \renewcommand*{\parttitlefont}{\normalfont\Huge\bfseries}%
4578 \renewcommand*{\printpartname}{\partnamefont \partname}%
4579 \renewcommand*{\partnamenum}{\space}%
4580 \renewcommand*{\printpartnum}{\partnumfont \thepart}%
4581 \renewcommand*{\printparttitle}[1]{\parttitlefont{##1}}%

Default \chapter style.
4582 \@chsofdef@ult%                default chapterstyle

Default \section style
4583 % section
4584 \setsechook{}
4585 \setsecindent{\z@}%
4586 \setbeforesecskip{-3.5ex \@plus -1ex \@minus -.2ex}%
4587 \setaftersecskip{2.3ex \@plus .2ex}%
4588 %%% \setsecheadstyle{\Large\bfseries\raggedright}%
4589 \setsecheadstyle{\Large\bfseries\memRTLraggedright}%

Default \subsection style
4590 % subsection
4591 \setsubsechook{}%
4592 \setsubsecindent{\z@}%
4593 \setbeforesubsecskip{-3.25ex \@plus -1ex \@minus -.2ex}%
4594 \setaftersubsecskip{1.5ex \@plus .2ex}%
4595 %%% \setsubsecheadstyle{\large\bfseries\raggedright}%
4596 \setsubsecheadstyle{\large\bfseries\memRTLraggedright}%

Default \subsubsection style
4597 % subsubsection
4598 \setsubsubsechook{}%
4599 \setsubsubsecindent{\z@}%
4600 \setbeforesubsubsecskip{-3.25ex \@plus -1ex \@minus -.2ex}%
4601 \setaftersubsubsecskip{1.5ex \@plus .2ex}%
4602 %%% \setsubsubsecheadstyle{\normalsize\bfseries\raggedright}%
4603 \setsubsubsecheadstyle{\normalsize\bfseries\memRTLraggedright}%

Default \paragraph style
4604 % paragraph
4605 \setparahook{}%
4606 \setparaindent{\z@}%
4607 \setbeforeparaskip{3.25ex \@plus 1ex \@minus .2ex}%
4608 \setafterparaskip{-1em}%
4609 \setparaheadstyle{\normalsize\bfseries}%

Default \paragraph style
4610 % subparagraph
4611 \setsubparahook{}%
4612 \setsubparaindent{\parindent}%
4613 \setbeforesubparaskip{3.25ex \@plus 1ex \@minus .2ex}%
4614 \setaftersubparaskip{-1em}%
4615 \setsubparaheadstyle{\normalsize\bfseries}}
4616

```

Set up and use the default head styles

```
4617 \makeheadstyles{default}{}
4618 \headstyles{default}
4619
```

\addperiod Puts a period at the end of its argument.

```
4620 \newcommand*\addperiod}[1]{#1.}
4621
```

\hds@memman Head styles used in the *The Memoir Class* user guide. In this, as the later ones, only changes from the defaults need specifying.

```
4622 \makeheadstyles{memman}{%
4623 % book changes
4624 \renewcommand*\booknamefont{\normalfont\huge\sffamily}
4625 \renewcommand*\booknumfont{\normalfont\huge\sffamily}
4626 \renewcommand*\booktitlefont{\normalfont\Huge\sffamily}
4627 \renewcommand*\midbookskip{\par\vskip 2\onelineskip}%
4628 % part changes
4629 \renewcommand*\partnamefont{\normalfont\huge\sffamily}
4630 \renewcommand*\partnumfont{\normalfont\huge\sffamily}
4631 \renewcommand*\parttitlefont{\normalfont\Huge\sffamily}
4632 \renewcommand*\midpartskip{\par\vskip 2\onelineskip}%
4633 % chapter
4634 \chapterstyle{demo3}
4635 % section
4636 \setbeforeseccskip{-1.333\onelineskip
4637 \@plus -0.5\onelineskip \@minus -.5\onelineskip}%
4638 \setafterseccskip{0.667\onelineskip \@plus 0.1\onelineskip}%
4639 %%% \setsecheadstyle{\normalfont\scshape\raggedright}%
4640 \setsecheadstyle{\normalfont\scshape\memRTLraggedright}%
4641 % subsection
4642 \setbeforesubseccskip{-0.667\onelineskip
4643 \@plus -0.25\onelineskip \@minus -0.25\onelineskip}%
4644 \setaftersubseccskip{0.333\onelineskip \@plus 0.1\onelineskip}%
4645 %%% \setsubseheadstyle{\normalfont\bfseries\raggedright}%
4646 \setsubseheadstyle{\normalfont\bfseries\memRTLraggedright}%
4647 % subsubsection
4648 \setbeforesubsubseccskip{-0.667\onelineskip
4649 \@plus -0.25\onelineskip \@minus -0.25\onelineskip}%
4650 \setaftersubsubseccskip{0.333\onelineskip \@plus 0.1\onelineskip}%
4651 %%% \setsubsubseheadstyle{\normalfont\normalsize\itshape\raggedright}%
4652 \setsubsubseheadstyle{\normalfont\normalsize\itshape\memRTLraggedright}%
4653 % paragraph
4654 \setbeforeparaskip{1.0\onelineskip
4655 \@plus 0.5\onelineskip \@minus 0.2\onelineskip}%
4656 \setafterparaskip{-1em}%
4657 \setparaheadstyle{\normalfont\normalsize\itshape\addperiod}%
4658 % subparagraph
4659 \setsubparaindent{\parindent}%

```

```

4660 \setbeforeparaskip{1.0\onelineskip
4661 \@plus 0.5\onelineskip \@minus 0.2\onelineskip}%
4662 \setaftersubparaskip{-1em}%
4663 \setsubparaheadstyle{\normalfont\normalsize\itshape\addperiod}}
4664

```

\hds@bringhurst Head styles based on Bringhurst's book

```

4665 \makeheadstyles{bringhurst}{%
4666 % chapter
4667 \chapterstyle{bringhurst}
4668 % section
4669 \setbeforeseccskip{-1\onelineskip
4670 \@plus -0.5\onelineskip \@minus -.5\onelineskip}%
4671 \setafterseccskip{1\onelineskip \@plus 0.1\onelineskip}%
4672 %%% \setsecheadstyle{\normalfont\raggedright\scshape\MakeLowercase}%
4673 \setsecheadstyle{\normalfont\memRTLraggedright\scshape\MakeLowercase}%
4674 % subsection
4675 \setbeforesubseccskip{-1.0\onelineskip
4676 \@plus -0.25\onelineskip \@minus -0.25\onelineskip}%
4677 \setaftersubseccskip{1.0\onelineskip \@plus 0.1\onelineskip}%
4678 %%% \setsubsecheadstyle{\sethangfrom{\noindent ###1}\normalfont\itshape\raggedright}%
4679 \setsubsecheadstyle{\sethangfrom{\noindent ###1}\normalfont\itshape\memRTLraggedright}%
4680 % subsubsection
4681 \setbeforesubsubseccskip{1.0\onelineskip
4682 \@plus 0.5\onelineskip \@minus 0.2\onelineskip}%
4683 \setaftersubsubseccskip{-1em}%
4684 \setsubsubsecheadstyle{\normalfont\normalsize\scshape\MakeLowercase}%
4685 % paragraph
4686 \setbeforeparaskip{1.0\onelineskip
4687 \@plus 0.5\onelineskip \@minus 0.2\onelineskip}%
4688 \setafterparaskip{-1em}%
4689 \setparaheadstyle{\normalfont\normalsize\itshape\addperiod}%
4690 % subparagraph
4691 \setsubparaindent{\parindent}%
4692 \setbeforesubparaskip{1.0\onelineskip
4693 \@plus 0.5\onelineskip \@minus 0.2\onelineskip}%
4694 \setaftersubparaskip{-1em}%
4695 \setsubparaheadstyle{\normalfont\normalsize\itshape\addperiod}}
4696

```

\hds@crosshead Head styles based on one of Bringhurst's suggestions.

```

4697 \makeheadstyles{crosshead}{%
4698 \chapterstyle{crosshead}
4699 % section
4700 \setbeforeseccskip{-1.25\onelineskip
4701 \@plus -0.5\onelineskip \@minus -.5\onelineskip}%
4702 \setafterseccskip{0.75\onelineskip \@plus 0.1\onelineskip}%
4703 \setsecheadstyle{\normalfont\centering\MakeUppercase}%
4704 % subsection
4705 \setbeforesubseccskip{-1.25\onelineskip

```

```

4706 \@plus -0.25\onelineskip \@minus -0.25\onelineskip}%
4707 \setaftersubsecskip{0.75\onelineskip \@plus 0.1\onelineskip}%
4708 \setsubsecheadstyle{\normalfont\centering\bfseries}%
4709 % subsection
4710 \setbeforesubsubsecskip{-.667\onelineskip
4711 \@plus -0.25\onelineskip \@minus -0.25\onelineskip}%
4712 \setaftersubsubsecskip{.333\onelineskip \@plus 0.1\onelineskip}%
4713 \setsubsubsecheadstyle{\normalfont\normalsize\centering\scshape\MakeLowercase}%
4714 % paragraph
4715 \setbeforeparaskip{-.667\onelineskip
4716 \@plus -0.25\onelineskip \@minus -0.25\onelineskip}%
4717 \setafterparaskip{.333\onelineskip \@plus 0.1\onelineskip}%
4718 \setparaheadstyle{\normalfont\normalsize\centering\itshape}%
4719 % subparagraph
4720 \setsubparaindent{\parindent}%
4721 \setbeforesubparaskip{1.0\onelineskip
4722 \@plus 0.5\onelineskip \@minus 0.2\onelineskip}%
4723 \setaftersubparaskip{-1em}%
4724 \setsubparaheadstyle{\normalfont\normalsize\scshape\MakeLowercase}}
4725

```

\hds@dowding Head styles based on Dowding's *Finer Points*.

```

4726 \makeheadstyles{dowding}{%
4727 % chapter
4728 \chapterstyle{dowding}
4729 % section
4730 \setbeforesecskip{-2\onelineskip
4731 \@plus -0.5\onelineskip \@minus -.5\onelineskip}%
4732 \setaftersecskip{1\onelineskip \@plus 0.1\onelineskip}%
4733 \setsecheadstyle{\normalfont\centering\MakeUppercase}%
4734 % subsection
4735 \setbeforesubsecskip{-1.2\onelineskip
4736 \@plus -0.25\onelineskip \@minus -0.25\onelineskip}%
4737 \setaftersubsecskip{0.8\onelineskip \@plus 0.1\onelineskip}%
4738 \setsubsecheadstyle{\normalfont\scshape\centering\MakeLowercase}%
4739 % subsubsection
4740 \setbeforesubsubsecskip{-0.667\onelineskip
4741 \@plus -0.25\onelineskip \@minus -0.25\onelineskip}%
4742 \setaftersubsubsecskip{0.333\onelineskip \@plus 0.1\onelineskip}%
4743 \setsubsubsecheadstyle{\normalfont\normalsize\centering\itshape}%
4744 % paragraph
4745 \setbeforeparaskip{1.0\onelineskip
4746 \@plus 0.5\onelineskip \@minus 0.2\onelineskip}%
4747 \setafterparaskip{-1em}%
4748 \setparaheadstyle{\normalfont\normalsize\itshape\addperiod}%
4749 % subparagraph
4750 \setsubparaindent{\parindent}%
4751 \setbeforesubparaskip{1.0\onelineskip
4752 \@plus 0.5\onelineskip \@minus 0.2\onelineskip}%
4753 \setaftersubparaskip{-1em}%

```

```

4754 \setsubparaheadstyle{\normalfont\normalsize\itshape\addperiod}}
4755

```

\hds@komalike Head styles based on KOMA script classes (scrbook.cls).

```

4756 \makeheadstyles{komalike}{%
4757 % part
4758 \renewcommand*{\partnamefont}{\huge\sffamily\bfseries}%
4759 \renewcommand*{\partnumfont}{\huge\sffamily\bfseries}%
4760 \renewcommand*{\parttitlefont}{\huge\sffamily\bfseries}%
4761 % chapter
4762 \chapterstyle{komalike}
4763 % section
4764 \setbeforeseccskip{-3.5ex \@plus -1ex \@minus -.2ex}%
4765 \setafterseccskip{2.3ex \@plus .2ex}%
4766 %% \setsechadstyle{\normalfont\Large\sffamily\bfseries\raggedright}%
4767 \setsechadstyle{\normalfont\Large\sffamily\bfseries\memRTLraggedright}%
4768 % subsection
4769 \setbeforesubseccskip{-3.25ex \@plus -1ex \@minus -.2ex}%
4770 \setaftersubseccskip{1.5ex \@plus .2ex}%
4771 %% \setsubsechadstyle{\normalfont\large\sffamily\bfseries\raggedright}%
4772 \setsubsechadstyle{\normalfont\large\sffamily\bfseries\memRTLraggedright}%
4773 % subsubsection
4774 \setbeforesubsubseccskip{-3.25ex \@plus -1ex \@minus -.2ex}%
4775 \setaftersubsubseccskip{1.5ex \@plus .2ex}%
4776 %% \setsubsubsechadstyle{\normalfont\normalsize\sffamily\bfseries\raggedright}%
4777 \setsubsubsechadstyle{\normalfont\normalsize\sffamily\bfseries\memRTLraggedright}%
4778 % paragraph
4779 \setbeforeparaskip{3.25ex \@plus 1ex \@minus .2ex}%
4780 \setafterparaskip{-1em}%
4781 \setparaheadstyle{\normalfont\normalsize\sffamily\bfseries}%
4782 % subparagraph
4783 \setsubparaindent{\parindent}%
4784 \setbeforesubparaskip{3.25ex \@plus 1ex \@minus .2ex}%
4785 \setaftersubparaskip{-1em}%
4786 \setsubparaheadstyle{\normalfont\normalsize\sffamily\bfseries}}
4787

```

\hds@ntgl like Head styles based on the NTG classes (boek.cls).

```

4788 \makeheadstyles{ntgl like}{%
4789 % part
4790 \renewcommand*{\partnamefont}{\Large\bfseries\MakeUppercase}%
4791 \renewcommand*{\partnumfont}{\Large\bfseries}%
4792 \renewcommand*{\parttitlefont}{\Large\MakeUppercase}%
4793 % chapter
4794 \chapterstyle{ntgl like}
4795 % section
4796 \setbeforeseccskip{-2\onelineskip
4797 \@plus -1\onelineskip \@minus -.5\onelineskip}%
4798 \setafterseccskip{0.5\onelineskip}%
4799 \setsechadstyle{\normalfont\large\bfseries}%

```

```

4800 % subsection
4801 \setbeforesecskip{-1\onelineskip
4802 \@plus -.5\onelineskip \@minus -.25\onelineskip}%
4803 \setaftersubsecskip{0.01\onelineskip}%
4804 \setsubsecheadstyle{\normalfont\normalsize\bfseries}%
4805 % subsubsection
4806 \setbeforesubsubsecskip{-1\onelineskip
4807 \@plus -.5\onelineskip \@minus -.25\onelineskip}%
4808 \setaftersubsubsecskip{0.01\onelineskip}%
4809 \setsubsubsecheadstyle{\normalfont\normalsize\slshape}%
4810 % paragraph
4811 \setbeforeparaskip{3.25ex \@plus 1ex \@minus .2ex}%
4812 \setafterparaskip{-1em}%
4813 \setparaheadstyle{\normalfont\normalsize\slshape}%
4814 % subparagraph
4815 \setsubparaindent{\parindent}%
4816 \setbeforesubparaskip{3.25ex \@plus 1ex \@minus .2ex}%
4817 \setaftersubparaskip{-1em}%
4818 \setsubparaheadstyle{\normalfont\normalsize\slshape}}
4819

\hds@tandh Head styles based on Thames & Hudson Typography.
4820 \makeheadstyles{tandh}{%
4821 % chapter
4822 \chapterstyle{tandh}
4823 % section
4824 \setbeforesecskip{-2\onelineskip
4825 \@plus -0.5\onelineskip \@minus -.5\onelineskip}%
4826 \setaftersecskip{1\onelineskip \@plus 0.1\onelineskip}%
4827 %%% \setsecheadstyle{\normalfont\raggedright\MakeUppercase}%
4828 \setsecheadstyle{\normalfont\memRTLraggedright\MakeUppercase}%
4829 % subsection
4830 \setbeforesubsecskip{-1.2\onelineskip
4831 \@plus -0.25\onelineskip \@minus -0.25\onelineskip}%
4832 \setaftersubsecskip{0.8\onelineskip \@plus 0.1\onelineskip}%
4833 %%% \setsubsecheadstyle{\normalfont\Large\itshape\raggedright}%
4834 \setsubsecheadstyle{\normalfont\Large\itshape\memRTLraggedright}%
4835 % subsubsection
4836 \setbeforesubsubsecskip{-0.667\onelineskip
4837 \@plus -0.25\onelineskip \@minus -0.25\onelineskip}%
4838 \setaftersubsubsecskip{0.333\onelineskip \@plus 0.1\onelineskip}%
4839 %%% \setsubsubsecheadstyle{\normalfont\normalsize\bfseries\raggedright}%
4840 \setsubsubsecheadstyle{\normalfont\normalsize\bfseries\memRTLraggedright}%
4841 % paragraph
4842 \setbeforeparaskip{1.0\onelineskip
4843 \@plus 0.5\onelineskip \@minus 0.2\onelineskip}%
4844 \setafterparaskip{-1em}%
4845 \setparaheadstyle{\normalfont\normalsize\itshape\addperiod}%
4846 % subparagraph
4847 \setsubparaindent{\parindent}%

```



```

4848 \setbeforeparaskip{1.0\onelineskip
4849 \@plus 0.5\onelineskip \@minus 0.2\onelineskip}%
4850 \setafterparaskip{-1em}%
4851 \setsubparaheadstyle{\normalfont\normalsize\itshape\addperiod}}
4852

```

`\hds@wilsondob` Head styles based on Adrian Wilson's *The Design of Books*.

```

4853 \makeheadstyles{wilsondob}{%
4854 % chapter
4855 \chapterstyle{wilsondob}
4856 % section
4857 \setbeforeseccskip{-1.333\onelineskip
4858 \@plus -0.5\onelineskip \@minus -.5\onelineskip}%
4859 \setafterseccskip{0.667\onelineskip \@plus 0.1\onelineskip}%
4860 %% \setsecheadstyle{\normalfont\raggedright\MakeUppercase}%
4861 \setsecheadstyle{\normalfont\memRTLraggedright\MakeUppercase}%
4862 % subsection
4863 \setbeforesubseccskip{-0.667\onelineskip
4864 \@plus -0.25\onelineskip \@minus -0.25\onelineskip}%
4865 \setaftersubseccskip{0.333\onelineskip \@plus 0.1\onelineskip}%
4866 %% \setsubsecheadstyle{\normalfont\Large\itshape\raggedright}%
4867 \setsubsecheadstyle{\normalfont\Large\itshape\memRTLraggedright}%
4868 % subsubsection
4869 \setbeforesubsubseccskip{-0.667\onelineskip
4870 \@plus -0.25\onelineskip \@minus -0.25\onelineskip}%
4871 \setaftersubsubseccskip{0.333\onelineskip \@plus 0.1\onelineskip}%
4872 %% \setsubsubsecheadstyle{\normalfont\normalsize\raggedright\scshape\MakeLowercase}%
4873 \setsubsubsecheadstyle{\normalfont\normalsize\memRTLraggedright\scshape\MakeLowercase}%
4874 % paragraph
4875 \setbeforeparaskip{1.0\onelineskip
4876 \@plus 0.5\onelineskip \@minus 0.2\onelineskip}%
4877 \setafterparaskip{-1em}%
4878 \setparaheadstyle{\normalfont\normalsize\itshape\addperiod}%
4879 % subparagraph
4880 \setsubparaindent{\parindent}%
4881 \setbeforesubparaskip{1.0\onelineskip
4882 \@plus 0.5\onelineskip \@minus 0.2\onelineskip}%
4883 \setaftersubparaskip{-1em}%
4884 \setsubparaheadstyle{\normalfont\normalsize\itshape\addperiod}}
4885

```

12.10 Appendices

<pre> \ifanappendix \anappendixtrue \anappendixfalse \appendix </pre>	<p>In the standard book class the <code>\appendix</code> command does the following:</p> <ul style="list-style-type: none"> • Resets the chapter and section counters to zero • Sets <code>\@chapapp</code> to <code>\appendixname</code>. • Redefines <code>\thechapter</code> to produce alphabetic appendix numbers.
---	--

```

4886 \newif\ifanappendix
4887   \anappendixfalse
4888 \newcommand{\appendix}{\par
4889   \setcounter{chapter}{0}%
4890   \setcounter{section}{0}%
4891   \gdef\@chapapp{\appendixname}%
4892   \gdef\thechapter{\@Alph\c@chapter}%
4893   \anappendixtrue}
4894

```

This class provides additional appending capabilities.

`\appendixpage` The command to typeset a page announcing the start of the appendices. It is based on the `\part` definition. The normal version makes an entry in the ToC but the starred version does not.

```

4895 \newcommand{\appendixpage}{%
4896   \@ifstar{\@sapppage}{\@apppage}}

```

`\memapppageinfo` `\memapppageinfo{\appendixpagename}`
`\memapppagestarinfo` `\memapppagestarinfo{\appendixpagename}`

```

4897 \newcommand{\memapppageinfo}[1]{}
4898 \newcommand{\memapppagestarinfo}[1]{}
4899

```

`\@apppage` `\@apppage` typesets an appendix page and makes an entry in the ToC.

```

4900 \def\@apppage{%
4901   \@setuppart
4902   \addappheadtotoc
4903   \partmark{\appendixpagename}%
4904   \memapppageinfo{\appendixpagename}%
4905   {\centering
4906     \interlinepenalty \@M
4907     \normalfont
4908     \printparttitle{\appendixpagename}\par}%
4909   \@endpart}

```

`\@sapppage` `\@sapppage` typesets an appendix page and does not make an entry in the ToC.

```

4910 \def\@sapppage{%
4911   \@setuppart
4912   \partmark{\appendixpagename}%
4913   \memapppagestarinfo{\appendixpagename}%
4914   {\centering
4915     \interlinepenalty \@M
4916     \normalfont
4917     \printparttitle{\appendixpagename}\par}%
4918   \@endpart}
4919

```

`\addappheadtotoc` This command adds an ‘appendices’ line to the ToC. The style is the same as used in `tocbibind` for the ‘List of figures’ line. That is, as a Chapter heading.

```
4920 \def\addappheadtotoc{%
4921   \phantomsection\addcontentsline{toc}{chapter}{\appendixtocname}}
```

`\@ppsavesec` For the `appendices` environment we need to save and restore the main document division number and the appendix number. The `\restoreapp` command is the one for the user.

```
\@ppsavesec
\@ppstoresec
\@ppsaveapp
\restoreapp 4922 \newcounter{@ppsavesec}
4923 \newcounter{@ppsaveapp}
4924 \setcounter{@ppsaveapp}{0}
4925 \newcommand{\@ppsavesec}{%
4926   \setcounter{@ppsavesec}{\value{chapter}}}
4927 \newcommand{\@ppstoresec}{%
4928   \setcounter{chapter}{\value{@ppsavesec}}}
4929 \newcommand{\@ppsaveapp}{%
4930   \setcounter{@ppsaveapp}{\value{chapter}}}
4931 \newcommand{\restoreapp}{%
4932   \setcounter{chapter}{\value{@ppsaveapp}}}
4933
```

`\@resets@pp` Resets the appropriate sectioning counters and names. This does almost exactly what the default `\appendix` command does, except that it saves and restores sectional numbering. It saves the sectional number at the start and restores the appendix number at the end.

```
4934 \newcommand{\@resets@pp}{%
4935   \par
4936   \@ppsavesec
4937   \setcounter{section}{0}%
4938   \setcounter{chapter}{0}%
4939   \renewcommand\@chapapp{\appendixname}%
4940   \renewcommand\thechapter{\@Alph{c}{chapter}}%
4941   \restoreapp
4942 }
4943
```

`appendices` This is the heart of the package. Start it off by doing the resetting done by the `\appendix` command but also save the main counters. At the end of the environment save the appendix number and restore the main counters.

```
4944 \newenvironment{appendices}%
4945   {\@resets@pp\anappendixtrue}%
4946   {\@ppsaveapp\@ppstoresec\anappendixfalse}
4947
```

`\setthesection` The user commands for specifying the numbering style for subappendices.

```
4948 \newcommand{\setthesection}{\thechapter.\Alph{section}}
4949
```

`\@resets@ppsub` Similar to `\@resets@pp` except that it is for use within the `subappendices` environment; as such, it is a bit simpler.

```
4950 \newcommand{\@resets@ppsub}{
4951   \par
4952   \setcounter{section}{0}
4953   \renewcommand{\thesection}{\setthesection}
4954 }
4955
```

`\ifnamesubappendix` Switch for adding an ‘appendix’ name before subappendix numbers.

```
\namesubappendixtrue 4956 \newif\ifnamesubappendix
\namesubappendixfalse 4957   \namesubappendixfalse
\namedsubappendices 4958 \newcommand*{\namedsubappendices}{\namesubappendixtrue}
\namedsubappendices 4959 \newcommand*{\unnamedsubappendices}{\namesubappendixfalse}
4960
```

`subappendices` The environment for subappendices. Start it off by doing the resetting of the `\section` command.

```
4961 \newenvironment{subappendices}{%
4962   \@resets@ppsub
```

Change the definition of `\addappheadtotoc` to give a section entry.

```
4963   \def\addappheadtotoc{\phantomsection
4964     \addcontentsline{toc}{section}{\appendixtocname}}
```

To implement the naming we do cunning things with the `\@secntformat` command.

```
4965   \ifnamesubappendix
4966     \def\sectionname{\appendixname}
4967     \def\@secntformat##1{\@ifundefined{##1name}%
4968       {}{\csname ##1name\endcsname\ }%
4969       \csname the##1\endcsname\quad}
4970   \fi
```

That’s it.

```
4971   }{}
4972
```

`\@formatsecmark@pp` Formats the page header for a redefined `\sectionmark`.

```
4973 \newcommand{\@formatsecmark@pp}[1]{%
4974   \MakeUppercase{\appendixname\space
4975     \ifnum \c@secnumdepth >\z@
4976       \thesection\quad
4977     \fi
4978   #1}}
```

12.11 Appendixpage-like pages

This capability was suggested to me by Lars Madsen on 2004/11/28.

```

\leadpagetoclevel \newleadpage[<page-style>]{<cmdname>}{<title>} creates new macros
\newleadpage      called \cmdname and \cmdname* that when called typeset a page like an
\renewleadpage    Appendixpage, with a title <title> using page style <page-style> (default
                  empty). The plain version adds an entry to the ToC but the starred \cmdname*
                  does not. \renewleadpage changes the definitions.
                  The ToC entry style is set by \leadpagetoclevel (default chapter). To have a
                  part-type entry:
                  \renewcommand*{\leadpagetoclevel}{part}.
                  The \partmark command is used if you need to mark the title.
4979 \newcommand*{\leadpagetoclevel}{chapter}
4980 \newcommand*{\newleadpage}[3][empty]{%
4981   \@namedef{#2}{\@ifstar{\dlfm@msapppage{#1}{#2}{#3}}%
4982                     {\dlfm@mappage{#1}{#2}{#3}}}%
4983 \newcommand*{\renewleadpage}[3][empty]{%
4984   \@namedef{#2}{\@ifstar{\dlfm@msapppage{#1}{#2}{#3}}%
4985                     {\dlfm@mappage{#1}{#2}{#3}}}%
4986

\memleadpageinfo \memleadpageinfo{pstyle}{name}{title}
\memleadpagestarinfo \memleadpagestarinfo{pstyle}{name}{title}
4987 \newcommand{\memleadpageinfo}[3]{}
4988 \newcommand{\memleadpagestarinfo}[3]{}
4989

\dlfm@msapppage Implement the starred and regular versions of \ (re)newleadpage
\dlfm@mappage 4990 \newcommand*{\dlfm@msapppage}[3]{%
4991   \@setuppart
4992   \partmark{#3}%
4993   \memleadpagestarinfo{#1}{#2}{#3}%
4994   {\centering
4995    \interlinepenalty \@M
4996    \normalfont
4997    \printparttitle{#3}\par
4998    \thispagestyle{#1}}%
4999   \dlfm@m@endpart{#1}}
5000 \newcommand*{\dlfm@mappage}[3]{%
5001   \@setuppart
5002   \phantomsection
5003   \addcontentsline{toc}{\leadpagetoclevel}{#3}%
5004   \partmark{#3}%
5005   \memleadpageinfo{#1}{#2}{#3}%
5006   {\centering
5007    \interlinepenalty \@M
5008    \normalfont
5009    \printparttitle{#3}\par
5010    \thispagestyle{#1}}%
5011   \dlfm@m@endpart{#1}}
5012

```

`\dlfm@m@endpart` Finishes off a part-like page.

```

5013 \newcommand*{\dlfm@m@endpart}[1]{%
5014   \if@twoside
5015     \if@openright
5016       \null
5017       \thispagestyle{#1}%
5018       \newpage
5019     \fi
5020   \fi
5021   \if@tempswa
5022     \twocolumn
5023   \fi}
5024
```

12.12 Paragraphs

Much of the code in this section is taken from my *Glisterings* columns [Wil07, Wil08].

`\memorigdb`s `\memorigdb`s saves the original definition of `\` and `\memorigpar` saves the original `\par`. The macro `\atcentercr` provides a user call to `\@centercr`. `\atcentercr` These could come in handy for odd paragraph shapes.

```

5025 \let\memorigdb\
5026 \let\memorigpar\par
5027 \let\atcentercr\@centercr
5028
```

12.12.1 Normal (block) paragraphs

`\flushletright` Sets the paragraphing to L^AT_EX's normal form.

```

5029 \newcommand*{\flushletright}{%
5030   \let\memorigdb\
5031   \leftskip\z@skip
5032   \rightskip\leftskip
5033   \parfillskip\@flushglue
5034   \everypar{}}
5035
```

`\linenottooshort` This declaration `\linenottooshort[<length>]` specifies paragraphs such that the last line is at least about *<length>* long (the default is 4em).

```

5036 \newcommand*{\linenottooshort}[1][4em]{%
5037   \@tempdima=\hsize
5038   \advance\@tempdima -#1
5039   \leftskip\z@skip
5040   \rightskip\leftskip
5041   \parfillskip=\@tempdima \@minus \@tempdima}
5042
```

`\russianpar` Using `\russianpar` instead of `\par` to end a paragraph causes it to be set according to Russian typography, where the last line of a multiline paragraph must be either at least as long as `\parindent` and have at least `\parindent` space at the right, or it must be flushleft and flushright.

```

5043 \newcommand*{\russianpar}{\ifhmode\unskip
5044   \strut\vadjust{}\nobreak
5045   \discretionary{}%
5046   {\hbox{\hskip2\parindent
5047     \vrule depth 273sp width 0sp height \ht\strutbox}}%
5048   {\hbox{\hskip\parindent}}%
5049   \hskip-2\parindent \@minus 2\parindent
5050   \hskip\hsize \@minus \hsize
5051   \kern\z@ \parfillskip\z@
5052   \memorigpar
5053   \ifdim\prevdepth=273sp
5054     \nobreak
5055     \vskip-2\baselineskip
5056     \hbox{\strut}%
5057   \fi\fi}
5058

```

`\lastlineparrule` Using `\lastlinerulefill` instead of `\par` to end a paragraph causes all short lines to be filled at the right by a rule (`\lastlineparrule`) extending to the righthand margin.

```

5059 \newcommand*{\lastlineparrule}{%
5060   \hrule height 0.5ex depth \@tempdimb\relax}
5061 \newcommand*{\lastlinerulefill}{%
5062   \let\\\@centercr
5063   \@tempdimb=-0.5ex \advance\@tempdimb 0.4pt
5064   \unskip\nobreak\space
5065   \leaders\lastlineparrule\hskip\@flushglue
5066   \vadjust{}\parfillskip\z@\memorigpar}}
5067

```

12.12.2 Centered lines

`\centerlastline` This declaration specifies normal paragraphs except that the last line of each is centered.

```

5068 \newcommand*{\centerlastline}{%
5069   %% \leftskip\@flushglue
5070   \memRTLleftskip\@flushglue
5071   %% \rightskip=\z@ plus -1fil
5072   \memRTLrightskip=\z@ plus -1fil
5073   \parfillskip=\z@ plus 2fil}
5074

```

`\leftcenterright` This declaration specifies paragraphs where the first line is flushleft (raggedright), the last is flushright (raggedleft) and all inbetween are centered. Set `\everypar{}` afterwards.

```

5075 \newcommand*{\leftcenterright}{%;
5076   \let\\\break
5077   \parindent\z@
5078   \leftskip\@flushglue
5079   \rightskip\leftskip
5080   \parfillskip \z@ \@plus -1fil
5081   \everypar={\hskip \z@ \@plus -1fil}}
5082

```

`\centerfloat` This is a version of `\centering` that can be used to center a wide float with respect to the text block (normally the left of a wide float is aligned with the left of the textblock). This can only be used for centering something where LaTeX knows the width (e.g., a figure or a table / tabular).

This is a modified version of code by Robin Fairbairns (CTT, *Re: Centering a table: problem with rotating.sty, maybe a strange document class?*, 3 Jan 2009).

```

5083 \newcommand*{\centerfloat}{%
5084   \parindent \z@
5085   \leftskip \z@ \@plus 1fil \@minus \textwidth
5086   \rightskip\leftskip
5087   \parfillskip \z@skip}
5088

```

12.12.3 Ragged

The kernel code for `raggedright` (in `lrmiscen.dtx`):

```

\def\raggedright{%
  \let\\\@centercr\@rightskip\@flushglue \rightskip\@rightskip
  \leftskip\z@skip
  \parindent\z@}
%% \@flushglue = Opt plus 1fil      %% from ltalloc.dtx
%% \z@skip = Opt plus Opt minus Opt %%

```

produces very ragged text with no paragraph indent.

`\ragrparindent` `\raggedyright` [*plus*] provides controllable ragged right paragraphs.

```

\raggedyright 5089 \newdimen\ragrparindent
5090   \setlength{\ragrparindent}{\parindent}
5091 \newcommand{\raggedyright}[1][2em]{%
5092   \let\\\@centercr\@rightskip \z@ \@plus #1\relax
5093   %% \rightskip\@rightskip
5094   \memRTLrightskip\@rightskip
5095   %% \leftskip\z@skip
5096   \memRTLleftskip\z@skip
5097   \parindent\ragrparindent}
5098

```

`\justlastraggedleft` This declaration specifies paragraphs where the lines are justified, except for the last which is `raggedleft` (flushright)..


```

5099 \newcommand*{\justlastraggedleft}{%
5100   %% \leftskip\@flushglue
5101   \memRTLleftskip\@flushglue
5102   %% \rightskip-\leftskip
5103   \memRTLrightskip-\memRTLleftskip
5104   \parfillskip\leftskip
5105   \parindent \z@}
5106

```

`\raggedrightthenleft` This declaration specifies paragraphs where the first line is raggedright (flushleft) and all the rest are raggedleft (flushright). Note that this alters `\everypar`, which may need to be reset afterwards to `\everypar{}`.

```

5107 \newcommand*{\raggedrightthenleft}{%
5108   \parindent \z@
5109   %% \leftskip \z@ \@plus 1fill
5110   \memRTLleftskip \z@ \@plus 1fill
5111   %% \rightskip\@flushglue
5112   \memRTLrightskip\@flushglue
5113   \parfillskip \z@
5114   \everypar{\hspace \z@ \@plus -1fill}}
5115

```

12.12.4 Hanging

`\hangfrom` This is a user-level version of the kernel `\@hangfrom` macro (only the name is changed) as defined in `ltsec.dtx`.
`\hangfrom{<text>}` puts `<text>` in a box and makes a hanging paragraph of the following material (a bit like a description item).

```

5116 \newcommand{\hangfrom}[1]{%
5117   \setbox\@tempboxa\hbox{#1}%
5118   \hangindent \wd\@tempboxa\noindent\box\@tempboxa}
5119

```

`\hangpara` `\hangpara{<indent>}{<afternum>}` at the start of a paragraph will make it hung. If `<indent>` is positive the left will be indented, otherwise the right. If `<afternum>`, say `N`, is positive the `N+1` th lines onwards will be indented. For `N` negative, the first `N` lines will be indented.

```

5120 \newcommand{\hangpara}[2]{\hangindent#1\hangafter#2\noindent}

```

`hangparas` `\begin{hangparas}{<indent>}{<afternum>}` hangs a series of paragraphs.

```

5121 \newenvironment{hangparas}[2]{\setlength{\parindent}{\z@}
5122   \everypar={\hangpara{#1}{#2}}}{\par}
5123

```

12.12.5 Miscellaneous

`\leftspringright` `\leftspringright{<leftfrac>}{<rightfrac>}{<lefttext>}{<righttext>}` sets the `<lefttext>` flushleft (raggedright) in a column `<leftfrac>` of the current textwidth

and the $\langle righttext \rangle$ flushright (raggedleft) in a column $\langle rightfrac \rangle$ of the textwidth, with space $(1.0 - \langle leftfrac \rangle - \langle rightfrac \rangle)$ of the textwidth between them. Both $\langle leftfrac \rangle$ and $\langle rightfrac \rangle$ must be given as decimal numbers (e.g., 0.25 not $1/4$).

```

5124 \newcommand{\leftspringright}[4]{%
5125   \@tempdimb=\hspace
5126   \par\noindent\hbox to\@tempdimb{%
5127     \vtop{\hspace=#1\@tempdimb \flushleft#3\par}\hss
5128     \vtop{\hspace=#2\@tempdimb \flushright#4\par}}
5129
```

`\sourceatright` Putting `\sourceatright[$\langle length \rangle$]{ $\langle text \rangle$ }` at the end of a paragraph will set $\langle text \rangle$ flushright on the same line provided the line is short enough to allow $\langle length \rangle$ (default 2em) between the end of the line and $\langle text \rangle$. If there is not enough space then $\langle text \rangle$ is set flushright on the following line.

```

5130 \newcommand*{\sourceatright}[2][2em]{%
5131   \unskip\nobreak\hfil\penalty50
5132   \hskip#1\hbox{}\nobreak\hfil{#2}
5133   \parfillskip\z@\finalhyphendemerits=0\par}}
5134
```

13 Lists

13.1 General List Parameters

The following commands are used to set the default values for the list environment's parameters. See the L^AT_EX manual for an explanation of the meanings of the parameters. Defaults for the list environment are set as follows. First, `\rightmargin`, `\listparindent` and `\itemindent` are set to 0pt. Then, for a Kth level list, the command `\@listK` is called, where 'K' denotes 'i', 'i', ... , 'vi'. (I.e., `\@listiii` is called for a third-level list.) By convention, `\@listK` should set `\leftmargin` to `\leftmarginK`.

```

\leftmargin For efficiency, level-one list's values are defined at top level, and \@listi is
\leftmargini defined to set only \leftmargin.
\leftmarginii In two column mode the margins should be smaller than in one column
\leftmarginiii typesetting.
\leftmarginiv 5135 \if@twocolumn
\leftmarginv 5136   \setlength{\leftmargini}{2em}
\leftmarginvi 5137 \else
5138   \setlength{\leftmargini}{2.5em}
5139 \fi
```

The value of `\leftmargin` has to be set at this outer level.

```
5140 \leftmargin \leftmargini
```

Lower level list margins are calculated so that they are indented more than the label in an outer list.

```

5141 \setlength{\leftmarginii}{2.2em}
5142 \setlength{\leftmarginiii}{1.87em}
5143 \setlength{\leftmarginiv}{1.7em}
5144 \if@twocolumn
5145   \setlength{\leftmarginv}{.5em}
5146   \setlength{\leftmarginvi}{.5em}
5147 \else
5148   \setlength{\leftmarginv}{1em}
5149   \setlength{\leftmarginvi}{1em}
5150 \fi

```

`\itemindent` Here we set the `\itemindent` which is the extra indentation before a label.

```

5151 \setlength{\itemindent}{\z@}

```

`\labelsep` `\labelsep` is the distance between the label and the text of an item;

`\labelwidth` `\labelwidth` is the width of the label.

```

5152 \setlength{\labelsep}{0.5em}
5153 \setlength{\labelwidth}{\leftmargini}
5154 \addtolength{\labelwidth}{-\labelsep}

```

`\@beginparpenalty` These penalties are inserted before and after a list or paragraph environment.

`\@endparpenalty` They are set to a bonus value to encourage page breaking at these points.

`\@itempenalty` This penalty is inserted between list items.

```

5155 \@beginparpenalty -\@lowpenalty
5156 \@endparpenalty   -\@lowpenalty
5157 \@itempenalty      -\@lowpenalty
5158

```

`\everylistparindent` The kernel sets `\listparindent` to zero within a `\list`, where it can be

`\list` overridden in `\list`'s second argument. Here it is set to `\everyparlistindent` by default, which in turn is 0pt.

```

5159 \newdimen\everylistparindent
5160 \everylistparindent \z@
5161 \renewcommand*{\list}[2]{%
5162   \ifnum \@listdepth > 5 \relax
5163     \@toodeep
5164   \else
5165     \global\advance\@listdepth\@ne
5166   \fi
5167   \rightmargin\z@
5168   \listparindent\everylistparindent
5169   \itemindent\z@
5170   \csname @list\romannumeral\the\@listdepth\endcsname
5171   \def\@itemlabel{#1}%
5172   \let\makelabel\@mklab
5173   \@nmbrowsefalse
5174   #2\relax
5175   \@trivlist

```

```

5176 \parskip\parsep
5177 \parindent\listparindent
5178 \advance\linewidth -\rightmargin
5179 \advance\linewidth -\leftmargin
5180 \advance\@totalleftmargin \leftmargin
5181 \parshape \@ne \@totalleftmargin \linewidth
5182 \ignorespaces}
5183

```

`\parsepi` Lists may be nested and the exact layout depends on the level of nesting. These
`\topsepi` lengths are used to control the nesting-level aspects.

```

\itemsepi 5184 \newlength{\itemsepi}
\parsepii 5185 \newlength{\topsepi}
\topsepii 5186 \newlength{\itemsepi}
\topsepiii 5187 \newlength{\parsepii}
5188 \newlength{\topsepii}
5189 \newlength{\topsepiii}
5190

```

`\itemsepii` We need some new lengths for lists to cater for non-zero `\parskip`.

```

\itemsepiii 5191 \newlength{\itemsepii}
\partopsepii 5192 \newlength{\itemsepiii}
\partopsepii 5193 \newlength{\partopsepii}
5194 \newlength{\partopsepiii}

```

`\setnzsplist` Common code for non-zero `\parskip` in lists.

```

5195 \newcommand*{\setnzsplist}{%
5196 \partopsep \p@ \@plus\z@ \@minus\p@
5197 \topsepi\z@
5198 \parsepi\parskip
5199 \itemsepi\z@
5200 \topsepii\z@
5201 \parsepii\parskip
5202 \itemsepii\z@
5203 \topsepiii\z@
5204 %% \parsepiii\parskip
5205 \itemsepiii\z@}
5206

```

`\defaultlists` The standard L^AT_EX classes have list parameters that give some separation
between lists and `\items` in lists. This macro sets those values. This is a
simplification of memoir's original, and will apply to any font size.

```

5207 \newcommand*{\defaultlists}{%
5208 \setlength{\partopsep}{0.2\onelineskip \@plus 0.1\onelineskip
5209 \@minus 0.1\onelineskip}%
5210 \parsepi = 0.3333\onelineskip \@plus 0.1667\onelineskip \@minus \p@
5211 \itemsepi = \parsepi
5212 \topsepi = 0.6667\onelineskip \@plus 0.3333\onelineskip
5213 \@minus 0.2\onelineskip

```

```

5214 \parsepii = 0.1667\onelineskip \@plus \p@ \@minus \p@
5215 \topsepii = \parsepi
5216 \topsepiii = \parsepii
5217 \everylistparindent \listparindent

```

Additional code to cater for non-zero \parskips.

```

5218 \itemsepii\parsepii
5219 \itemsepiii\topsepiii
5220 \partopsepiii \p@ \@plus\z@ \@minus\p@
5221 \ifm@mnzpskip
5222 \setnzplist
5223 \fi}
5224 \defaultlists
5225

```

`\firmlists` These give approximately half the vertical spacing of the default lists, with all
`\firmlists*` spaces equal. The starred version allows slightly less space before and after the
`\m@msfirmlists` list when it is preceded by a blank line.

```

\m@mfirmmlists 5226 \newcommand*{\firmlists}{%
5227   \@ifstar{\m@msfirmlists}{\m@mfirmmlists}}
5228
5229 \newcommand*{\m@msfirmlists}{
5230   \setlength{\partopsep}{\z@ \@plus \p@ \@minus \p@}%
5231   \parsepi = 0.1667\onelineskip \@plus 0.0833\onelineskip \@minus \p@
5232   \itemsepi = \parsepi
5233   \topsepi = \parsepi
5234   \parsepii = 0.0833\onelineskip \@plus \p@ \@minus \p@
5235   \topsepii = \parsepi
5236   \topsepiii = \parsepii
5237   \everylistparindent\listparindent}
5238
5239 \newcommand*{\m@mfirmmlists}{
5240   \setlength{\partopsep}{0.1\onelineskip \@plus 0.05\onelineskip
5241     \@minus 0.05\onelineskip}%
5242   \parsepi = 0.1667\onelineskip \@plus 0.0833\onelineskip \@minus \p@
5243   \itemsepi = \parsepi
5244   \topsepi = \parsepi
5245   \parsepii = 0.0833\onelineskip \@plus \p@ \@minus \p@
5246   \topsepii = \parsepi
5247   \topsepiii = \parsepii
5248   \everylistparindent\listparindent}
5249

```

`\tightlists` This macro sets the parameters for lists that have less open vertical space in
`\tightlists*` them. I think that these look neater than the defaults. Effectively, no additional
`\m@mtightlists` vertical space is added. The starred version allows slightly no extra space before
`\m@mtightlists` and after the list when it is preceded by a blank line, whereas the unstarred
 version puts half a \onelineskip before *and* after.

```

5250 \newcommand*{\tightlists}{%

```

```

5251 \@ifstar{\m@mstightlists}{\m@mtightlists}}
5252
5253 \newcommand*{\m@mstightlists}{%
5254 \setlength{\partopsep}{\z@ \@plus \p@ \@minus \p@}%
5255 \parsepi = \z@ \@plus \p@ \@minus \p@
5256 \itemsepi = \parsepi
5257 \topsepi = \z@ \@plus \p@ \@minus \p@
5258 \parsepii = \z@ \@plus \p@ \@minus \p@
5259 \topsepii = \parsepi
5260 \topsepiii = \parsepii
5261 \everylistparindent\parindent
5262 \ifm@mnzpskip
5263 \setnzplist
5264 \partopsepiii\partopsep
5265 \fi}
5266
5267 \newcommand*{\m@mtightlists}{%
5268 \setlength{\partopsep}{0.5\onelineskip \@plus \p@ \@minus \p@}%
5269 \parsepi = \z@ \@plus \p@ \@minus \p@
5270 \itemsepi = \parsepi
5271 \topsepi = \z@ \@plus \p@ \@minus \p@
5272 \parsepii = \z@ \@plus \p@ \@minus \p@
5273 \topsepii = \parsepi
5274 \topsepiii = \parsepii
5275 \everylistparindent\parindent
5276 \ifm@mnzpskip
5277 \setnzplist
5278 \partopsepiii\partopsep
5279 \fi}
5280

```

`\firmlist` These two macros can be used at the start of a list environment to reduce the vertical gaps. `\tightlist` removes all interior spaces while `\firmlist` only removes some.

```

5281 \newcommand{\firmlist}{%
5282 \setlength{\itemsep}{0.5\itemsep}\setlength{\parskip}{0.5\parskip}}
5283 \newcommand{\tightlist}{%
5284 \setlength{\itemsep}{0pt}\setlength{\parskip}{0pt}}
5285

```

The space before and after a `trivlist` environment is controlled by the `\topsep` and `\partopsep` skips. There are several environments, such as `center`, that are defined as a `trivlist`.

`\m@msavetopsep` Two skips to store the `\topsep` and `\partopsep` values and a means of setting them and restoring them.

```

\saveptrivseps 5286 \newskip\m@msavetopsep
\restoretrivseps 5287 \newskip\m@msavepartopsep
5288 \newcommand*{\savetrivseps}{%

```

```

5289 \m@msavetopsep\topsep
5290 \m@msavepartopsep\partopsep}
5291 \newcommand*{\restoretrivseps}{%
5292 \topsep\m@msavetopsep
5293 \partopsep\m@msavepartopsep}

```

Save the initial \topsep and \partopsep values.

```

5294 \savetrivseps
5295

```

\zerotrivseps A macro to zero \topsep and \partopsep.

```

5296 \newcommand*{\zerotrivseps}{%
5297 \topsep\z@
5298 \partopsep\z@}
5299

```

\@listI \@listI defines top level and \@listi values of \leftmargin, \parsep, \@listi \topsep, and \itemsep

```

5300 \def\@listi{\leftmargin\leftmarginI
5301 \parsep\parsepi
5302 \topsep\topsepi
5303 \itemsep\itemsepi}
5304 \let\@listI\@listi

```

We should initialise these parameters to the standard defaults

```

5305 \defaultlists
5306 \@listi
5307

```

\@listii Here are the same macros for the lower level lists.

```

\@listiii 5308 \def\@listii{\leftmargin\leftmarginii
\@listiv 5309 \labelwidth\leftmarginii
\@listv 5310 \advance\labelwidth-\labelsep
\@listvi 5311 \topsep\topsepii
5312 \parsep\parsepii
5313 \itemsep\parsepii}
5314
5315 \def\@listiii{\leftmargin\leftmarginiii
5316 \labelwidth\leftmarginiii
5317 \advance\labelwidth-\labelsep
5318 \topsep\topsepiii
5319 \parsep\z@
5320 %%% \itemsep\topsep
5321 %%% \partopsep \p@ \@plus\z@ \@minus\p@
5322 \itemsep\itemsepiii
5323 \partopsep\partopsepiii}
5324
5325 \def\@listiv{\leftmargin\leftmarginiv
5326 \labelwidth\leftmarginiv
5327 \advance\labelwidth-\labelsep}

```

```

5328
5329 \def\@listv{\leftmargin\leftmarginv
5330           \labelwidth\leftmarginv
5331           \advance\labelwidth-\labelsep}
5332
5333 \def\@listvi{\leftmargin\leftmarginvi
5334           \labelwidth\leftmarginvi
5335           \advance\labelwidth-\labelsep}
5336

```

13.2 Enumerate

The enumerate environment uses four counters: *enumi*, *enumii*, *enumiii* and *enumiv*, where *enumN* controls the numbering of the Nth level enumeration.

```

\theenumi  The counters are already defined in latex.dtx, but their representation is
\theenumii changed here.
\theenumiii 5337 \renewcommand{\theenumi}{\@arabic\c@enumi}
\theenumiv 5338 \renewcommand{\theenumii}{\@alph\c@enumii}
           5339 \renewcommand{\theenumiii}{\@roman\c@enumiii}
           5340 \renewcommand{\theenumiv}{\@Alph\c@enumiv}

\labelenumi The label for each item is generated by the commands
\labelenumii \labelenumi ... \labelenumiv.
\labelenumiii 5341 \newcommand{\labelenumi}{\theenumi.}
\labelenumiv 5342 \newcommand{\labelenumii}{\theenumii}
           5343 \newcommand{\labelenumiii}{\theenumiii.}
           5344 \newcommand{\labelenumiv}{\theenumiv.}

\p@enumii  The expansion of \p@enumN\theenumN defines the output of a \ref command
\p@enumiii when referencing an item of the Nth level of an enumerated list.
\p@enumiv 5345 \renewcommand{\p@enumii}{\theenumi}
           5346 \renewcommand{\p@enumiii}{\theenumi(\theenumii)}
           5347 \renewcommand{\p@enumiv}{\p@enumiii\theenumiii}

```

The following is taken directly from David Carlisle's *enumerate* package.
START OF DAVID CARLISLE'S CODE AND COMMENTARY

This package gives the enumerate environment an optional argument which determines the style in which the counter is printed.

An occurrence of one of the tokens A a I i or 1 produces the value of the counter printed with (respectively) \Alph \alph \Roman \roman or \arabic. These letters may be surrounded by any strings involving any other TeX expressions, however the tokens A a I i 1 must be inside a { } group if they are not to be taken as special.

```

\@enlab  Internal token register used to build up the label command from the optional
         argument.

```

```

5348 \newtoks\@enLab

```


`\@enQmark` This just expands to a ‘?’ . `\ref` will produce this, if no counter is printed.

```
5349 \def\@enQmark{?}
```

The next four macros build up the command that will print the item label. They each gobble one token or group from the optional argument, and add corresponding tokens to the register `\@enLab`. They each end with a call to `\@enloop`, which starts the processing of the next token.

`\@enLabel` Add the counter to the label. #2 will be one of the ‘special’ tokens `A a I i 1`, and is thrown away. #1 will be a command like `\Roman`.

```
5350 \def\@enLabel#1#2{%
5351   \edef\@enThe{\noexpand#1{\@enumctr}}%
5352   \@enLab\expandafter{\the\@enLab\csname the\@enumctr\endcsname}%
5353   \@enloop}
```

`\@enSpace` Add a space to the label. The tricky bit is to gobble the space token, as you can
`\@enSpace` not do this with a macro argument.

```
5354 \def\@enSpace{\afterassignment\@enSpace\let\@memtempa= }
5355 \def\@enSpace{\@enLab\expandafter{\the\@enLab\space}\@enloop}
```

`\@enGroup` Add a `{ }` group to the label.

```
5356 \def\@enGroup#1{\@enLab\expandafter{\the\@enLab{#1}}\@enloop}
```

`\@enOther` Add anything else to the label

```
5357 \def\@enOther#1{\@enLab\expandafter{\the\@enLab#1}\@enloop}
```

`\@enloop` The body of the main loop. Eating tokens this way instead of using `\@tfor` lets
`\@enloop@` you see spaces and **all** braces. `\@tfor` would treat `a` and `{a}` as special, but not `{{a}}`.

```
5358 \def\@enloop{\futurelet\@entemp\@enloop@}
5359 \def\@enloop@{%
5360   \ifx A\@entemp      \def\@memtempa{\@enLabel\Alph } \else
5361   \ifx a\@entemp      \def\@memtempa{\@enLabel\alph } \else
5362   \ifx i\@entemp      \def\@memtempa{\@enLabel\roman } \else
5363   \ifx I\@entemp      \def\@memtempa{\@enLabel\Roman } \else
5364   \ifx 1\@entemp      \def\@memtempa{\@enLabel\arabic } \else
5365   \ifx \@sptoken\@entemp \let\@memtempa\@enSpace      \else
5366   \ifx \bgroup\@entemp  \let\@memtempa\@enGroup        \else
5367   \ifx \@enum@\@entemp  \let\@memtempa\@gobble         \else
5368   \let\@memtempa\@enOther
```

Hook for possible extensions

```
5369                                     \@enhook
5370   \fi\fi\fi\fi\fi\fi\fi\fi
```

Process the current token, then look at the next.

```
5371   \@memtempa}
```

`\@enhook` Hook for possible extensions. Some packages may want to extend the number of special characters that are associated with counter representations. This feature was requested to enable Russian alphabetic counting, but here I give an example of a footnote symbol counter, triggered by `*`.
 To enable a new counter type based on a letter, you just need to add a new `\ifx` clause by analogy with the code above. So for example to make `*` trigger footnote symbol counting, a package should do the following.
 Initialise the hook, in case the package is loaded before `enumerate`.

```
\providecommand\@enhook{}
```

Add to the hook a new `\ifx` clause that associates `*` with the `\fnsymbol` counter command.

```
\g@addto@macro\@enhook{%
  \ifx *\@entemp
    \def\@mentempa{\@enLabel\fnsymbol}%
  \fi}
```

This code sequence should work whether it is loaded before or after this `enumerate` package. Any number of new counter types may be added in this way. At this point we just need initialise the hook, taking care not to over write any definitions another package may already have added. (PRW: as this is now in a class, it can be defined instead of `\provided`).

```
5372 %% \providecommand\@enhook{}
5373 \newcommand\@enhook{}
```

`\enumerate` The new `enumerate` environment. This is the first half of the original `enumerate` environment. If there is an optional argument, call `\@@enum@` to define the label commands, otherwise call `\@enum@` which is the second half of the original definition.

```
5374 \def\enumerate{%
5375   \ifnum \@enumdepth >3 \@toodeep\else
5376     \advance\@enumdepth \@ne
5377     \edef\@enumctr{enum\romannumeral\the\@enumdepth}\fi
5378   \@ifnextchar [{\@@enum@}{\@enum@}]}
```

`\@@enum@` Handle the optional argument..

```
5379 \def\@@enum@[#1]{%
```

Initialise the loop which will break apart the optional argument. The command to print the label is built up in `\@enlab`. `\@enThe` will be used to define `\theenum n`.

```
5380 \@enLab{}\let\@enThe\@enQmark
```

The `\@enum@` below is never expanded, it is used to detect the end of the token list.

```
5381 \@enloop#1\@enum@
```

Issue a warning if we did not find one of the ‘special’ tokens.

```
5382 \ifx\@enThe\@enQmark\@warning{The counter will not be printed.%
5383 ~^J\space\@spaces\@spaces\@spaces The label is: \the\@enLab}\fi
```

Define `\labelenumn` and `\theenumn`.

```
5384 \expandafter\edef\csname label\@enumctr\endcsname{\the\@enLab}%
5385 \expandafter\let\csname the\@enumctr\endcsname\@enThe
```

Set the counter to 7 so that we get the width of ‘vii’ if roman numbering is in force then set `\leftmarginn` to the width of the label plus `\labelsep`.

```
5386 \csname c@\@enumctr\endcsname7
5387 \expandafter\settowidth
5388 \csname leftmargin\romannumeral\@enumdepth\endcsname
5389 {\the\@enLab\hspace{\labelsep}}%
```

Finally call `\@enum@` which is the second half of the original definition.

```
5390 \@enum@}
```

`\@enum@` All the list parameters have now been defined, so call `\list`. This is taken straight from the original definition of `\enumerate`.

```
5391 \def\@enum@{\list{\csname label\@enumctr\endcsname}%
5392 {\usecounter{\@enumctr}\def\makelabel##1{\hss\llap{##1}}}}
5393
```

END OF DAVID CARLISLE’S CODE AND COMMENTARY

13.3 Itemize

`\labelitemi` Itemization is controlled by the commands: `\labelitemi`, `\labelitemii`, etc.,
`\labelitemii` which define the labels of the various itemization levels: the symbols used are:
`\labelitemiii` bullet (●), bold en-dash (–), centered asterisk (*), and centered dot (·).

```
\labelitemiii 5394 \newcommand{\labelitemi}{\textbullet}
5395 \newcommand{\labelitemii}{\normalfont\bfseries \textendash}
5396 \newcommand{\labelitemiii}{\textasteriskcentered}
5397 \newcommand{\labelitemiv}{\textperiodcentered}
```

It seems like a reasonable idea to give the `itemize` environment an optional argument to match `enumerate`. Fortunately this seems to be much simpler and I might even be able to work it out for myself.

`itemize` This is a hack at the kernel code for `itemize`.

```
5398 \renewcommand{\itemize}[1][\@empty]{%
5399 \ifnum \@itemdepth > \thr@@\@toodeep\else
5400 \advance\@itemdepth\@ne
5401 \ifx \@empty #1\else % optional argument
5402 \onamedef{labelitem\romannumeral\the\@itemdepth}{#1}%
5403 \fi
5404 \edef\@itemitem{labelitem\romannumeral\the\@itemdepth}%
5405 \expandafter
```

```

5406 \list
5407 \csname\@itemitem\endcsname
5408 {\def\makelabel##1{\hss\llap{##1}}}%
5409 \fi}
5410 \let\enditemize=\endlist
5411

```

13.4 Description

The class defines two description environment, the standard one and a ‘block’ one, and also two semi-configurable versions.

description The description environment is defined here – while the default itemize and enumerate environments are defined in `latex.dtx`.

```

5412 \newenvironment{description}%
5413 {\list{}\labelwidth\z@ \itemindent-\leftmargin
5414 \let\makelabel\descriptionlabel}}%
5415 {\endlist}

```

`\descriptionlabel` To change the formatting of the label, you must redefine `\descriptionlabel`.

```

5416 \newcommand*{\descriptionlabel}[1]{\hspace\labelsep
5417 \normalfont\bfseries #1}

```

blockdescription The ‘block’ description environment.

```

5418 \newenvironment{blockdescription}%
5419 {\list{}\labelwidth\z@ \itemindent 0.5em \labelsep 0.5em
5420 \let\makelabel\blockdescriptionlabel}}%
5421 {\endlist}

```

`\blockdescriptionlabel` In order to change the formatting of the label, you must redefine the macro `\blockdescriptionlabel`.

```

5422 \newcommand*{\blockdescriptionlabel}[1]{%% \hspace\labelsep
5423 \normalfont\bfseries #1}

```

labelled This is a version of the description environment which takes the name, without the backslash, of some `\labelcode` as its argument. For example:

```

\newcommand*{\sflabel}[1]{\hspace\labelsep \normalfont\sffamily #1}
\begin{labelled}{sflabel}
\item[label] using a sans font for the labels

```

```

5424 \newenvironment{labelled}[1]%
5425 {\list{}\labelwidth\z@ \itemindent-\leftmargin
5426 \def\m@malabel{\@nameuse{#1}} \let\makelabel\m@malabel}}%
5427 {\endlist}
5428

```

flexlabelled

`\begin{flexlabelled}{labelcode}{labelwidth}{labelsep}{itemindent}{leftmargin}{rightmargin}` lets you specify some of the parameters for a description-like list. The first argument is as for the labelled environment. The others are all lengths for the various parameters; a * instead of a length means that that argument is to be ignored.

```

5429 \newenvironment{flexlabelled}[6]%
5430   {\list{}\{\nametest{#2}{*}%
5431     \ifsamename\else \labelwidth #2 \fi
5432     \nametest{#3}{*}%
5433     \ifsamename\else \labelsep #3 \fi
5434     \nametest{#4}{*}%
5435     \ifsamename\else \itemindent #4 \fi
5436     \nametest{#5}{*}%
5437     \ifsamename\else \leftmargin #5 \fi
5438     \nametest{#6}{*}%
5439     \ifsamename\else \rightmargin #6 \fi
5440     \def\m@malabel{\@nameuse{#1}} \let\makelabel\m@malabel}}%
5441   {\endlist}
5442

```

13.5 Quotation

quotation The quotation environment is defined by making clever use of the list environment's parameters. The lines in the environment are set smaller than `\textwidth`. The first line of a paragraph inside this environment is indented.

```

5443 \newenvironment{quotation}%
5444   {\list{}\{\listparindent 1.5em%
5445     \itemindent \listparindent
5446     \rightmargin \leftmargin
5447     \parsep \z@ \@plus\p@}%
5448   \item[]}%
5449   {\endlist}

```

13.6 Quote

quote The quote environment is like the quotation environment except that paragraphs are not indented.

```

5450 \newenvironment{quote}%
5451   {\list{}\{\rightmargin\leftmargin}%
5452   \item[]}%
5453   {\endlist}

```

13.7 Theorem

This document class does not define its own theorem environments, the defaults, supplied by `latex.dtx` are available.

13.8 Listing of symbols and abbreviations

Another element is the listing of symbols and abbreviations.

```

symbols
\sympollabel 5454 \newcommand{\sympollabel}[1]{\#1 \hfill}

5455 \newenvironment{symbols}{\list{}}%
5456     {\itemindent 0em \leftmargin 8em
5457     \labelsep 1em \labelwidth 5em
5458     \let\makelabel\sympollabel}}%
5459     {\endlist}

\symboldef Within a symbols environment the command
\symboldef{\symbol}{\langle meaning\rangle} is used to specify and explain each symbol
or abbreviation.
5460 \newcommand{\symboldef}[2]{\item[#1] #2}

```

14 Abstracts

Books usually do not have abstracts, but I decided to add in the code from the `abstract` package [Wil01a].

We just have the `report` or `article` style for the abstract with no `titlepage` option. The original code, from `classes.dtx` for this is:

```

\newenvironment{abstract}{%
  \if@twocolumn
    \section*{\abstractname}
  \else
    \small
    \begin{center}%
      {\bfseries \abstractname\vspace{-.5em}\vspace{\z@}}}%
    \end{center}%
    \quotation
  \fi}%
{\if@twocolumn\else\endquotation\fi}

```

The following `\if...` commands are for implementing various options.

```

\if@bsonecol
\ifadd@bstotoc 5461 \newif\if@bsonecol
\ifnumber@bs 5462 \@bsonecoltrue
\if@bsrunin 5463 \newif\ifadd@bstotoc
5464 \add@bstotocfalse
5465 \newif\ifnumber@bs
5466 \number@bsfalse
5467 \newif\if@bsrunin
5468 \@bsruninfalse
5469

```

`\abstractcol` These are the use-level commands for setting the options. If the abstract is
`\abstractintoc` runin, then it must not be numbered otherwise adding it to the ToC might result
`\abstractnum` in something peculiar.

```
\abstractrunin 5470 \newcommand{\abstractcol}{\@bsonecolfalse}
5471 \newcommand{\abstractintoc}{\add@bstotoctrue}
5472 \newcommand{\abstractnum}{\number@bstrue\@bsruninfalse}
5473 \newcommand{\abstractrunin}{\@bsrunintrue\number@bsfalse}
5474
```

The next set of macros comprise the implementation of the `abstract` environment.

`\abstractnamefont` These two macros are for specifying the fonts for typesetting the abstract's title
`\abstracttextfont` and text. They are initialised for the default case (i.e., no class options).

```
5475 \newcommand{\abstractnamefont}{\normalfont\small\bfseries}
5476 \newcommand{\abstracttextfont}{\normalfont\small}
```

`\abscolnamefont` These two macros are for specifying the fonts for typesetting the abstract's title
`\abscoltextfont` and text in a two column document where the abstract is part of a column.

```
5477 \newcommand{\abscolnamefont}{\normalfont\Large\bfseries}
5478 \newcommand{\abscoltextfont}{\normalfont}
5479
```

`\absnamepos` `\absnamepos` specifies the environment in which the abstract's title name will be
`\abstitlekip` typeset, and the length `\abstitlekip` is an adjustment to the vertical space
between the title and the abstract's text. These are initialised for the default
case.

```
5480 \newcommand{\absnamepos}{center}
5481 \newlength{\abstitlekip} \setlength{\abstitlekip}{-0.5em}
```

`\absleftindent` The abstract's text is typeset as a single item list, called `@bstr@ctlist`. These
`\abs@leftindent` lengths set the left and right margin indents, the paragraph indentation, and the
`\absrightindent` inter-paragraph vertical space. Their initial values are all class option-dependent.

```
\absparindent 5482 \newlength{\absleftindent}
\absparsep 5483 \absleftindent=\leftmargin
5484 \newdimen\abs@leftindent
5485 \abs@leftindent=\leftmargin
5486 \newlength{\absrightindent}
5487 \absrightindent=\leftmargin
5488 \newlength{\absparindent}
5489 \newlength{\absparsep}
5490
```

`\abslabeldelim` The contents of `\abslabeldelim` are typeset after a run-in heading.

```
\@bslabeldelim 5491 \newcommand{\abslabeldelim}[1]{\def\@bslabeldelim{#1}}
5492 \abslabeldelim{}
```

`\@bsrunintitle` This macro typeset the run-in heading.

```
5493 \newcommand{\@bsrunintitle}{%
5494   \hspace*{\abstitlekip}{\abstractnamefont\abstractname\@bslabeldelim}}
5495
```

`\setup@bstract` Now arrange to set all the class option-dependent values.

```
5496 \newcommand{\setup@bstract}{%
5497   \abs@leftindent=\absleftindent
5498   \if@twocolumn
5499     Values for the twocolumn class option.
5500     \if@bsonocol
5501       \abs@leftindent=\z@
5502       \abs@rightindent=\z@
5503       \renewcommand*{\abstractnamefont}{\abscolnamefont}
5504       \renewcommand*{\abstracttextfont}{\abscoltextfont}
5505       \renewcommand*{\absnamepos}{\flushleft}
5506       \setlength{\abstitlekip}{-2ex}
5507     \fi
5508   \fi}
5509
5510 \AtBeginDocument{\setlength{\absparindent}{\parindent}
5511                  \setlength{\absparsep}{\parskip}}
5512
```

`@bstr@ctlist` The abstract's text is typeset within the `@bstr@ctlist` list environment.

```
5513 \newenvironment{@bstr@ctlist}{%
5514   \list{}{%
5515     %%\topsep      \z@
5516     \partopsep    \z@
5517     \listparindent \absparindent
5518     \itemindent   \listparindent
5519     \leftmargin   \abs@leftindent
5520     \rightmargin  \abs@rightindent
5521     \parsep       \absparsep}%
5522   \item\relax}
5523 {\endlist}
5524
```

`\put@bsintoc` This macro adds the abstract's title to the ToC. It does nothing if the abstract is being numbered.

```
5525 \newcommand{\put@bsintoc}{%
5526   \ifadd@bstotoc
5527   \ifnumber@bs\else
5528     \phantomsection
5529     \addcontentsline{toc}{chapter}{\abstractname}
5530   \fi
5531 \fi}
5532
```


`\num@bs` This macro generates a numbered abstract heading.

```
5533 \newcommand{\num@bs}{\chapter{\abstractname}}
5534
```

abstract At last we are in position to define the `abstract` environment. This follows very much along the lines of the standard class definitions, but with macros inserted at strategic points.

The `twocolumn` option and the default case. These use the same code as any style differences are embedded in the new macros.

```
5535 \newenvironment{abstract}{%
5536   \setup@bstract
5537   \if@bsrunin\else
5538     \ifnumber@bs \num@bs \else
5539       \begin{\absnamepos}\abstractnamefont\abstractname\end\absnamepos%
5540       \vspace{\abstitleskip}%
5541     \fi
5542   \fi
5543   \put@bsintoc%
5544   \begin{@bstr@ctlist}\if@bsrunin\@bsrunintitle\fi\abstracttextfont}%
5545   {\par\end{@bstr@ctlist}}
5546
```

onecolabstract An environment for typesetting a single column abstract, particularly as the optional argument to the `\twocolumn` command.

```
5547 \newenvironment{onecolabstract}{%
5548   \begin{@twocolumnfalse}\begin{abstract}}{%
5549   \end{abstract}\end{@twocolumnfalse}}
5550
```

`\thanks` We have to keep the contents of the `\thanks` commands as normally these are emptied by the `\maketitle` command. I do this by extending the definition of the `\thanks` (from `ltsect.dtx`) command, so that `\@bs@thanks` has a copy of the contents of `\@thanks`.

```
5551 \addtoargdef{\thanks}{}{%
5552   \protected@xdef\@bs@thanks{\@bs@thanks
5553     \protect\footnotetext[\the\c@footnote]{#1}}%
5554 }
5555 \let\@bs@thanks\@empty
5556
```

`\saythanks` This macro typesets any `\thanks` commands *after* using `onecolabstract`.

```
5557 \newcommand{\saythanks}{\begingroup
5558   \renewcommand{\thefootnote}{\fnsymbol{footnote}}\@bs@thanks
5559   \endgroup\global\let\@bs@thanks\@empty}
5560
```

15 Verse

The class provides a more flexible `verse` environment than the standard classes. This is based on the `verse` package [Wil01j].

15.1 Environments

Before proceeding with the main, here are some macros for aspects of line numbering.

```

\c@vslineno We need counters for stanza and poem lines. The memfvslineno counter is for
\c@poemline adjusting the starting line for verse line numbers.
\c@modulo@vs 5561 \newcounter{vslineno}
\c@memfvslineno 5562 \newcounter{poemline}
5563 \newcounter{modulo@vs}
5564 \newcounter{memfvslineno}
5565

\poemlines \poemlines{<nth>} specifies that every <nth> line of a poem is to be numbered.
5566 %%%\newcommand{\poemlines}[1]{\linenumberfrequency{#1}%
5567 %%% \@memwarn{Use \string\linenumberfrequency\space
5568 %%% instead of \string\poemlines}}
5569

\linenumberfont Set line numbering font(s).
\vlvnumfont 5570 \newcommand{\linenumberfont}[1]{\def\vlvnumfont{#1}}
5571 %%% \linenumberfont{\small\rmfamily}
5572

\ifbvcountlines Looking ahead, TRUE for (boxed) verbatim line numbers to be printed. Default
\bvcountlinestrue is not to print them. Not that \linenumberfrequency twiddles with this.
\bvcountlinesfalse 5573 \newif\ifbvcountlines% TRUE to print line numbers of (boxed) verbatim lines
5574 \bvcountlinesfalse
5575

\linenumberfrequency Set numbering intervals (number modulo). Default is modulus 0.
\linemodnum 5576 \newcommand{\linenumberfrequency}[1]{%
5577 \ifnum #1< \@ne
5578 \def\linemodnum{0\relax}
5579 \bvcountlinesfalse
5580 \else
5581 \def\linemodnum{#1\relax}
5582 \bvcountlinestrue
5583 \fi}
5584
5585 %%%\linenumberfrequency{0}
5586
```

`\setverselinenums` `\setverselinenums{⟨firstline⟩}{⟨startnumsat⟩}` sets the first line number to `⟨firstline⟩` and the first line number to be printed is `⟨startnumsat⟩`. Use within the verse environment before the first verse. Note that

$$\text{firstline} \leftarrow \text{startnumsat} < \text{firstline} + \text{linenumberfrequency}$$

```

5587 \newcommand*{\setverselinenums}[2]{%
5588   \c@poemline #1\relax \advance\c@poemline \m@ne
5589   \refstepcounter{poemline}%
5590   \ifnum\z@<\linemodnum% we are printing line numbers
5591     \@tempcnta #2\relax
5592     \divide\@tempcnta\linemodnum
5593     \multiply\@tempcnta\linemodnum
5594     \c@memfvline #2\relax
5595     \advance\c@memfvline-\@tempcnta
5596   \fi}
5597
\getthelinenumbers \getthelinenumbers{⟨counter⟩}{⟨start⟩} returns \thecounter if it is exactly
divisible by \linenumberfrequency, provided this is not zero. ⟨start⟩ is the first
number.
5598 \newcommand{\getthelinenumbers}[2]{%
5599   \ifnum\@ne>\linemodnum% no line numbers
5600   \else
5601     \ifnum\@ne=\linemodnum% every line numbered
5602       \@nameuse{the#1}%
5603     \else
5604       \@tempcnta=\@nameuse{c@#1}%
5605       \advance\@tempcnta -\@nameuse{c@#2}%
5606       \divide\@tempcnta \linemodnum
5607       \multiply\@tempcnta \linemodnum
5608       \advance\@tempcnta \@nameuse{c@#2}%
5609       \ifnum\@tempcnta=\@nameuse{c@#1}\@nameuse{the#1}\fi
5610     \fi
5611   \fi}
5612
\ifaltindent This should be set TRUE for indenting alternate lines.
5613 \newif\ifaltindent
5614 \altindentfalse

\ifpattern This should be set TRUE for indenting lines according to a pattern.
5615 \newif\ifpattern
5616 \patternfalse

\ifstarpattern This should be set TRUE for indenting lines according in a patverse*
environment.
5617 \newif\ifstarpattern
5618 \starpatternfalse
5619

```

`\vleftskip` Skips to the left and right of a line of verse.

```

\vrightskip 5620 \newlength{\vleftskip}
              5621 \setlength{\vleftskip}{3em}
              5622 \newlength{\vrightskip}
              5623 \setlength{\vrightskip}{1em}
              5624

```

`\stanzaskip` Skip between stanzas.

```

              5625 \newlength{\stanzaskip}
              5626 \setlength{\stanzaskip}{\onelineskip}
              5627

```

`\flagverse` `\flagverse{<flag>}` inserts *<flag>* at the left (of a line).

```

              5628 \newcommand{\flagverse}[1]{%
              5629 %%% \hskip-\vleftskip\llap{#1}\hskip\vleftskip\ignorespaces}
              5630 \hskip-\memRTL\vleftskip\llap{#1}\hskip\memRTL\vleftskip\ignorespaces}
              5631

```

`\versewidth` The length `\versewidth` is a convenience length for the user.

```

              5632 \newlength{\versewidth}

```

`\vgap` The length `\vgap` is used as the basis for spacing. `\vin` makes a horizontal space
`\vin` of `\vgap` and `\vindent` is the indentation of wrapped lines.

```

\vindent 5633 \newlength{\vgap} \setlength{\vgap}{1.5em}
              5634 \newcommand{\vin}{\hspace*{\vgap}}
              5635 \newlength{\vindent} \setlength{\vindent}{2\vgap}

```

`\vinphantom` Macro to leave blank space corresponding to a string.

```

              5636 \newcommand{\vinphantom}[1]{\leavevmode\phantom{#1}}

```

`\vleftofline` Macro to insert something immediately to the left of the start of the line.

```

              5637 \newcommand*\vleftofline[1]{\leavevmode\llap{#1}}

```

`\vleftmargin` Length to adjust the default left margin within a verse environment.

```

              5638 \newdimen\vleftmargin
              5639 \vleftmargin=\leftmargini
              5640

```

`\verselinebreak` Break a verse line by inserting `\newline`.

```

              5641 \newcommand{\verselinebreak}[1][\z@]{\newline\hspace*{#1}\ignorespaces}
              5642

```

`\incr@vslin` Increment the line counters.

```

              5643 \newcommand{\incr@vslin}{%
              5644 \refstepcounter{poemline}%
              5645 \stepcounter{vslineno}}
              5646

```

`\@vsifbang` Like the kernel `\@ifstar` except it looks for an exclamation mark!

```
5647 \newcommand{\@vsifbang}[1]{\@ifnextchar !{\@firstoftwo{#1}}}
```

`\@vsifgt` Like the kernel `\@ifstar` except it looks for a > character.

```
5648 \newcommand{\@vsifgt}[1]{\@ifnextchar >{\@firstoftwo{#1}}}
```

```
5649
```

`\verselinenumberright` Declarations for setting line numbers at the right (the default) or the left..

```
\verselinenumberright 5650 \newcommand*{\verselinenumberright}{\def\@vstypelinenum{\@vslnumright}}
```

```
\@vstypelinenum 5651 \newcommand*{\verselinenumberright}{\def\@vstypelinenum{\@vslnumleft}}
```

```
5652 \verselinenumberright
```

```
5653
```

`\@vslnumright` Internal code for right/left line numbers.

```
\@vslnumleft 5654 \newcommand*{\@vslnumright}{%
```

```
5655 \hfill\rlap{%\kern\vrightskip\kern\rightmargin%
```

```
5656 \kern\memRTLvrightskip\kern\rightmargin%
```

```
5657 \vlnumfont\getthelinenum{poemline}{memfvslne}}}
```

```
5658 \newcommand*{\@vslnumleft}{%
```

```
5659 \hfill\rlap{%\kern-\textwidth\kern-\vrightskip%
```

```
5660 \kern-\textwidth\kern-\memRTLvrightskip%
```

```
5661 \vlnumfont\getthelinenum{poemline}{memfvslne}}}
```

`\@vscentercr` This puts the poem line number in the margin, increments the line numbers, and then deals with the options. It is based on the kernel `\@centercr`. This has to handle various forms of the `\\` command: `\\`, `*`, `\\!`, and `\\>`, together with an optional length argument.

```
5662 \newcommand{\@vscentercr}{%
```

```
5663 \ifhmode \unskip\else \@nolnerr\fi
```

```
5664 \@vstypelinenum%
```

For > call `\verselinebreak` to process it.

```
5665 \@vsifgt{\verselinebreak}{%
```

```
5666 \incr@vslne
```

If the call is `*...` call `\@vsxcentercr` to handle the `*...`. If the call is `\\!`, do nothing. If the call is `\\![...]`, call `\@vsicentercr` to handle the `[...]`.

Otherwise, call `\@vsxcentercr`.

```
5667 \par\@ifstar{\nobreak\@vsxcentercr}{%
```

```
5668 \@vsifbang{\@ifnextchar[ {\@vsicentercr}{}}{\@vsxcentercr}}}
```

`\@vsxcentercr` Processes `*`, and either calls `cs@vsicentercr` to handle a `[length]`, or `\start@vslne`.

```
5669 \newcommand{\@vsxcentercr}{\addvspace{-\parskip}%
```

```
5670 \@ifnextchar[ {\@vsicentercr}{\start@vslne}}
```

`\@vsicentercr` Processes `(\\...)[length]` and then calls `\start@vslne`.

```
5671 \def\@vsicentercr[#1]{\vskip #1\ignorespaces \start@vslne}
```

`\start@vsline` This is called at the start of every verse line except the first.

```
5672 \newcommand{\start@vsline}{%
5673   \ifaltindent\ifodd\c@vslineno\else\vin\fi\fi%
5674   \ifpattern\get@vsindent\fi%
5675   \ifstarpattern\getstar@vsindent\fi}
5676
```

`\theHpoemline` For the `hyperref` package need a way of distinguishing lines of a poem. See the thread *PDFTEX/Hyperef hates memoir verse environment?* on CTT October 2002.

```
5677 \newcounter{verse}
5678 \setcounter{verse}{0}
5679 \newcommand{\theHpoemline}{\theverse.\thepoemline}
5680
```

`verse` The extended `verse` environment. It sets the verse line counter, then defines the particular list environment adjusting the margins to center according to the length parameter. If the length parameter is at least the `\linewidth` then the ‘centering’ defaults to the original `verse` layout.

```
5681 \newenvironment{verse}[1][\linewidth]{%
5682   \refstepcounter{verse}%
5683   \setcounter{poemline}{0}\refstepcounter{poemline}%
5684   \setcounter{vslineno}{1}%
5685   \let\@vscentercr
5686   \list{}{\itemsep      \z@
5687             \itemindent  -\vindent
5688             \listparindent\itemindent
5689             \leftmargin  \vleftmargin
5690             \parsep      \stanzaskip
5691             \ifdim #1<\linewidth%   %% short line
5692               \rightmargin \z@
5693               \leftmargin  \linewidth
5694               \advance\leftmargin -#1\relax
5695               \advance\leftmargin -0.5\leftmargin
5696               \advance\leftmargin \vindent
5697             \else
5698               \ifdim #1>\linewidth%   %% long line
5699                 \rightmargin \z@
5700                 \leftmargin  \vindent
5701               \else%                   %% default
5702                 \rightmargin \leftmargin
5703                 \advance\leftmargin \vindent
5704               \fi
5705             \fi}
5706   \item[]{\endlist}
5707
```

`altverse` This sets `\altindenttrue` (afterwards false) and initialises the line counter.

```
5708 \newenvironment{altverse}{%
```

```

5709 {\starpatternfalse\patternfalse\altindenttrue
5710 \setcounter{vslineno}{1}}%
5711 {\altindentfalse}
5712

```

15.2 Patterns

The pattern code is based on the idea of converting a string of digits to an array of digits, and then being able to access the digit at a particular position in the array.

\ifbounderror A flag set TRUE if an attempt is made to access an array element outside the array limits.

```

5713 \newif\ifbounderror
5714 \bounderrorfalse

```

\ifinteger A flag to indicate if a ‘number’ is an integer (TRUE) or not (FALSE).

```

5715 \newif\ifinteger
5716

```

\c@chrsinstr A counter for the number of characters.

```

5717 \newcounter{chrsinstr} % CHARactersINSTRing
5718

```

\newarray `\newarray{<arrayname>}{<low>}{<high>}` defines an array called `<arrayname>` (no backslash e.g. `MyArray`), with low and high limits `<low>` and `<high>`.

```

5719 \newcommand{\newarray}[3]{%
5720 \nameedef{#1-low}{#2}%
5721 \nameedef{#1-high}{#3}%
5722 \ifnum #3<#2
5723 \memerror{Limits for array #1 are in reverse order}{\@ehc}%
5724 \fi}
5725

```

\stringtoarray `\stringtoarray{<arrayname>}{<string>}` puts each character from `<string>` sequentially into the `<arrayname>` array, starting with `<low> = 1`. It checks for an empty `<string>` and handles that specially.

```

5726 \newcommand{\stringtoarray}[2]{%
5727 \def\vsarrayname{#1}%
5728 \protected@edef\thevsstring{#2}%
5729 \newarray{\vsarrayname}{1}{1}%
5730 \ifmtarg{#2}{%
5731 \c@chrsinstr \z@
5732 \namedef{\vsarrayname-1}{}}
5733 }{%
5734 \c@chrsinstr \@ne
5735 \expandafter\vsstringtoarray \thevsstring\vsend
5736 }}
5737

```

`\@vsstringtoarray` Recursively adds characters to the array `\@vsarrayname`, incrementing the array's high limit.

```

5738 \def\@vsstringtoarray #1#2\@vsend{%
5739   \@namedef{\@vsarrayname-\the\c@chrsinstr}{#1}
5740   \@nameedef{\@vsarrayname-high}{\the\c@chrsinstr}
5741   \@ifmtarg{#2}{%
5742     \def\@vsinext{%
5743     }{%
5744       \advance\c@chrsinstr \@ne
5745       \def\@vsinext{%
5746         \@vsstringtoarray #2\@vsend%
5747       }%
5748     }
5749   \@vsinext}
5750
```

`\setarrayelement` `\setarrayelement{<arrayname>}{<index>}{<value>}` sets the `<arrayname>` array's element at `<index>` to `<value>`.

```

5751 \newcommand{\setarrayelement}[3]{%
5752   \checkarrayindex{#1}{#2}%
5753   \@nameedef{#1-#2}{#3}}

```

`\getarrayelement` `\getarrayelement{<arrayname>}{<index>}{<value>}` defines the parameterless macro `<value>` (e.g., `\result`) to be the value at `<index>` in the `<arrayname>` array.

```

5754 \newcommand{\getarrayelement}[3]{%
5755   \checkarrayindex{#1}{#2}%
5756   \protected@edef#3{\@nameuse{#1-#2}}
5757
```

`\checkarrayindex` `\checkarrayindex{<arrayname>}{<index>}` checks that the `<index>` of the `<arrayname>` array is valid. `\ifbounderror` is set FALSE if everything is OK, otherwise it is set TRUE.

```

5758 \newcommand{\checkarrayindex}[2]{%
5759   \bounderrorfalse
5760   \expandafter\ifx\csname #1-low\endcsname\relax%
5761     \ifpattern\else
5762       \@memerror{No array called #1}{\@ehc}%
5763     \fi
5764     \bounderrortrue
5765   \fi
5766   \ifnum #2<\@nameuse{#1-low}\relax%
5767     \ifpattern\else
5768       \@memerror{Index #2 outside limits for array #1}{\@ehc}%
5769     \fi
5770     \bounderrortrue
5771   \fi
5772   \ifnum #2>\@nameuse{#1-high}\relax%
5773     \ifpattern\else

```



```

5774      \@memerror{Index #2 outside limits for array #1}{\@ehc}%
5775      \fi
5776      \bounderrortrue
5777      \fi}
5778

```

`\arraytostring` `\arraytostring{⟨arrayname⟩}{⟨string⟩}` converts the characters in the `⟨arrayname⟩` array into the parameterless macro `⟨string⟩` (e.g., `\MyString`).

```

5779 \newcommand{\arraytostring}[2]{%
5780   \def#2{}%
5781   \c@chrsinstr = \@nameuse{#1-low}%
5782   \@vsarraytostring{#1}{#2}}
5783

```

`\vsarraytostring` `\@vsarraytostring{⟨arrayname⟩}{⟨string⟩}` recursively adds the (character) elements from `⟨arrayname⟩` to `⟨string⟩`.

```

5784 \newcommand{\@vsarraytostring}[2]{%
5785   \ifnum\c@chrsinstr>\@nameuse{#1-high}\else
5786     \protected@edef#2{#2\@nameuse{#1-\thechrsinstr}}%
5787     \advance\c@chrsinstr\@ne%
5788     \@vsarraytostring{#1}{#2}%
5789   \fi}
5790

```

`\checkifinteger` `\checkifinteger{⟨num⟩}` checks if `⟨num⟩` is an integer. If it is, then `\ifinteger` is set TRUE, otherwise it is set FALSE. (Code based on Donald Arseneau's `cite` package).

```

5791 \newcommand{\checkifinteger}[1]{%
5792   \protected@edef\@vsa{#1}%
5793   \ifcat _\ifnum9<1\gobm{#1} _\else A\fi
5794   \integertrue%
5795   \else
5796     \integerfalse%
5797   \fi}

```

`\gobm` `\gobm{⟨num⟩}` is defined as `⟨num⟩`. It could be defined as:

```

\newcommand{\gobm}[1]{\ifx-#1\expandafter\gobm\else#1\fi}

```

which would remove a leading minus sign (hyphen) from its argument (`gobm` = gobble minus sign). (Code from a posting to CTT by Donald Arseneau on 1997/07/21).

```

5798 \newcommand{\gobm}[1]{#1}
5799

```

`\indentpattern` `\indentpattern{⟨digits⟩}` stores `⟨digits⟩` for use as a verse indentation pattern.

```

5800 \newcommand{\indentpattern}[1]{%
5801   \stringtoarray{Array@vs}{#1}}

```

`\get@vsindent` `\get@vsindent` gets the indent pattern digit for the `\thevslineno`, then uses this to specify the line indentation as `digit*\vgap`.

```

5802 \newcommand{\get@vsindent}{%
5803   \getarrayelement{Array@vs}{\number\value{vslineno}}{\@vspat}%
5804   \ifbounderror
5805     \arraytostring{Array@vs}{\@vsp@t}%
5806     \@memwarn{%
5807       Index ‘\thevslineno’ for pattern ‘\@vsp@t’ is out of bounds}%
5808     \def\@vspat{0}%
5809   \else
5810     \checkifinteger{\@vspat}%
5811     \ifinteger\else
5812       \arraytostring{Array@vs}{\@vsp@t}%
5813       \@memwarn{%
5814         ‘\@vspat’ at index ‘\thevslineno’ in pattern ‘\@vsp@t’
5815         is not a digit}%
5816       \def\@vspat{0}%
5817     \fi
5818   \fi
5819   \ifcase\@vspat\else\hspace*{\@vspat\vgap}\fi}

```

`\getstar@vsindent` `\getstar@vsindent` gets the indent pattern digit for the `patverse*` environment, then uses this to specify the line indentation as `digit*\vgap`. It lets the pattern repeat by resetting the `vslineno` counter.

```

5820 \newcommand{\getstar@vsindent}{%
5821   \expandafter\ifx\csname Array@vs-high\endcsname\relax
5822   \@memerror{A pattern has not been specified}{\@ehc}
5823   \else
5824     \ifnum\c@vslineno>\@nameuse{Array@vs-high}%
5825     \setcounter{vslineno}{1}%
5826     \fi
5827     \get@vsindent
5828   \fi}
5829

```

patverse The environment for setting verse line indents according to a pattern. It starts by setting `\ifpattern` TRUE, any other flags to FALSE, and initialises the line number. It ends by setting `\ifpattern` FALSE.

```

5830 \newenvironment{patverse}%
5831   {\starpatternfalse\patterntrue\altindentfalse
5832    \setcounter{vslineno}{1}}%
5833   {\patternfalse}
5834

```

patverse* The environment for setting verse line indents according to a repeating pattern. It starts by setting `\ifstarpattern` TRUE, any other flags to FALSE, and initialises the line number. It ends by setting `\ifstarpattern` FALSE.

```

5835 \newenvironment{patverse*}%
5836   {\starpatterntrue\patternfalse\altindentfalse

```

```

5837 \setcounter{vslineno}{1}}%
5838 {\starpatternfalse}
5839

```

15.3 Titles

`\poemtitle` Typeset a poem title (like `\section` or other). The actual work is done by `\@vsptitle` (plain) or `\@vssptitle` (starred).

```

5840 \newcommand{\poemtitle}{\par%
5841 \secdef\@vsptitle\@vssptitle}

```

`\poemtoc` The kind of entry `\poemtitle` is to make in the ToC.

```

5842 \newcommand{\poemtoc}{section}
5843

```

```

\mempoeminfo \mempoeminfo{title}
\mempoemstarinfo \mempoemstarinfo{title}

5844 \newcommand{\mempoeminfo}[1]{}
5845 \newcommand{\mempoemstarinfo}[1]{}
5846

```

`\@vsptitle` Typeset a `\poemtitle`.

```

5847 \long\def\@vsptitle[#1]#2{%
5848 \phantomsection
5849 \addcontentsline{toc}{\poemtoc}{#1}%
5850 \M@gettitle{#1}%
5851 \mempoeminfo{#1}%
5852 \poemtitlemark{#1}%
5853 \@vstypeptitle{#2}%
5854 \@afterheading}
5855

```

`\@vssptitle` Typeset a `\poemtitle*`.

```

5856 \long\def\@vssptitle#1{%
5857 \M@gettitle{#1}%
5858 \mempoemstarinfo{#1}%
5859 \@vstypeptitle{#1}%
5860 \@afterheading}
5861

```

`\@vstypeptitle` This *really* typesets the title.

```

5862 \newcommand{\@vstypeptitle}[1]{%
5863 \vspace{\beforepoemtitleskip}%
5864 {\poemtitlefont #1\par}%
5865 \vspace{\afterpoemtitleskip}%
5866 }
5867

```

```

\poemtitlefont Sets the appearance to the title of a poem, and something for a header.
\poemtitlemark 5868 \newcommand{\poemtitlefont}{\normalfont\large\bfseries\centering}
5869 \newcommand{\poemtitlemark}[1]{%
5870
\beforepoemtitleskip Lengths before and after a poem title, using approximately \section values.
\afterpoemtitleskip 5871 \newlength{\beforepoemtitleskip}
5872 \setlength{\beforepoemtitleskip}{3.5ex \@plus 1ex \@minus .2ex}
5873 \newlength{\afterpoemtitleskip}
5874 \setlength{\afterpoemtitleskip}{2.3ex \@plus .2ex}
5875

\if@numptitle Following the \NumberPoemTitle declaration \PoemTitles are numbered. The
\@numptitletrue \PlainPoemTitle declaration produces unnumbered titles.
\@numptitlefalse 5876 \newif\if@numptitle
\NumberPoemTitle 5877 \newcommand*\NumberPoemTitle{\@numptitletrue}
\PlainPoemTitle 5878 \newcommand*\PlainPoemTitle{\@numptitlefalse}
5879 \NumberPoemTitle
5880

\c@poem Counter for poem titles.
\thepoem 5881 \newcounter{poem}\setcounter{poem}{0}
\theHpoem 5882 \renewcommand*\thepoem{\@arabic\c@poem}
5883 \newcommand*\theHpoem{\arabic{poem}}
5884

\poemtitlestarmark \poemtitlemark{\poemf@rhdr} is used for marks for a \PoemTitle and
\poemtitlepstyle \poemtitlestarmark{\poemf@hdr} is for marks for \PoemTitle*. The
\poemtitlestarpstyle \poemtitlepstyle and \poemtitlestarpstyle macros are provided as hooks so
that, for example, pagestyles can be set for the regular and starred versions of
\PoemTitle. By default these macros do nothing.
5885 \newcommand*\poemtitlestarmark[1]{%
5886 \newcommand*\poemtitlepstyle{}
5887 \newcommand*\poemtitlestarpstyle{}
5888

\PoemTitle The command to produce a article style chapter-like (numbered) poem title.
5889 \newcommand\PoemTitle{%
5890 \par
5891 \@afterindentfalse
5892 \@ifstar{\@m@msPoemTitle}{\@m@mPoemTitle}}
5893

\@m@mPoemTitle Intermediate and support macros for the extra optional argument to
\poemt@c \PoemTitle. Have to do this long windedly otherwise dear old hyperref barfs.
5894 \newcommand{\@m@mPoemTitle}[1][{}]{%
5895 \def\poemt@c{#1}% capture first optional arg
5896 \ifnextchar[{\@PoemTitle}{\@PoemTitle[]}%
5897 }
5898

```

```

\memPoemTitleinfo \memPoemTitleinfo{num}{toc}{head}{full}
\memPoemTitlestarinfo \memPoemTitlestarinfo{short}{full}
5899 \newcommand{\memPoemTitleinfo}[4]{}
5900 \newcommand{\memPoemTitlestarinfo}[2]{}
5901

\@PoemTitle \@PoemTitle[(tocmark)]{<title>} typesets the title of a \PoemTitle. There is a
\poemf@rtoc number for \@numptitletrue and \@mainmattertrue.
\poemf@rhdr 5902 \def\@PoemTitle[#1]#2{%
5903   \phantomsection
5904   \ifx\poemt@c\@empty % no optional args
5905     \def\poemf@rtoc{#2}%
5906     \def\poemf@rhdr{#2}%
5907   \else % at least one opt arg
5908     \let\poemf@rtoc\poemt@c
5909     \ifx\@empty#1\@empty
5910       \let\poemf@rhdr\poemt@c
5911     \else
5912       \def\poemf@rhdr{#1}%
5913     \fi
5914   \fi
5915   \m@m@Andfalse
5916   \if@numptitle
5917     \if@mainmatter
5918       \m@m@Andtrue
5919     \fi
5920   \fi
5921   \ifm@m@And
5922     \refstepcounter{poem}%
5923   \fi

  Store the (short) title via \poemtitlemark and call \makePoemTitlehead to do
  the typesetting.
5924   \@makePoemTitlehead{#2}%
5925   \@afterheading
5926   \poemtitlemark{\poemf@rhdr}%
5927   \poemtitlepstyle

  Add the title to the ToC.
5928   \ifm@m@And
5929     \addcontentsline{toc}{\poemtoc}{%
5930       \protect\numberline{\thepoem}\poemf@rtoc}%
5931     \memPoemTitleinfo{\thepoem}{\poemf@rtoc}{\poemf@rhdr}{#2}%
5932   \else
5933     \addcontentsline{toc}{\poemtoc}{\poemf@rtoc}%
5934     \memPoemTitleinfo{}{\poemf@rtoc}{\poemf@rhdr}{#2}%
5935   \fi

  Add hook for title referencing.
5936   \ifheadnameref\M@getttitle{\poemf@rhdr}\else\M@getttitle{\poemf@rtoc}\fi}
5937

```

```

\@makePoemTitlehead This really typesets a \PoemTitle. Leave some whitespace.
5938 \def\@makePoemTitlehead#1{%
5939   \PoemTitleheadstart
5940   \parindent \z@ \normalfont
      If there is a number, typeset it, otherwise call \printPoemTitlenonum.
5941   \ifm@m@And
5942     \printPoemTitlenum
5943     \afterPoemTitlenum
5944   \else
5945     \printPoemTitlenonum
5946   \fi
      Typeset the title.
5947   \interlinepenalty\@M
5948   \printPoemTitledtitle{#1}%
5949   \afterPoemTitle}}
5950

\@PTchs@def@ult This sets up all the definitions used in \@makePoemTitlehead and
\@makesPoemTitlehead.

\PoemTitleheadstart
\printPoemTitlenum 5951 \newcommand{\@PTchs@def@ult}{%
\afterPoemTitlenum 5952   \def\PoemTitleheadstart{\vspace{\beforePoemTitleskip}}
\printPoemTitlenonum 5953   \def\printPoemTitlenum{\PoemTitlenumfont \thepoem}
\printPoemTitledtitle 5954   \def\afterPoemTitlenum{\par\nobreak\vskip \midPoemTitleskip}
\afterPoemTitle 5955   \def\printPoemTitlenonum{}
5956   \def\printPoemTitledtitle##1{\PoemTitlefont ##1}
5957   \def\afterPoemTitle{\par\nobreak\vskip \afterPoemTitleskip}}
5958 \@PTchs@def@ult
5959

\PoemTitlenumfont Fonts for setting the Poem Title number and title.
\PoemTitlefont 5960 \newcommand*\PoemTitlenumfont{\normalfont\large\centering}
5961 \newcommand*\PoemTitlefont{\normalfont\large\centering}

\beforePoemTitleskip Lengths separating the various parts of a Poem Title heading.
\midPoemTitleskip 5962 \newlength{\beforePoemTitleskip}
\afterPoemTitleskip 5963 \setlength{\beforePoemTitleskip}{1\onelineskip}
5964 \newlength{\midPoemTitleskip}
5965 \setlength{\midPoemTitleskip}{0pt}
5966 \newlength{\afterPoemTitleskip}
5967 \setlength{\afterPoemTitleskip}{1\onelineskip}
5968

\@m@msPoemTitle This deals with the optional argument for starred PoemTitles.
5969 \newcommand{\@m@msPoemTitle}[2][\@empty]{%
5970   \@sPoemTitle{#2}%
5971   \ifx \@empty#1

```

```

5972 \def\poemf@rhdr{#2}%
5973 \else % opt arg
5974 \def\poemf@rhdr{#1}%
5975 \fi
5976 \poemtitlestarmark{\poemf@rhdr}%
5977 \poemtitlestarpstyle
5978 \memPoemTitlestarinfo{\poemf@rhdr}{#2}}
5979

```

`\@sPoemTitle` `\@sPoemTitle{<long>}` typesets the title of a `\PoemTitle*`. It is easier than the `\@PoemTitle` as there is no number or ToC entry to worry about.

```

5980 \newcommand{\@sPoemTitle}[1]{%
5981 \@makesPoemTitlehead{#1}%
5982 \@afterheading
5983 \M@gettitle{#1}}
5984

```

`\@makesPoemTitlehead` This *really* typesets a `\PoemTitle*`, and is similar to `\@makePoemTitlehead`.

```

5985 \def\@makesPoemTitlehead#1{%
5986 \PoemTitleheadstart
5987 \parindent \z@ \normalfont
5988 \printPoemTitlenonum
5989 \interlinepenalty\@M
5990 \printPoemTitletitle{#1}
5991 \afterPoemTitle}}
5992

```

16 Setting parameters for existing environments

16.1 Array and tabular

`\arraycolsep` The columns in an array environment are separated by `2\arraycolsep`.

```

5993 \setlength\arraycolsep{5\p@}

```

`\tabcolsep` The columns in an tabular environment are separated by `2\tabcolsep`.

```

5994 \setlength\tabcolsep{6\p@}

```

`\arrayrulewidth` The width of rules in the array and tabular environments is given by `\arrayrulewidth`.

```

5995 \setlength\arrayrulewidth{.4\p@}

```

`\doublerulesep` The space between adjacent rules in the array and tabular environments is given by `\doublerulesep`.

```

5996 \setlength\doublerulesep{2\p@}

```

16.2 Tabbing

`\tabbingsep` This controls the space that the `\'` command puts in. (See L^AT_EX manual for an explanation.)

```
5997 \setlength\tabbingsep{\labelsep}
5998
```

16.3 Minipage

`\@minipagerestore` The macro `\@minipagerestore` is called upon entry to a minipage environment to set up things that are to be handled differently inside a minipage environment. In the standard styles it does nothing, as was the case originally for memoir. Memoir extends it to support verbatim footnotes (see later) in minipages and also for abnormal parskip (see later).

```
5999 \newcommand{\@minipagerestore}{%
    For \verbfootnote to work in a minipage we have to use
    \verbmpfootnotetext instead of \@verbfootnotetext.
6000 \let\@verbfootnotetext\verbmpfootnotetext
    The next is for enabling the extended footnotes in a minipage.
6001 \m@doextrafeetmini
    And this is for abnormal parskip.
6002 \ifm@mznzpskip \parskip=\m@mabparskip\fi}
6003
```

`\@mpfootins` Minipages have their own footnotes; `\skip\@mpfootins` plays the same rôle for footnotes in a minipage as `\skip\footins` does for ordinary footnotes.

```
6004 \skip\@mpfootins = \skip\footins
6005
```

16.4 Framed boxes

`\fboxsep` The space left by `\fbox` and `\framebox` between the box and the text in it.

`\fboxrule` The width of the rules in the box made by `\fbox` and `\framebox`.

```
6006 \setlength\fboxsep{3\p@}
6007 \setlength\fboxrule{.4\p@}
6008
```

16.5 Equation and eqnarray

`\theequation` The equation counter will be reset at beginning of a new chapter and the equation number will be prefixed by the chapter number. This code must follow the `\chapter` definition, or more exactly the definition of the chapter counter.

```
6009 \@addtoreset{equation}{chapter}
6010 \renewcommand{\theequation}{%
6011 \ifnum \c@chapter>\z@ \thechapter.\fi \@arabic\c@equation}
```


`\jot` `\jot` is the extra space added between lines of an `eqnarray` environment. The default value is used.

```
6012 % \setlength\jot{3pt}
```

`\@eqnnum` The macro `\@eqnnum` defines how equation numbers are to appear in equations. Again the default is used.

```
6013 % \def\@eqnnum{(\theequation)}
```

```
6014
```

17 Array and tabular

17.1 Array

The description and code are essentially copied from the `array` package [MC98].

```
6015
```

```
6016 %%%%%%%%% Array package code %%%%%%%%%%
```

```
6017 %%%%%%%%% With acknowledgements to %%%%%%%%%%
```

```
6018 %%%%%%%%% Frank Mittelbach & David Carlisle %%%%%%%%%%
```

```
6019
```

17.1.1 The construction of the preamble

PW: The original array package has redundant code (caused by a quick mashing of an early array and the newarray packages). I have tried to remove the redundancies and overlaps.

It is obvious that those environments will consist mainly of an `\halign`, because \TeX typesets tables using this primitive. That is why we will now take a look at the algorithm which determines a preamble for a `\halign` starting with a given user preamble using the options mentioned above.

The most interesting macros of this implementation are without doubt those which are responsible for the construction of the preamble for the `\halign`. The underlying algorithm was developed by LAMPORT (resp. KNUTH, see texhax V87#??), and it has been extended and improved.

The user preamble will be read token by token. A token is a single character like `c` or a block enclosed in `{...}`. For example the preamble of `\begin{tabular}{lc|c@{\hspace{1cm}}}` consists of the token `l`, `c`, `|`, `|`, `@` and `\hspace{1cm}`. The currently used token and the one, used before, are needed to decide on how the construction of the preamble has to be continued. In the example mentioned above the `l` causes the preamble to begin with `\hskip\tabcolsep`. Furthermore `# \hfil` would be appended to define a flush left column. The next token is a `c`. Because it was preceded by an `l` it generates a new column. This is done with `\hskip \tabcolsep & \hskip \tabcolsep`. The column which is to be centered will be appended with `\hfil # \hfil`. The token `|` would then add a space of `\hskip \tabcolsep` and a vertical line because the last tokens was a `c`. The following token `|` would only add a space `\hskip \doublerulesep` because

it was preceded by the token `|`. We will not discuss our example further but rather take a look at the general case of constructing preambles.

The example shows that the desired preamble for the `\halign` can be constructed as soon as the action of all combinations of the preamble tokens are specified. There are 18 such tokens so we have $19 \cdot 18 = 342$ combinations if we count the beginning of the preamble as a special token. Fortunately, there are many combinations which generate the same spaces, so we can define token classes. We will identify a token within a class with a number, so we can insert the formatting (for example of a column). Table 5 lists all token classes and their corresponding numbers.

token	\@chclass	\@chnum	token	\@chclass	\@chnum
c	0	0	Start	4	--
l	0	1	@-arg	5	--
r	0	2	!	6	--
p-arg	0	3	@	7	--
t-arg	0	4	<	8	--
b-arg	0	5	>	9	--
	1	0	p	10	3
!-arg	1	1	t	10	4
<-arg	2	--	b	10	5
>-arg	3	--			

Table 5: Classes of preamble tokens

```

\@chclass The class and the number of the current token are saved in the count registers
\@chnum   \@chclass and \@chnum, while the class of the previous token is stored in the
\@lastchclass count register \@lastchclass. All of the mentioned registers are already allocated
              in latex.tex, which is the reason why the following three lines of code are
              commented out. Later throughout the text I will not mention it again explicitly
              whenever I use a % sign. These parts are already defined in latex.tex.

6020 % \newcount \@chclass
6021 % \newcount \@chnum
6022 % \newcount \@lastchclass

\@addtopreamble We will save the already constructed preamble for the \halign in the global
                 macro \@preamble. This will then be enlarged with the command
                 \@addtopreamble.

6023 \def\@addtopreamble#1{\xdef\@preamble{\@preamble #1}}

\@testpach With the help of \@lastchclass we can now define a macro which determines
            the class and the number of a given preamble token and assigns them to the
            registers \@chclass and \@chnum.

6024 \def\@testpach{\@chclass

```

First we deal with the cases in which the token (#1) is the argument of `!`, `@`, `<` or `>`. We can see this from the value of `\@lastchclass`:

```
6025 \ifnum \@lastchclass=6 \@ne \@chnum \@ne \else
6026 \ifnum \@lastchclass=7 5 \else
6027 \ifnum \@lastchclass=8 \tw@ \else
6028 \ifnum \@lastchclass=9 \thr@@
```

Otherwise we will assume that the token belongs to the class 0 and assign the corresponding number to `\@chnum` if our assumption is correct.

```
6029 \else \z@
```

If the last token was a `p`, `m` or a `b`, `\@chnum` already has the right value. This is the reason for the somewhat curious choice of the token numbers in class 10.

```
6030 \ifnum \@lastchclass = 10 \else
```

Otherwise we will check if `\@nextchar` is either a `c`, `l` or an `r`. Some applications change the catcodes of certain characters like “@” in `amstex.sty`. As a result the tests below would fail since they assume non-active character tokens. Therefore we evaluate `\@nextchar` once thereby turning the first token of its replacement text into a char. At this point here this should have been the only char present in `\@nextchar` which put into via a `\def`.

```
6031 \edef\@nextchar{\expandafter\string\@nextchar}%
6032 \@chnum
6033 \if \@nextchar c\z@ \else
6034 \if \@nextchar l\@ne \else
6035 \if \@nextchar r\tw@ \else
```

If it is a different token, we know that the class was not 0. We assign the value 0 to `\@chnum` because this value is needed for the `|`-token. Now we must check the remaining classes. Note that the value of `\@chnum` is insignificant here for most classes.

```
6036 \z@ \@chclass
6037 \if \@nextchar |\@ne \else
6038 \if \@nextchar !6 \else
6039 \if \@nextchar @7 \else
6040 \if \@nextchar <8 \else
6041 \if \@nextchar >9 \else
```

The remaining permitted tokens are `p`, `m` and `b` (class 10).

```
6042 10
6043 \@chnum
6044 \if \@nextchar m\thr@@\else
6045 \if \@nextchar p4 \else
6046 \if \@nextchar b5 \else
```

Now the only remaining possibility is a forbidden token, so we choose class 0 and number 0 and give an error message. Then we finish the macro by closing all `\if`'s.

```
6047 \z@ \@chclass \z@ \@preamerr \z@ \fi \fi \fi \fi
6048 \fi \fi \fi \fi \fi \fi \fi \fi \fi \fi \fi \fi
```

The preamble will be enlarged with the help of `\xdef`, but the arguments of `>`, `<`, `!` and `@` are not supposed to be expanded during the construction (we want an implementation that doesn't need a `\protect`). So we have to find a way to inhibit the expansion of those arguments.

We will solve this problem with `token` registers. We need one register for every `!` and `@`, while we need two for every `c`, `l`, `r`, `m`, `p` or `b`. This limits the number of columns of a table because there are only 256 `token` registers. But then, who needs tables with more than 100 columns?

So how do we proceed? Let us assume that we had `!{foo}` in the user preamble and say we saved `foo` in `token` register 5. Then we call

`\@addtopreamble{\the@toks5}` where `\the@toks` is defined in a way that it does not expand (for example it could be equivalent to `\relax`). Every following call of `\@addtopreamble` leaves `\the@toks5` unchanged in `\@preamble`. If the construction of the preamble is completed we change the definition of `\the@toks` to `\the\toks` and expand `\@preamble` for the last time. During this process all parts of the form `\the@toks⟨Number⟩` will be substituted by the contents of the respective `token` registers.

As we can see from this informal discussion the construction of the preamble has to take place within a group, so that the `token` registers we use will be freed later on. For that reason we keep all assignments to `\@preamble` global; therefore the replacement text of this macro will remain the same after we leave the group.

`\count@` We further need a `count` register to remember which `token` register is to be used next. This will be initialized with -1 if we want to begin with the `token` register 0. We use the PLAIN T_EX scratch register `\count@` because everything takes place locally. All we have to do is insert `\the@toks \the \count@` into the preamble. `\the@toks` will remain unchanged and `\the\count@` expands into the saved number.

`\prepnext@tok` The macro `\prepnext@tok` is in charge of preparing the next `token` register. For that purpose we increase `\count@` by 1:

```
6049 \def\prepnext@tok{\advance \count@ \@ne
```

Then we locally delete any contents the `token` register might have.

```
6050 \toks\count@{}}
```

```
6051
```

`\save@decl` During the construction of the preamble the current `token` is always saved in the macro `\@nextchar` (see the definition of `\@mkpream`). The macro `\save@decl` saves it into the next free `token` register, i.e. in `\toks\count@`. We do not assume that the `token` register is free, we add the new declarations to the front of the register. This is to allow user preambles of the form, `>{foo}>{bar}...`. Users are not encouraged to enter such expressions directly, but they may result from the rewriting of `\newcolumn`'s.

```
6052 \def\save@decl{\toks \count@ = \expandafter\expandafter\expandafter
```

```
6053 \expandafter\@nextchar\the\toks\count@}}
```

```
6054
```

How does the situation look like, if we want to add another column to the preamble, i.e. if we have found a `c`, `l`, `r`, `p`, `m` or `b` in the user preamble? In this case we have the problem of the `token` register from `>{..}` and `<{..}` having to be inserted at this moment because formatting instructions like `\hfil` have to be set around them. On the other hand it is not known yet, if any `<{..}` instruction will appear in the user preamble at all.

We solve this problem by adding two `token` registers at a time. This explains, why we have freed the `token` registers in `\prepnext@tok`.

`\insert@column` We now define the macro `\insert@column` which will do this work for us.

```
\@sharp 6055 \def\insert@column{%
```

Here, we assume that the count register `\@tempcnta` has saved the value `\count@-1`.

```
6056 \the@toks \the \@tempcnta
```

Next follows the `#` sign which specifies the place where the text of the column shall be inserted. To avoid errors during the expansions in `\@addtopreamble` we hide this sign in the command `\@sharp` which is temporarily occupied with `\relax` during the build-up of the preamble. To remove unwanted spaces before and after the column text, we set an `\ignorespaces` in front and a `\unskip` afterwards.

```
6057 \ignorespaces \@sharp \unskip
```

Then the second `token` register follows whose number should be saved in `\count@`. We make sure that there will be no further expansion after reading the number, by finishing with `\relax`. The case above is not critical since it is ended by `\ignorespaces`.

```
6058 \the@toks \the \count@ \relax}
```

`\m@mold@addamp` In the preamble a `&` has to be inserted between any two columns; before the first column there should not be a `&`. As the user preamble may start with a `|` we have to remember somehow if we have already inserted a `#` (i.e. a column). This is done with the boolean variable `\if@firstamp` that we test in `\@addamp`, the macro that inserts the `&`.

PW: Later on, for horizontal lines in ‘continuous’ tabulars, I need to know how many columns there are in a tabular. I need a modified kernel’s `\@addamp` to do this and use the kernel’s `\@curtab` (a counter used in tabbing) to store the number of columns.

`\m@mold@addamp` stores the kernel’s definition of `\@addamp`, and `\m@m@addamp` is the revised definition.

```
6059 % \newif \if@firstamp
6060 \let\m@mold@addamp\@addamp
6061 \newcommand*{\m@m@addamp}{%
6062 \if@firstamp
6063 \@firstampfalse
6064 \global\@curtab\@ne
6065 \else
```

```

6066 \addtopreamble{&}
6067 \global\advance\@curtab\@ne
6068 \fi}
6069 \let\@addamp\m@m@addamp
6070

```

\acol We will now define some abbreviations for the extensions, appearing most often in the preamble build-up. Here \colsep is a dimen register which is set equivalent to \arraycolsep in an array-environment, otherwise it is set equivalent to \tabcolsep.

```

6071 \newdimen\colsep
6072 \def\acol{\addtopreamble{\hskip\colsep}}
6073 % \def\acolampacol{\acol\@addamp\acol}

```

\mkpream Now we can define the macro which builds up the preamble for the \halign. \thetoks First we initialize \@preamble, \@lastchclass and the boolean variable \iffirstamp.

```

6074 \def\mkpream#1{\gdef\@preamble{}\@lastchclass 4 \@firstamptrue

```

During the build-up of the preamble we cannot directly use the # sign; this would lead to an error message in the next \addtopreamble call. Instead, we use the command \sharp at places where later a # will be. This command is at first given the meaning \relax; therefore it will not be expanded when the preamble is extended. In the macro \array, shortly before the \halign is carried out, \sharp is given its final meaning.

In a similar way, we deal with the commands \startpbox and \endpbox, although the reason is different here: these macros expand in many tokens which would delay the build-up of the preamble.

```

6075 \let\sharp\relax \let\startpbox\relax \let\endpbox\relax

```

Now we remove possible *-forms and user-defined column specifiers in the user preamble by repeatedly executing the list \NC@list until the re-writes have no more effect. The expanded preamble will then be in the token register \temptokena. Actually we need to know at this point that this is not \toks0.

```

6076 \temptokena{#1}\tempswatrue
6077 \@whilesw\iftempswa\fi{\@tempswafalse\the\NC@list}%

```

Afterwards we initialize all registers and macros, that we need for the build-up of the preamble. Since we want to start with the token register 0, \count@ has to contain the value -1.

```

6078 \count@\m@ne
6079 \let\thetoks\relax

```

Then we call up \prepnext@tok in order to prepare the token register 0 for use.

```

6080 \prepnext@tok

```

Having expanded all tokens defined using \newcolumnntype (including *), we evaluate the remaining tokens, which are saved in \temptokena. We use the L^AT_EX-macro \tfor to inspect each token in turn. The strange appearing

construction with `\expandafter` is based on the fact that we have to put the replacement text of `\@tempa` and not the macro `\@tempa` to this L^AT_EX-macro.

```
6081 \expandafter \@tfor \expandafter \@nextchar
6082 \expandafter : \expandafter =\the\@temptokena \do
```

The body of this loop (the group after the `\do`) is executed for one token at a time, whereas the current token is saved in `\@nextchar`. At first we evaluate the current token with the already defined macro `\@testpach`, i.e. we assign to `\@chclass` the character class and to `\@chnum` the character number of this token.

```
6083 {\@testpach
```

Then we branch out depending on the value of `\@chclass` into different macros that extend the preamble respectively.

```
6084 \ifcase \@chclass \@classz \or \@classi \or \@classii
6085 \or \save@decl \or \or \@classv \or \@classvi
6086 \or \@classvii \or \@classviii
```

Class 9 is equivalent to class 10.

```
6087 \or \@classx
6088 \or \@classx \fi
```

Two cases deserve our special attention: Since the current token cannot have the character class 4 (start) we have skipped this possibility. If the character class is 3, only the content of `\@nextchar` has to be saved into the current token register; therefore we call up `\save@decl` directly and save a macro name. After the preamble has been extended we assign the value of `\@chclass` to the counter `\@lastchclass` to assure that this information will be available during the next run of the loop.

```
6089 \@lastchclass\@chclass}%
```

After the loop has been finished space must still be added to the created preamble, depending on the last token. Depending on the value of `\@lastchclass` we perform the necessary operations.

```
6090 \ifcase \@lastchclass
```

If the last class equals 0 we add a `\hskip \col@sep`.

```
6091 \@acol \or
```

If it equals 1 we do not add any additional space so that the horizontal lines do not exceed the vertical ones.

```
6092 \or
```

Class 2 is treated like class 0 because a `<{...}` can only directly follow after class 0.

```
6093 \@acol \or
```

Most of the other possibilities can only appear if the user preamble was defective. Class 3 is not allowed since after a `>{..}` there must always follow a `c`, `l`, `r`, `p,m` or `b`. We report an error and ignore the declaration given by `{...}`.

```
6094 \@preamerr \thr@@ \or
```

If `\@lastchclass` is 4 the user preamble has been empty. To continue, we insert a # in the preamble.

```
6095 \preamerr \tw@ \@addtopreamble\@sharp \or
```

Class 5 is allowed again. In this case (the user preamble ends with `@{...}`) we need not do anything.

```
6096 \or
```

Any other case means that the arguments to `@`, `!`, `<`, `>`, `p`, `m` or `b` have been forgotten. So we report an error and ignore the last token.

```
6097 \else \@preamerr \@ne \fi
```

Now that the build-up of the preamble is almost finished we can insert the token registers and therefore redefine `\the@toks`. The actual insertion, though, is performed later.

```
6098 \def\the@toks{\the\toks{}}
```

The preamble is extended by the macros `\@classz` to `\@classx` which are called by `\@mkpream` depending on `\@lastchclass` (i.e. the character class of the last token).

`\@classx` First we define `\@classx` because of its important rôle. When it is called we find that the current token is `p`, `m` or `b`. That means that a new column has to start.

```
6099 \def\@classx{%
```

Depending on the value of `\@lastchclass` different actions must take place:

```
6100 \ifcase \@lastchclass
```

If the last character class was 0 we separate the columns by `\hskip\col@sep` followed by `&` and another `\hskip\col@sep`.

```
6101 \@acolampacol \or
```

If the last class was class 1 — that means that a vertical line was drawn, — before this line a `\hskip\col@sep` was inserted. Therefore there has to be only a `&` followed by `\hskip\col@sep`. But this `&` may be inserted only if this is not the first column. This process is controlled by `\if@firststamp` in the macro `\addamp`.

```
6102 \@addamp \@acol \or
```

Class 2 is treated like class 0 because `<{...}` can only follow after class 0.

```
6103 \@acolampacol \or
```

Class 3 requires no actions because all things necessary have been done by the preamble token `>`.

```
6104 \or
```

Class 4 means that we are at the beginning of the preamble. Therefore we start the preamble with `\hskip\col@sep` and then call `\@firststampfalse`. This makes sure that a later `\@addamp` inserts the character `&` into the preamble.

```
6105 \@acol \@firststampfalse \or
```

For class 5 tokens only the character `&` is inserted as a column separator. Therefore we call `\@addamp`.

```
6106 \@addamp
```


Other cases are impossible. For an example `\@lastchclass = 6` — a s it might appear in a preamble of the form `...!p...` — p would have been taken as an argument of ! by `\@testpach`.

```
6107 \fi}
```

`\@classz` If the character class of the last token is 0 we have c, l, r or an argument of m, b or p. In the first three cases the preamble must be extended the same way as if we had class 10. The remaining two cases do not require any action because the space needed was generated by the last token (i.e. m, b or p). Since `\@lastchclass` has the value 10 at this point nothing happens when `\@classx` is called. So the macro `\@chclassz` may start like this:

```
6108 \def\@classz{\@classx
```

According to the definition of `\insert@column` we must store the number of the token register in which a preceding `>{..}` might have stored its argument into `\@tempcnta`.

```
6109 \@tempcnta \count@
```

To have `\count@ = \@tempcnta + 1` we prepare the next token register.

```
6110 \prepnext@tok
```

Now the preamble must be extended with the column whose format can be determined by `\@chnum`.

```
6111 \@addtopreamble{\ifcase \@chnum
```

If `\@chnum` has the value 0 a centered column has to be generated. So we begin with stretchable space.

```
6112 \hfil
```

The command `\d@llarbegin` follows expanding into `\begingroup` (in the `tabular`-environment) or into `$`. Doing this (provided an appropriate setting of `\d@llarbegin`) we achieve that the contents of the columns of an `array`-environment are set in math mode while those of a `tabular`-environment are set in LR mode.

```
6113 \d@llarbegin
```

Now we insert the contents of the two token registers and the symbol for the column entry (i.e. # or more precise `\@sharp`) using `\insert@column`.

```
6114 \insert@column
```

We end this case with `\d@llarend` and `\hfil` where `\d@llarend` again is either `$` or `\endgroup`.

```
6115 \d@llarend \hfil \or
```

The templates for l and r (i.e. `\@chnum 1` or `2`) are generated the same way. Since one `\hfil` is missing the text is moved to the relevant side. The `\kern\z@` is needed in case of an empty column entry. Otherwise the `\unskip` in `\insert@column` removes the `\hfil`. Changed to `\hskip1sp` so that it interacts better with `\@bsphack`.

```
6116 \hskip1sp\d@llarbegin \insert@column \d@llarend \hfil \or
```

```
6117 \hfil\hskip1sp\d@llarbegin \insert@column \d@llarend \or
```

The templates for `p`, `m` and `b` mainly consist of a box. In case of `m` it is generated by `\vcenter`. This command is allowed only in math mode. Therefore we start with a `$`.

```
6118    $\vcenter{%
```

The part of the templates which is the same in all three cases (`p`, `m` and `b`) is built by the macros `\@startpbox` and `\@endpbox`. `\@startpbox` has an argument: the width of the column which is stored in the current token (i.e. `\@nextchar`).

Between these two macros we find the well known `\insert@column`.

```
6119    \@startpbox{\@nextchar}\insert@column \@endpbox $ \or%
```

The templates for `p` and `b` are generated in the same way though we do not need the `$` characters because we use `\vtop` or `\vbox`.

```
6120    \vtop \@startpbox{\@nextchar}\insert@column \@endpbox \or
```

```
6121    \vbox \@startpbox{\@nextchar}\insert@column \@endpbox
```

Other values for `\@chnum` are impossible. Therefore we end the arguments to `\@addtopreamble` and `\ifcase`. Before we come to the end of `\@classz` we have to prepare the next token register.

```
6122    \fi}\prepnext@tok}
```

```
6123
```

`\@classix` Class 9 (`>`-token) prevented repeated `>` declarations for the same column. This restriction has been eased, making class 9 equivalent to class 10.

```
6124 \let\@classix\relax
```

```
6125
```

`\@classviii` If the current token is a `<` the last character class must be 0, or 2 (as repeated `<` expressions are allowed). In this case it is not necessary to extend the preamble. Otherwise we output an error message, set `\@chclass` to 6 and call `\@classvi`. By doing this we achieve that `<` is treated like `!`.

```
6126 \def\@classviii{\ifnum \@lastchclass >\z@\ifnum \@lastchclass=\tw@\else
```

```
6127     \@preamerr 4\@chclass 6 \@classvi \fi\fi}
```

```
6128
```

`\@arrayrule` There is only one incompatibility with the original definition: the definition of `\@arrayrule`. In the original a line without width (e.g., the space between `cc` and `c|c` is equal) is created by multiple insertions of `\hskip .5\arrayrulewidth`. We only insert a vertical line into the preamble. This is done to prevent problems with `TEX`'s main memory when generating tables with many vertical lines in them (especially in the case of floats).

```
6129 \def\@arrayrule{\@addtopreamble \vline}
```

```
6130
```

`\@classvii` As a consequence it follows that in case of class 7 (`@` token) the preamble need not to be extended. In the original definition `\@lastchclass = 1` is treated by inserting `\hskip .5\arrayrulewidth`. We only check if the last token was of class 3 which is forbidden.

```
6131 \def\@classvii{\ifnum \@lastchclass = \thr@@
```

If this is true we output an error message and ignore the declarations stored by the last `>{...}`, because these are overwritten by the argument of `@`.

```
6132 \@@preamerr \thr@@ \fi}
6133
```

`\@classvi` If the current token is a regular `!` and the last class was 0 or 2 we extend the preamble with `\hskip\col@sep`. If the last token was of class 1 (for instance `|`) we extend with `\hskip \doublerulesep` because the construction `!{...}` has to be treated like `|`.

```
6134 \def\@classvi{\ifcase \@lastchclass
6135     \@acol \or
6136     \@addtopreamble{\hskip \doublerulesep}\or
6137     \@acol \or
```

Now `\@preamerr...` should follow because a user preamble of the form `..>{...}!` is not allowed. To save memory we call `\@classvii` instead which also does what we want.

```
6138     \@classvii
```

If `\@lastchclass` is 4 or 5 nothing has to be done. Class 6 to 10 are not possible. So we finish the macro.

```
6139     \fi}
6140
```

`\@classii` In the case of character classes 2 and 3 (i.e. the argument of `<` or `>`) we only have
`\@classiii` to store the current token (`\@nextchar`) into the corresponding token register since the preparation and insertion of these registers are done by the macro `\@classz`. This is equivalent to calling `\save@decl` in the case of class 3. To save command identifiers we do this call up in the macro `\@mkpream`. Class 2 exhibits a more complicated situation: the token registers have already been inserted by `\@classz`. So the value of `\count@` is too high by one. Therefore we decrease `\count@` by 1.

```
6141 \def\@classii{\advance \count@ \m@ne
```

Next we store the current token into the correct token register by calling `\save@decl` and then increase the value of `\count@` again. At this point we can save memory once more (at the cost of time) if we use the macro `\prepnext@tok`.

```
6142     \save@decl\prepnext@tok}
6143
```

`\@classv` Class 5 is `@`-expressions (and is also called by class 1) We do not expand the `@`-expression, but instead explicitly replace an `\extracolsep` command by an assignment to `\tabskip` by a method similar to the `\newcolumn`type system described later.

```
6144 \def\@classv{\save@decl
6145     \expandafter\NC@ecs\@nextchar\extracolsep{}\extracolsep\@@@
6146     \@addtopreamble{\dollarbegin\the@toks\the\count@\relax\dollarend}%
6147     \prepnext@tok}
```

`\NC@ecs` Rewrite the first occurrence of `\extracolsep{1in}` to `\tabskip1in\relax`. As a side effect discard any tokens after a second `\extracolsep`, there is no point in the user entering two of these commands anyway, so this is not really a restriction.

```
6148 \def\NC@ecs#1\extracolsep#2#3\extracolsep#4\@@@{\def\@tempa{#2}%
6149 \ifx\@tempa\@empty\else\toks\count@=#1\tabskip#2\relax#3}\fi}
6150
```

`\@classi` In the case of class 0 we were able to generate the necessary space between columns by using the macro `\@classx`. Analogously the macro `\@classvi` can be used for class 1.

```
6151 \def\@classi{\@classvi
    Depending on \@chnum a vertical line
6152 \ifcase \@chnum \@arrayrule \or
    or (in case of !{...}) the current token — stored in \@nextchar — has to be
    inserted into the preamble. This corresponds to calling \@classv.
6153 \@classv \fi}
6154
```

`\@startpbox` In `\@classz` the macro `\@startpbox` is used. The width of the parbox is passed as an argument. `\vcenter`, `\vtop` or `\vbox` are already in the preamble. So we start with the braces for the wanted box.

```
6155 \def\@startpbox#1{\bgroup
    The argument is the width of the box. This information has to be assigned to
    \hsize. Then we assign default values to several parameters used in a parbox.
6156 \setlength\hsize{#1}\@arrayparboxrestore
```

Our main problem is to obtain the same distance between succeeding lines of the parbox. We have to remember that the distance between two parboxes should be defined by `\@arstrut`. That means that it can be greater than the distance in a parbox. Therefore it is not enough to set a `\@arstrut` at the beginning and at the end of the parbox. This would dimension the distance between first and second line and the distance between the two last lines of the parbox wrongly. To prevent this we set an invisible rule of height `\@arstrutbox` at the beginning of the parbox. This has no effect on the depth of the first line. At the end of the parbox we set analogously another invisible rule which only affects the depth of the last line. It is necessary to wait inserting this strut until the paragraph actually starts to allow for things like `\parindent` changes via `>{...}`.

```
6157 \everypar{%
6158 \vrule \@height \ht\@arstrutbox \@width \z@
6159 \everypar{}}%
6160 }
```

`\@endpbox` If there are any declarations defined by `>{...}` and `<{...}` they now follow in the macro `\@classz` — the contents of the column in between. So the macro

`\@endpbox` must insert the `specialstrut` mentioned earlier and then close the group opened by `\@startpbox`.

```
6161 \def\@endpbox{\@finalstrut\@arstrutbox \egroup\hfil}
```

17.1.2 Building and calling `\halign`

`\@array` After we have discussed the macros needed for the evaluation of the user preamble we can define the macro `\@array` which uses these macros to create a `\halign`. It has two arguments. The first one is a position argument which can be `t`, `b` or `c`; the second one describes the wanted preamble, e.g. it has the form `|c|c|c|`.

```
6162 \def\@array[#1]#2{%
```

First we define a `strut` whose size basically corresponds to a normal `strut` multiplied by the factor `\arraystretch`. This `strut` is then inserted into every row and enforces a minimal distance between two rows. Nevertheless, when using horizontal lines, large letters (like accented capital letters) still collide with such lines. Therefore at first we add to the height of a normal `strut` the value of the parameter `\extrarowheight`.

```
6163   \@tempdima \ht \strutbox
6164   \advance \@tempdima by\extrarowheight
6165   \setbox \@arstrutbox \hbox{\vrule
6166       \@height \arraystretch \@tempdima
6167       \@depth \arraystretch \dp \strutbox
6168       \@width \z@}%
```

Then we open a group, in which the user preamble is evaluated by the macro `\@mkpream`. As we know this must happen locally. This macro creates a preamble for a `\halign` and saves its result globally in the control sequence `\@preamble`.

```
6169   \begingroup
6170   \@mkpream{#2}%
```

We again redefine `\@preamble` so that a call up of `\@preamble` now starts the `\halign`. Thus also the arguments of `>`, `<`, `@` and `!`, saved in the `token` registers are inserted into the preamble. The `\tabskip` at the beginning and end of the preamble is set to `0pt` (in the beginning by the use of `\ialign`). Also the command `\@arstrut` is build in, which inserts the `\@arstrutbox`, defined above. Of course, the opening brace after `\ialign` has to be implicit as it will be closed in `\endarray` or another macro.

The `\noexpand` in front of `\ialign` does no harm in standard \LaTeX and was added since some experimental support for using text glyphs in math redefines `\halign` with the result that it becomes expandable with disastrous results in cases like this. In the kernel definition for this macro the problem does not surface because there `\protect` is set (which is not necessary in this implementation as there is no arbitrary user input that can get expanded) and the experimental code made the redefinition robust. Whether this is the right approach is open to question; consider the `\noexpand` a curtesy to allow an

unsupported redefinition of a T_EX primitive for the moment (as people rely on that experimental code).

```
6171 \xdef\@preamble{\noexpand \ialign \@halign to
6172             \bgroup \@arstrut \@preamble
6173             \tabskip \z@ \cr}%
```

What we have not explained yet is the macro `\@halign to` that was just used. Depending on its replacement text the `\halign` becomes a `\halign to <dimen>`. Now we close the group again. Thus `\@startpbox` and `\@endpbox` as well as all token registers get their former meaning back.

```
6174 \endgroup
```

To support other packages we include a hook into this part of the code which is a no-op in the main package.

```
6175 \@arrayleft
```

Now we decide depending on the position argument in which box the `\halign` is to be put. (`\vcenter` may be used because we are in math mode.)

```
6176 \if #1t\vtop \else \if#1b\vbox \else \vcenter \fi \fi
```

Now another implicit opening brace appears; then definitions which shall stay local follow. While constructing the `\@preamble` in `\@mkpream` the `#` sign must be hidden in the macro `\@sharp` which is `\let to \relax` at that moment (see definition of `\@mkpream`). All these now get their actual meaning.

```
6177 \bgroup
6178 \let \@sharp ##\let \protect \relax
```

With the above defined struts we fix down the distance between rows by setting `\lineskip` and `\baselineskip` to `0pt`. Since there have to be set `$`'s around every column in the `array`-environment the parameter `\mathsurround` should also be set to `0pt`. This prevents additional space between the rows. The PLAIN T_EX-macro `\m@th` does this.

```
6179 \lineskip \z@
6180 \baselineskip \z@
6181 \m@th
```

Beside, we have to assign a special meaning (which we still have to specify) to the line separator `\\`. We also have to redefine the command `\par` in such a way that empty lines in `\halign` cannot do any damage. We succeed in doing so by choosing something that will disappear when expanding. After that we only have to call up `\@preamble` to start the wanted `\halign`.

```
6182 \let\\ \@arraycr \let\tabularnewline\\ \let\par \@empty \@preamble}
```

`\extrarowheight` The `dimen` parameter used above also needs to be allocated. As a default value we use `0pt`, to ensure compatibility with standard L^AT_EX.

```
6183 \newdimen \extrarowheight
6184 \extrarowheight=0pt
```

`\@arstrut` Now the insertion of `\@arstrutbox` through `\@arstrut` is easy since we know exactly in which mode T_EX is while working on the `\halign` preamble.

```
6185 \def\@arstrut{\unhcopy\@arstrutbox}
```

17.1.3 The line separator \\\

`\@arraycr` In the macro `\@array` the line separator `\\` is `\let` to the command `\@arraycr`. Its definition starts with a special brace which I have directly copied from the original definition. It is necessary, because the `\futurlet` in `\@ifnextchar` might expand a following `&` token in a construction like `\\ &`. This would otherwise end the alignment template at a wrong time. On the other hand we have to be careful to avoid producing a real group, i.e., `{}`, because the command will also be used for the array environment, i.e., in math mode. In that case an extra `{}` would produce an ord atom which could mess up the spacing. For this reason we use a combination that does not really produce a group at all but modifies the master counter so that a `&` will not be considered belonging to the current `\halign` while we are looking for a `*` or `[`. For further information see [Knu84, Appendix D].

```
6186 \def\@arraycr{\relax\iffalse{\fi\ifnum 0='{}\fi}
```

Then we test whether the user is using the star form and ignore a possible star (I also disagree with this procedure, because a star does not make any sense here).

```
6187 \ifstar \@xarraycr \@xarraycr}
```

`\@xarraycr` In the command `\@xarraycr` we test if an optional argument exists.

```
6188 \def\@xarraycr{\@ifnextchar [%
```

If it does, we branch out into the macro `\@argarraycr` if not we close the special brace (mentioned above) and end the row of the `\halign` with a `\cr`.

```
6189 \@argarraycr {\ifnum 0='{}\fi\cr}}
```

`\@argarraycr` If additional space is requested by the user this case is treated in the macro `\@argarraycr`. First we close the special brace and then we test if the additional space is positive.

```
6190 \def\@argarraycr[#1]{\ifnum 0='{}\fi\ifdim #1>\z@
```

If this is the case we create an invisible vertical rule with depth `\dp\@arstrutbox + <wanted space>`. Thus we achieve that all vertical lines specified in the user preamble by a `|` are now generally drawn. Then the row ends with a `\cr`.

If the space is negative we end the row at once with a `\cr` and move back up with a `\vskip`.

While testing these macros I found out that the `\endtemplate` created by `\cr` and `&` is something like an `\outer` primitive and therefore it should not appear in incomplete `\if` statements. Thus the following solution was chosen which hides the `\cr` in other macros when T_EX is skipping conditional text.

```
6191 \expandafter\@xargarraycr\else
```

```
6192 \expandafter\@yargarraycr\fi{#1}}
```

`\@xargarraycr` The following macros were already explained above.

`\@yargarraycr` 6193 `\def\@xargarraycr#1{\unskip`

```
6194 \@tempdima #1\advance\@tempdima \dp\@arstrutbox
```

```
6195 \vrule \@depth\@tempdima \@width\z@ \cr}
```

```
6196 \def\@yargarraycr#1{\cr\noalign{\vskip #1}}
```

17.1.4 Spanning several columns

`\multicolumn` If several columns should be held together with a special format the command `\multicolumn` must be used. It has three arguments: the number of columns to be covered; the format for the result column and the actual column entry.

```
6197 \long\def\multicolumn#1#2#3{%
```

First we combine the given number of columns into a single one; then we start a new block so that the following definition is kept local.

```
6198   \multispan{#1}\begingroup
```

Since a `\multicolumn` should only describe the format of a result column, we redefine `\@addamp` in such a way that one gets an error message if one uses more than one `c`, `l`, `r`, `p`, `m` or `b` in the second argument. One should consider that this definition is local to the build-up of the preamble; an `array-` or `tabular-`environment in the third argument of the `\multicolumn` is therefore worked through correctly as well.

```
6199   \def\@addamp{\if@firstamp \@firstampfalse \else
6200               \@preammerr 5\fi}%
```

Then we evaluate the second argument with the help of `\@mkpream`. Now we still have to insert the contents of the `token` register into the `\@preamble`, i.e. we have to say `\xdef\@preamble{\@preamble}`. This is achieved shorter by writing:

```
6201   \@mkpream{#2}\@addtopreamble\@empty
```

After the `\@preamble` is created we forget all local definitions and occupations of the `token` registers.

```
6202   \endgroup
```

In the special situation of `\multicolumn` `\@preamble` is not needed as preamble for a `\halign` but it is directly inserted into our table. Thus instead of `\sharp` there has to be the column entry (`#3`) wanted by the user.

```
6203   \def\@sharp{#3}%
```

Now we can pass the `\@preamble` to `TeX`. For safety we start with an `\@arstrut`. This should usually be in the template for the first column however we do not know if this template was overwritten by our `\multicolumn`. We also add a `\null` at the right end to prevent any following `\unskip` (for example from `\\[...]`) to remove the `\tabcolsep`.

```
6204   \@arstrut \@preamble
6205   \null
6206   \ignorespaces}
```

17.1.5 The environment definitions

After these preparations we are able to define the environments. They only differ in the initialisations of `\dollar...`, `\colsep` and `\@halignto`.

`\@halignto` In order to relieve the save stack we assign the replacement texts for `\@halignto`
`\dollarbegin` globally. `\dollar` has to be local since otherwise nested `tabular` and `array`
`\dollarend`

environments (via `\multicolumn`) are impossible. When the new font selection scheme is in force we have to surround all `\halign` entries with braces. See remarks in TUGboat 10#2. Actually we are going to use `\begingroup` and `\endgroup`. However, this is only necessary when we are in text mode. In math the surrounding dollar signs will already serve as the necessary extra grouping level. Therefore we switch the settings of `\dollarbegin` and `\dollarend` between groups and dollar signs.

```
6207 \let\dollarbegin\begingroup
6208 \let\dollarend\endgroup
```

`\array` Our new definition of `\array` then reads:

```
6209 \def\array{\col@sep\arraycolsep
6210 \def\dollarbegin{$}\let\dollarend\dollarbegin\gdef\halignto{}}%
```

Since there might be an optional argument we call another macro which is also used by the other environments.

```
6211 \@tabarray}
```

`\@tabarray` This macro tests for an optional bracket and then calls up `\@array` or `\@array[c]` (as default).

```
6212 \def\@tabarray{\ifnextchar[{\@array}{\@array[c]}}
```

`\@array` This macro could then test an optional delimiter before the left brace of the main preamble argument. At this point it simply is let to be `\array`.

```
6213 \let\@array\array
```

`\tabular` The environments `tabular` and `tabular*` differ only in the initialisation of the `\tabular*` command `\halignto`. Therefore we define

```
6214 \def\tabular{\gdef\halignto{}}\@tabular}
```

and analogously for the star form. We evaluate the argument first using `\setlength` so that users of the `calc` package can write code like `\begin{tabular*}{(\columnwidth-1cm)/2}...`

```
6215 \expandafter\def\csname tabular*\endcsname#1{%
6216 \setlength\dimen@{#1}%
6217 \xdef\halignto{to\the\dimen@}\@tabular}
```

`\@tabular` The rest of the job is carried out by the `\@tabular` macro:

```
6218 \def\@tabular{%
```

First of all we have to make sure that we start out in hmode. Otherwise we might find our table dangling by itself on a line.

```
6219 \leavevmode
```

It should be taken into consideration that the macro `\@array` must be called in math mode. Therefore we open a box, insert a `$` and then assign the correct values to `\col@sep` and `\dollar`...

```
6220 \hbox \bgroup $\col@sep\tabcolsep \let\dollarbegin\begingroup%$
6221 \let\dollarend\endgroup
```

Now everything tabular specific is done and we are able to call the `\@tabarray` macro.

```
6222 \endarray}
```

`\endarray` When the processing of `array` is finished we have to close the `\halign` and afterwards the surrounding box selected by `\@array`. To save token space we then redefine `\@preamble` because its replacement text isn't longer needed. We have to declare the hook, `\@arrayleft`, we put into `\@array` above. A similar hook '`\@arrayright`' is inserted into the `\endarray` to gain control. Both defaults to empty.

```
6223 \def\endarray{\crrc \egroup \egroup \@arrayright \gdef\@preamble{}}
```

```
6224 \let\@arrayleft\@empty
```

```
6225 \let\@arrayright\@empty
```

`\endtabular` To end a tabular or `tabular*` environment we call up `\endarray`, close the math mode and then the surrounding `\hbox`.

```
\endtabular*
```

```
6226 \def\endtabular{\endarray $\egroup}%
```

```
6227 \expandafter\let\csname endtabular*\endcsname=\endtabular
```

We should `\let` all macros to `\relax` that were used in the kernel but are no longer necessary.

```
6228 \let\@ampacol=\relax \let\@expast=\relax
```

```
6229 \let\@arrayclassiv=\relax \let\@arrayclassz=\relax
```

```
6230 \let\@tabclassiv=\relax \let\@tabclassz=\relax
```

```
6231 \let\@arrayacol=\relax \let\@tabacol=\relax
```

```
6232 \let\@tabularcrrc=\relax \let\@endpbox=\relax
```

```
6233 \let\@argtabularcrrc=\relax \let\@xtabularcrrc=\relax
```

`\@preamerr` We also have to redefine the error routine `\@preamerr` since new kind of errors are possible. The code for this macro is not perfect yet; it still needs too much memory.

```
6234 \def\@preamerr#1{\def\@tempd{{..} at wrong position: }%
```

```
6235 \ClassError{memoir}{%
```

```
6236 \ifcase #1 Illegal pream-token (\@nextchar): 'c' used\or %0
```

```
6237 Missing arg: token ignored\or %1
```

```
6238 Empty preamble: 'l' used\or %2
```

```
6239 >\@tempd token ignored\or %3
```

```
6240 <\@tempd changed to !{..}\or %4
```

```
6241 Only one column-spec. allowed.\fi}\@ehc} %5
```

```
6242
```

17.1.6 Defining your own column specifiers

`\newcolumntype` In `newarray.sty` the macro for specifying new columns was named `\newcolumn`. When the functionality was added to `array.sty` the command was renamed `\newcolumntype`. The `\newcolumntype` macro gives users the chance to define letters, to be used in the same way as the primitive column specifiers, 'c' 'p' etc.

```
6243 \def\newcolumntype#1{%
```

\NC@char is so that active characters, like @ in AMS \LaTeX may be used. Note that we need to use the possibly active token, #1, in several places, as that is the token that actually appears in the preamble argument.

```
6244 \edef\NC@char{\string#1}%
```

First we check whether there is already a definition for this column. Unlike \newcommand we give a warning rather than an error if it is defined. If it is a new column, add \NC@do <column> to the list \NC@list.

```
6245 \@ifundefined{NC@find@\NC@char}%
6246   {\@tfor\next:=<>\clrbp@!|\do{\if\noexpand\next\NC@char
6247     \@memwarn{Redefining primitive column \NC@char}\fi}%
6248   \NC@list\expandafter{\the\NC@list\NC@do#1}}%
6249   {\@memwarn{Column \NC@char\space is already defined}}%
```

Now we define a macro with an argument delimited by the new column specifier, this is used to find occurrences of this specifier in the user preamble.

```
6250 \namedef{NC@find@\NC@char}##1#1{\NC@{##1}}%
```

If an optional argument was not given, give a default argument of 0.

```
6251 \@ifnextchar[{\newcol@{\NC@char}}{\newcol@{\NC@char}[0]}}
```

\newcol@ We can now define the macro which does the rewriting, \reargdef takes the same arguments as \newcommand, but does not check that the command is new. For a column, say ‘D’ with one argument, define a command \NC@rewrite@D with one argument, which recursively calls \NC@find on the user preamble after replacing the first token or group with the replacement text specified in the \newcolumn type command. \NC@find will find the next occurrence of ‘D’ as it will be \let equal to \NC@find@D by \NC@do.

```
6252 \def\newcol@#1[#2]#3{\expandafter\reargdef
6253   \csname NC@rewrite@#1\endcsname[#2]{\NC@find#3}}
```

\NC@ Having found an occurrence of the new column, save the preamble before the column in \@temptokena, then check to see if we are at the end of the preamble. (A dummy occurrence of the column specifier will be placed at the end of the preamble by \NC@do).

```
6254 \def\NC@#1{%
6255   \@temptokena\expandafter{\the\@temptokena#1}\futurelet\next\NC@ifend}
```

\NC@ifend We can tell that we are at the end as \NC@do will place a \relax after the dummy column.

```
6256 \def\NC@ifend{%
```

If we are at the end, do nothing. (The whole preamble will now be in \@temptokena.)

```
6257   \ifx\next\relax
```

Otherwise set the flag \if@tempwa, and rewrite the column.

```
6258   \else\@tempwa true\expandafter\NC@rewrite\fi}
```

`\NC@do` If the user has specified ‘C’ and ‘L’ as new columns, the list of rewrites (in the token register `\NC@list`) will look like `\NC@do * \NC@do C \NC@do L`. So we need to define `\NC@do` as a one argument macro which initialises the rewriting of the specified column. Let us assume that ‘C’ is the argument.

```
6259 \def\NC@do#1{%
```

First we let `\NC@rewrite` and `\NC@find` be `\NC@rewrite@C` and `\NC@find@C` respectively.

```
6260 \expandafter\let\expandafter\NC@rewrite
6261 \csname NC@rewrite@\string#1\endcsname
6262 \expandafter\let\expandafter\NC@find
6263 \csname NC@find@\string#1\endcsname
```

Clear the token register `\@temptokena` after putting the present contents of the register in front of the token `\NC@find`. At the end we place the tokens ‘C\relax’ which `\NC@ifend` will use to detect the end of the user preamble.

```
6264 \expandafter\@temptokena\expandafter{\expandafter}%
6265 \expandafter\NC@find\the\@temptokena#1\relax}
```

`\showcols` This macro is useful for debugging `\newcolumn` type specifications, it is the equivalent of the primitive `\show` command for macro definitions. All we need to do is locally redefine `\NC@do` to take its argument (say ‘C’) and then `\show` the (slightly modified) definition of `\NC@rewrite@C`. Actually as the list always starts off with `\NC@do *` and we do not want to print the definition of the *-form, define `\NC@do` to throw away the first item in the list, and then redefine itself to print the rest of the definitions.

```
6266 \def\showcols{{\def\NC@do#1{\let\NC@do\NC@show}\the\NC@list}}
```

`\NC@show` If the column ‘C’ is defined as above, then `\show\NC@rewrite@C` would output `\long macro: ->\NC@find >{\$}c<{\$}`. We want to strip the `\long macro: ->` and the `\NC@find`. So first we use `\meaning` and then apply the macro `\NC@strip` to the tokens so produced and then `\typeout` the required string.

```
6267 \def\NC@show#1{%
6268 \typeout{Column #1\expandafter\expandafter\expandafter\NC@strip
6269 \expandafter\meaning\csname NC@rewrite@#1\endcsname\@@}}
```

`\NC@strip` Delimit the arguments to `\NC@strip` with ‘:’, ‘->’, a space, and `\@@` to pull out the required parts of the output from `\meaning`.

```
6270 \def\NC@strip#1:#2->#3 #4\@@{#2 -> #4}
```

`\NC@list` Allocate the token register used for the rewrite list.

```
6271 \newtoks\NC@list
```

17.1.7 The *-form

We view the *-form as a slight generalisation of the system described in the previous subsection. The idea is to define a * column by a command of the form:

```

\newcolumnntype{*}[2]{%
  \count@=#1\ifnum\count@>0
    \advance\count@ by -1 #2*{\count@}{#2}\fi}

```

`\NC@rewrite@*` This does not work however as `\newcolumnntype` takes great care not to expand anything in the preamble, and so the `\if` is never expanded. `\newcolumnntype` sets up various other parts of the rewrite correctly though so we can define:

```
6272 \newcolumnntype{*}[2]{}
```

Now we must correct the definition of `\NC@rewrite@*`. The following is probably more efficient than a direct translation of the idea sketched above, we do not need to put a `*` in the preamble and call the rewrite recursively, we can just put `#1` copies of `#2` into `\@temptokena`. (Nested `*` forms will be expanded when the whole rewrite list is expanded again, see `\@mkpream`)

```
6273 \long\@namedef{NC@rewrite@*}#1#2{%
```

Store the number.

```
6274   \count@#1\relax
```

Put `#1` copies of `#2` in the token register.

```
6275   \loop
```

```
6276   \ifnum\count@>\z@
```

```
6277     \advance\count@\m@ne
```

```
6278     \@temptokena\expandafter{\the\@temptokena#2}%
```

```
6279   \repeat
```

`\NC@do` will ensure that `\NC@find` is `\let` equal to `\NC@find@*`.

```
6280   \NC@find}
```

```
6281
```

17.2 Getting the spacing around rules right

The Companion [GMS94, p.137] suggests two additional commands to control the alignments in case of tabulars with horizontal lines. They are now added to this package.

`\extratabsurround` The extra space around a table when `\firstline` or `\lastline` are used.

```
6282 \newlength{\extratabsurround}
```

```
6283 \setlength{\extratabsurround}{2pt}
```

`\backup@length` This register will be used internally by `\firstline` and `\lastline`.

```
6284 \newlength{\backup@length}
```

`\firstline` This code can probably be improved but for the moment it should serve.

We start by producing a single tabular row without any visible content that will produce the external reference point in case `[t]` is used.

```
6285 \newcommand{\firstline}{%
```

```
6286   \multicolumn1c{%
```

Within this row we calculate `\backup@length` to be the height plus depth of a standard line. In addition we have to add the width of the `\hline`, something that was forgotten in the original definition.

```
6287 \global\backup@length\ht\@arstrutbox
6288 \global\advance\backup@length\dp\@arstrutbox
6289 \global\advance\backup@length\arrayrulewidth
```

Finally we do want to make the height of this first line be a bit larger than usual, for this we place the standard array strut into it but raised by

```
\extratabsurround
6290 \raise\extratabsurround\copy\@arstrutbox
```

Having done all this we end the line and back up by the value of `\backup@length` and then finally place our `\hline`. This should place the line exactly at the right place but keep the reference point of the whole tabular at the baseline of the first row.

```
6291 }\\[-\backup@length]\hline
6292 }
```

`\lasthline` For `\lasthline` the situation is even worse and I got it completely wrong initially.

The problem in this case is that if the optional argument `[b]` is used we do want the reference point of the tabular be at the baseline of the last row but at the same time do want the the depth of this last line increased by `\extratabsurround` without changing the placement `\hline`.

We start by placing the rule followed by an invisible row.

```
6293 \newcommand{\lasthline}{\hline\multicolumn1c{%
```

We now calculate `\backup@length` to be the height and depth of two lines plus the width of the rule.

```
6294 \global\backup@length2\ht\@arstrutbox
6295 \global\advance\backup@length2\dp\@arstrutbox
6296 \global\advance\backup@length\arrayrulewidth
```

This will bring us back to the baseline of the second last row:

```
6297 }\\[-\backup@length]%
```

Thus if we now add another invisible row the reference point of that row will be at the baseline of the last row (and will be the reference for the whole tabular).

Since this row is invisible we can enlarge its depth by the desired amount.

```
6298 \multicolumn1c{%
6299 \lower\extratabsurround\copy\@arstrutbox
6300 }%
6301 }
```

Beside a larger functionality `array.sty` has one important difference to the standard `tabular` and `array` environments: horizontal and vertical rules make a table larger or wider, e.g., `\doublerulesep` really denotes the space between two rules and isn't measured from the middle of the rules.

`\@xhline` For vertical rules this is implemented by the definitions above, for horizontal rules we have to take out the backspace.

```

6302 \CheckCommand*\@xhline{\ifx\reserved@a\hline
6303         \vskip\doublerulesep
6304         \vskip-\arrayrulewidth
6305         \fi
6306         \ifnum0='{ \fi}}
6307 \renewcommand*\@xhline{\ifx\reserved@a\hline
6308         \vskip\doublerulesep
6309         \fi
6310         \ifnum0='{ \fi}}
6311
6312 %%%%%%%%% end Array package code %%%%%%%%%
6313

```

17.3 D column specifiers

The description and code are essentially copied from the `dcolumn` package [Car01].

```

6314 %%%%%%%%% Dcolumn package code %%%%%%%%%
6315 %%%%%%%%% With acknowledgements to David Carlisle %%%%%%%%%
6316

```

The `dcolumn` package [Car01] defines `D` to be a column specifier with three arguments.

`D{⟨sep.tex⟩}{⟨sep.dvi⟩}{⟨decimal places⟩}` for columns which are to be aligned on a ‘decimal point’.

The basic ideas behind these macros have been explained earlier. However they use three tricks which may be useful in other contexts.

- The separator is surrounded in extra `{ }`, so that it is set with `\mathord` spacing, otherwise, for instance a ‘,’ would have extra space after it.
- The separator is not given its special definition by making it active, as this would not work for an entry such as `& .5 &`, as the first token of an alignment entry is read *before* the preamble part, incase it is an `\omit`, in which case the preamble is to be omitted. Instead we switch the mathcode to (hex) 8000, which makes the token act as if it were active.
- Although `\mathcode'."8000` makes `.` act as if it were active, it is still not allowed in constructions such as `\def.{}`, even in math-mode, so we have to construct an active version of the separator, this is done by making it the uppercase of `~`, and then using the construct `\uppercase{\def~}{⟨definition⟩}`.
Note that the `⟨definition⟩` is not uppercased, so the definition can refer to the standard, non-active use of the separator.

`\DC@` Set up uppercase tables as required, and then grab the first part of the numerical argument into `\count@`.

```
6317 \def\DC@#1#2#3{%
6318   \uccode'\~='#1\relax
6319   \m@th
6320   \afterassignment\DC@x\count@#3\relax{#1}{#2}}
```

`\DC@x` If `\count@` is negative, centre on the decimal point. If it is positive either #1 will be empty in which case pad out decimal part to the number of digits specified by `\count@` or it is none empty in which case `\count@` contains the number of digits to the left of the point, and #1 contains a junk token (probably `.`) followed by the number of digits to the right of the point. In either of these latter cases, `\DC@right` is used.

```
6321 \def\DC@x#1\relax#2#3{%
6322   \ifnum\z@>\count@
6323     \expandafter\DC@centre
6324   \else
6325     \expandafter\DC@right
6326   \fi
6327   {#2}{#3}{#1}}
```

`\DC@centre` If centering on the decimal point, just need to box up the two halves.

```
6328 \def\DC@centre#1#2#3{%
6329   \let\DC@end\DC@endcentre
6330   \uppercase{\def~}{\egroup\setbox\tw@=\hbox\bgroup${#2}}}%
6331   \setbox\tw@=\hbox{${\phantom{#{2}}}}}%
6332   \setbox\z@=\hbox\bgroup$\mathcode'\#1="8000 }%}
```

`\DC@endcentre` and then pad out the smaller of the two boxes so there is the same amount of stuff either side of the point.

```
6333 \def\DC@endcentre{${\egroup}%
6334   \ifdim \wd\z@>\wd\tw@
6335     \setbox\tw@=\hbox to\wd\z@{\unhbox\tw@\hfill}%
6336   \else
6337     \setbox\z@=\hbox to\wd\tw@{\hfill\unhbox\z@}\fi
6338   \box\z@\box\tw@}
```

`\DC@right` This deals with both the cases where a specified number of decimal places is given.

```
6339 \def\DC@right#1#2#3{%
6340   \ifx\relax#3\relax
```

If #3 is empty, add `\hfill` to right align the column, and Just set `\DC@rl` to begin a group, so nothing fancy is done with the whole number part.

```
6341   \hfill
6342   \let\DC@rl\bgroup
6343   \else
```


Otherwise set `\DC@rl` so that the whole number part is put in a box `\count@` times as wide as a digit. In order to share code with the other branch, then move `#3` (the number of decimal places) into `\count@` throwing away the ‘.’ from the user syntax.

```
6344 \edef\DC@rl{to\the\count@\dimen@ii\bgroup\hss\hfill}%
6345 \count@\@gobble#3\relax
6346 \fi
6347 \let\DC@end\DC@endright
```

Box 2 contains the decimal part, set to `\dimen@` which is calculated below to be `\count@` times the width of a digit, plus the width of the ‘decimal point’.

```
6348 \uppercase{\def~}{\egroup\setbox\tw@\hbox to\dimen@\bgroup${#2}}%
6349 \setbox\z@\hbox{$1$}\dimen@ii\wd\z@
6350 \dimen@\count@\dimen@ii
6351 \setbox\z@\hbox{$#{#2}$}\advance\dimen@\wd\z@
6352 \setbox\tw@\hbox to\dimen@{~}%

```

Box 0 contains the whole number part, either just at its natural size for right aligned columns, or set to (the old value of) `\count@` times the width of a digit. `\DC@rl` defined above determines the two cases.

```
6353 \setbox\z@\hbox\DC@rl$\mathcode‘#1="8000 }%$
```

`\DC@endright` Just finish off the second box, and then put out both boxes.

```
6354 \def\DC@endright{${\hfil\egroup\box\z@\box\tw@}%$
```

- D The user interface, define the `D` column to take three arguments. For special purposes, you may need to directly access `\DC@` rather than the `D` column, eg to get a bold version you could use

```
\newcolumnntype{E}[3]{>{\boldmath\DC@{#1}{#2}{#3}}c<{\DC@end}}
6355 \newcolumnntype{D}[3]{>{\DC@{#1}{#2}{#3}}c<{\DC@end}}
6356
6357 %%%%%%%%% end Dcolumn package code %%%%%%%%%
6358
```

17.4 Support for delimiters

The description and code are essentially from the `delarray` package [Car94], v1.01 1994/03/14.

```
6359 %%%%%%%%% Delarray package code %%%%%%%%%
6360 %%%%%%%%% With acknowledgements to David Carlisle %%%%%%%%%
6361
```

The array syntax is extended by supporting the notation of delimiters. To this end we extend the array parsing mechanism to include a hook which can be used by this (or another) package to do some additional parsing.

`\@array` This macro tests for an optional delimiter before the left brace of the main preamble argument. If there is no delimiter, `\@arrayleft` and `\@arrayright` are made a no-ops, and `\@array` is called with the positional argument. Otherwise call `\@delarray`.

```
6362 \def\@array[#1]{\ifnextchar\bgroup
6363   {\let\@arrayleft\relax\let\@arrayright\relax\@array[#1]}%
6364   {\@delarray[#1]}}
```

`\@delarray` We now know that we have an `array` (or `tabular`) with delimiters.

```
6365 \def\@delarray[#1]#2#3#4{%
```

The following line is completely redundant but it does catch errors involving delimiters before the processing of the alignment begins. A common error is likely to be omitting the ‘.’ in a `\cases`-type construction. This causes the first token of the alignment to be gobbled, possibly causing lots of spurious errors before the cause of the error, the missing delimiter, is discovered as `\@arrayright` puts the alignment and the delimiters together.

```
6366   \setbox\z@\hbox{${\left#2\right#4$}}%
```

In the case of a ‘c’ argument we do not need to rebox the alignment, so we can define `\@arrayleft` and `\@arrayright` just to insert the delimiters.

```
6367   \if#1c\def\@arrayleft{\left#2}\def\@arrayright{\right#4}%
```

Otherwise we (should) have a [t] or [b] argument, so first we store the alignment, without delimiters in `box0`.

```
6368   \else\def\@arrayleft{\setbox\z@}%
```

Then after the alignment is finished:

```
6369   \def\@arrayright{%
```

Calculate the amount the box needs to be lowered (this will be negative in the case of [b]). A little bit of arithmetic cf. *The TeXbook*, Appendix G, rule 8. We calculate the amount this way, rather than just taking the difference between the depth of `box0` and the depth of the box defined below, as the depth of that box may be affected by the delimiters if `\delimitershortfall` or `\delimiterfactor` have non-standard values.

```
6370     \dimen@=\dp\z@
6371     \advance\dimen@-\ht\z@
6372     \divide \dimen@ by \tw@
6373     \advance\dimen@ by\fontdimen22 \textfont\tw@
```

Now lower the alignment and the delimiters into place.

```
6374     \lower\dimen@\hbox{${\left#2\center{\unvbox\z@}\right#4$}}%
```

End the `\if#1c`

```
6375   \fi
```

Now that we have defined `\@arrayleft` and `\@arrayright`, call `\@array`.

```
6376   \@array[#1]{#3}}
```

```
6377 %%%%%%%%% end Delarray package code %%%%%%%%%
```

```
6378
```

17.5 The tabularx environment

The code and description are essentially from David Carlisle's `tabularx` package [Car99], v2.07 1999/01/07.

```
6379 %%%%%%%%% Tabularx package code %%%%%%%%%
6380 %%%%%%%%% With acknowledgements to David Carlisle %%%%%%%%%
6381
```

The `tabularx` package [Car99] implements a version of the `tabular` environment in which the widths of certain columns are calculated so that the table is a specified width. The columns that may stretch are marked by an X in the preamble argument.

First some registers etc. that we need.

```
6382 \newdimen\TX@col@width
6383 \newdimen\TX@old@table
6384 \newdimen\TX@old@col
6385 \newdimen\TX@target
6386 \newdimen\TX@delta
6387 \newcount\TX@cols
6388 \newif\ifTX@
```

Now a trick to get the body of an environment into a token register, without doing any expansion. This does not do any real checking of nested environments, so if you should need to nest one `tabularx` inside another, the inner one must be surrounded by `{ }`.

`\tabularx` Just save the width specification separately, then clear the token register `\toks@`. Finally call `\TX@get@body` to begin saving the body of the table. The `{\ifnum0='}\fi` allows `tabularx` to appear inside a `\halign`.¹⁵ This mechanism of grabbing an environment body does have the disadvantage (shared with the AMS alignment environments) that you can not make extension environments by code such as

```
\newenvironment{foo}{\begin{tabularx}{XX}}{\end{tabularx}}
```

as the code is looking for a literal string `\end{tabularx}` to stop scanning. One may avoid this problem by using `\tabularx` and `\endtabularx` directly in the definition:

```
\newenvironment{foo}{\tabularx{XX}}{\endtabularx}
```

The scanner now looks for the end of the current environment (`foo` in this example.) There are some restrictions on this usage, the principal one being that `\endtabularx` is the *first* token of the 'end code' of the environment.

```
6389 \def\tabularx#1{%
```

¹⁵This adds an extra level of grouping, which is not really needed. Instead, I could use `\iffalse{\fi\ifnum0='}\fi` here, and `\ifnum0='{ }\fi` below, however the code here would then have to be moved after the first line, because of the footnote to page 386 of *The TeXbook*, and I do not think I should be writing code that is so obscure as to be documented in a footnote in an appendix called "Dirty Tricks"!

Allow `\tabularx \endtabularx` (but not `\begin{tabularx} \end{tabularx}`) to be used in `\newenvironment` definitions.

```
6390 \edef\TX@{\@currentvir}%
6391   {\ifnum0='}\fi
      \setlength lets you use the calc package so you could use a width of, say
      (\textwidth-12pt)/2.
6392   \setlength\TX@target{#1}%
6393   \TX@typeout{Target width: #1 = \the\TX@target.}%
6394   \toks@{\TX@get@body}
```

`\endtabularx` This does not do very much...

```
6395 \let\endtabularx\relax
```

`\TX@get@body` Place all tokens as far as the first `\end` into a token register. Then call `\TX@find@end` to see if we are at `\end{tabularx}`.

```
6396 \long\def\TX@get@body#1\end
6397   {\toks@\expandafter{\the\toks@#1}\TX@find@end}
```

`\TX@find@end` If we are at `\end{tabularx}`, call `\TX@endtabularx`, otherwise add `\end{...}` to the register, and call `\TX@get@body` again.

```
6398 \def\TX@find@end#1{%
6399   \def\@tempa{#1}%
6400   \ifx\@tempa\TX@\expandafter\TX@endtabularx
6401   \else\toks@\expandafter
6402     {\the\toks@\end{#1}}\expandafter\TX@get@body\fi}
```

`\TX@` The string `tabularx` as a macro for testing with `\ifx`.

```
6403 \def\TX@{tabularx}
```

Now that all the parts of the table specification are stored in registers, we can begin the work of setting the table.

The algorithm for finding the correct column widths is as follows. Firstly set the table with each X column the width of the final table. Assuming that there is at least one X column, this will produce a table that is too wide. Divide the excess width by the number of X columns, and reduce the column width by this amount. Reset the table. If the table is not now the correct width, a `\multicolumn` entry must be ‘hiding’ one of the X columns, and so there is one less X column affecting the width of the table. So we reduce by 1 the number of X columns and repeat the process.

`\TX@endtabularx` Although I have tried to make `tabularx` look like an environment, it is in fact a command, all the work is done by this macro.

```
6404 \def\TX@endtabularx{%
      Define the X column, with an internal version of the \newcolumn type command.
      The \expandafter commands enable \NC@newcol to get the expansion of
      \tabularxcolumn{\TX@col@width} as its argument. This will be the definition
      of an X column.
6405   \expandafter\TX@newcol\expandafter{\tabularxcolumn{\TX@col@width}}%
```

Initialise the column width, and the number of X columns. The number of X columns is set to one, which means that the initial count will be one too high, but this value is decremented before it is used in the main loop.

```
6406 \let\verb\TX@verb
```

Save the values of all L^AT_EX counters; the list `\cl@ckpt` contains the names of all the L^AT_EX counters that have been defined so far. We expand `\setcounter` at this point, as it results in fewer tokens being stored in `\TX@ckpt`, but the actual resetting of the counters occurs when `\TX@ckpt` is expanded after each trial run. Actually, use something equivalent to the expansion of the original definition of `\setcounter`, so that `tabularx` works in conjunction with `calc.sty`.

```
6407 \def\@elt##1{\global\value{##1}\the\value{##1}\relax}%
6408 \edef\TX@ckpt{\cl@ckpt}%
6409 \let\@elt\relax
6410 \TX@old@table\maxdimen
6411 \TX@col@width\TX@target
6412 \global\TX@cols\@ne
```

Typeout some headings (unless this is disabled).

```
6413 \TX@typeout@
6414 {\@spaces Table Width\@spaces Column Width\@spaces X Columns}%
```

First attempt. Modify the X definition to count X columns.

```
6415 \TX@trial{\def\NC@rewrite@X{%
6416 \global\advance\TX@cols\@ne\NC@find p{\TX@col@width}}}%
```

Repeatedly decrease column width until table is the correct width, or stops shrinking, or the columns become too narrow. If there are no multicolumn entries, this will only take one attempt.

```
6417 \loop
6418 \TX@arith
6419 \ifTX@
6420 \TX@trial{}%
6421 \repeat
```

One last time, with warnings back on (see appendix D) use `tabular*` to put it in a box of the right size, in case the algorithm failed to find the correct size.

Locally make `\footnotetext` save its argument in a token register. `\toks@` contains the preamble specification, and possible optional argument, as well as the table body.

```
6422 {\let\@footnotetext\TX@fntext\let\@xfootnotenext\TX@xftntext
6423 \csname tabular*\expandafter\endcsname\expandafter\TX@target
6424 \the\toks@
6425 \csname endtabular*\endcsname}%
```

Now the alignment is finished, and the `}` has restored the original meaning of `\@footnotetext` expand the register `\TX@ftn` which will execute a series of `\footnotetext[<num>]{<note>}` commands. We need to be careful about clearing the register as we may be inside a nested `tabularx`.

```
6426 \global\TX@ftn\expandafter{\expandafter}\the\TX@ftn
```

Now finish off the `tabularx` environment. Note that we need `\end{tabularx}` here as the `\end{tabularx}` in the user's file is never expanded. Now use `\TX@` rather than `tabularx`.

We also need to finish off the group started by `{\ifnum0='}\fi` in the macro `\tabularx`.

```
6427 \ifnum0='{ \fi}%
6428 \expandafter\end\expandafter{\TX@}
```

`\TX@arith` Calculate the column width for the next try, setting the flag `\ifTX@` to false if the loop should be aborted.

```
6429 \def\TX@arith{%
6430 \TX@false
6431 \ifdim\TX@old@table=\wd\@tempboxa
```

If we have reduced the column width, but the table width has not changed, we stop the loop, and output the table (which will cause an over-full alignment) with the previous value of `\TX@col@width`.

```
6432 \TX@col@width\TX@old@col
6433 \TX@typeout@{Reached minimum width, backing up.}%
6434 \else
```

Otherwise calculate the amount by which the current table is too wide.

```
6435 \dimen@\wd\@tempboxa
6436 \advance\dimen@ -\TX@target
6437 \ifdim\dimen@<\TX@delta
```

If this amount is less than `\TX@delta`, stop. (`\TX@delta` should be non-zero otherwise we may miss the target due to rounding error.)

```
6438 \TX@typeout@{Reached target.}%
6439 \else
```

Reduce the number of effective X columns by one. (Checking that we do not get 0, as this would produce an error later.) Then divide excess width by the number of effective columns, and calculate the new column width. Temporarily store this value (times -1) in `\dimen@`.

```
6440 \ifnum\TX@cols>\@ne
6441 \advance\TX@cols\m@ne
6442 \fi
6443 \divide\dimen@\TX@cols
6444 \advance\dimen@ -\TX@col@width
6445 \ifdim \dimen@ >\z@
```

If the new width would be too narrow, abort the loop. At the moment too narrow, means less than 0pt!

If the loop was aborted here and the X columns were left with the width of the previous run it may make the table far too wide as initial guesses are always too big. Force to `\TX@error@width` which defaults to be 1em.

```
6446 \@memwarn{X Columns too narrow (table too wide)\MessageBreak}%
6447 \TX@col@width\TX@error@width\relax
6448 \else
```

Otherwise save the old settings, and set the new column width. Set the flag to true so that the table will be set, and the loop will be executed again.

```

6449      \TX@old@col\TX@col@width
6450      \TX@old@table\wd\@tempboxa
6451      \TX@col@width-\dimen@
6452      \TX@true
6453      \fi
6454      \fi
6455      \fi}

```

`\TX@error@width` If the calculated width is negative, use this instead.

```
6456 \def\TX@error@width{1em}
```

`\TX@delta` Accept a table that is within `\hfuzz` of the correct width.

```
6457 \TX@delta\hfuzz
```

Initialise the X column. The definition can be empty here, as it is set for each `tabularx` environment.

```
6458 \newcolumnntype{X}{}
```

`\tabularxcolumn` The default definition of X is `p{#1}`.

```
6459 \def\tabularxcolumn#1{p{#1}}
```

`\TX@newcol` A little macro just used to cut down the number of `\expandafter` commands needed.

```
6460 \def\TX@newcol{\newcol@{X}[0]}
```

`\TX@trial` Make a test run.

```

6461 \def\TX@trial#1{%
6462   \setbox\@tempboxa\hbox{%

```

Any extra commands. This is used on the first run to count the number of X columns.

```
6463   #1\relax
```

Make `\footnotetext` gobble its arguments. Also locally clear `\TX@vwarn` so that the warning is generated by the final run, and does not appear in the middle of the table if `\tracingtabularx`.

```

6464   \let\@footnotetext\TX@trial@ftn
6465   \let\TX@vwarn\@empty

```

Do not nest `tabularx` environments during trial runs. This would waste time, and the global setting of `\TX@cols` would break the algorithm.

```

6466   \expandafter\let\expandafter\tabularx\csname tabular*\endcsname
6467   \expandafter\let\expandafter\endtabularx\csname endtabular*\endcsname

```

Dissable `\writes` during a trial run. This trick is from *The T_EXbook*.¹⁶

```

6468   \def\write{\begingroup
6469     \def\let{\afterassignment\endgroup\toks@}%
6470     \afterassignment\let\count@}%

```

¹⁶Actually the trick does not work correctly, so it has been changed.

Turn off warnings (see appendix D). Also prevent them being turned back on by setting the parameter names to be registers.

```
6471 \hbadness\@M
6472 \hfuzz\maxdimen
6473 \let\hbadness\@tempcnta
6474 \let\hfuzz\@tempdima
```

Make the table, and finish the hbox. `\toks@` contains the preamble specification, and possible optional argument, as well as the table body.

```
6475 \expandafter\table\the\toks@
6476 \endtable}%
```

Reset all L^AT_EX counters, by executing `\TX@ckpt`.

```
6477 \TX@ckpt
```

Print some statistics.

```
6478 \TX@typeout{\@spaces
6479 \expandafter\TX@align
6480 \the\wd\@tempboxa\space\space\space\space\space\@@
6481 \expandafter\TX@align
6482 \the\TX@col@width\space\space\space\space\space\@@
6483 \@spaces\the\TX@cols}}
```

`\TX@align` Macro to improve the printing of the tracing info.

```
6484 \def\TX@align#1.#2#3#4#5#6#7#8#9\@@{%
6485 \ifnum#1<10 \space\fi
6486 \ifnum#1<100 \space\fi
6487 \ifnum#1<\@m\space\fi
6488 \ifnum#1<\@M\space\fi
6489 #1.#2#3#4#5#6#7#8\space\space}
```

`\arraybackslash` `\` hack.

```
6490 \def\arraybackslash{\let\\\@arraycr}
```

`\tracingtabularx` Print statistics on column and table widths.

```
6491 \def\tracingtabularx{%
6492 \def\TX@typeout{\ClassWarningNoLine{memoir}}}%
6493 \def\TX@typeout@#1{\typeout{(tabularx) #1}}}
```

`\TX@typeout` The default is to be to be quiet

```
6494 \let\TX@typeout\@gobble
6495 \let\TX@typeout@\@gobble
```

`\TX@ftn` A token register for saving footnote texts.

```
6496 \newtoks\TX@ftn
```

`\TX@ftntext` Inside the alignment just save up the footnote text in a token register.

```
\TX@xftntext 6497 \long\def\TX@ftntext#1{%
6498 \edef\@tempa{\the\TX@ftn\noexpand\footnotetext
6499 [\the\c@name c@\@mpfn\endc@name}}%
```



```

6500 \global\TX@ftn\expandafter{\@tempa{#1}}}%
6501 \long\def\TX@xftntext[#1]#2{%
6502 \global\TX@ftn\expandafter{\the\TX@ftn\footnotetext[#1]{#2}}}

```

`\TX@trial@ftn` On trial runs, gobble footnote texts.

```

6503 \long\def\TX@trial@ftn#1{}

```

This last section was added at Version 1.02. Previous versions documented the fact that `\verb` did not work inside `tabularx`, but that did not stop people using it! This usually put L^AT_EX into an irrecoverable error position, with error messages that did not mention the cause of the error. The ‘poor man’s `\verb`’ (and `\verb*`) defined here is based on page 382 of *The T_EXbook*. As explained there, doing verbatim this way means that spaces are not treated correctly, and so `\verb*` may well be useless, however I consider this section of code to be error-recovery, rather than a real implementation of verbatim.

The mechanism is quite general, and any macro which wants to allow a form of `\verb` to be used within its argument may `\let\verb=\TX@verb`. (Making sure to restore the real definition later!)

`\verb` and `\verb*` are subject to the following restrictions:

1. Spaces in the argument are not read verbatim, but may be skipped according to T_EX’s usual rules.
2. Spaces will be added to the output after control words, even if they were not present in the input.
3. Unless the argument is a single space, any trailing space, whether in the original argument, or added as in (2), will be omitted.
4. The argument must not end with `\`, so `\verb|\|` is not allowed, however, because of (3), `\verb|\ |` produces `\`.
5. The argument must be balanced with respect to `{` and `}`. So `\verb|{|` is not allowed.
6. A comment character like `%` will not appear verbatim. It will act as usual, commenting out the rest of the input line!
7. The combinations `?‘` and `!‘` will appear as `¿` and `¡` if the `cmtt` font is being used.

`\TX@verb` The internal definition of `\verb`. Spaces will be replaced by `~`, so for the star-form, `\let ~ be \verb*| |`, which we obtain as `\uppercase{*}`. Use `{\ifnum0=‘}\fi` rather than `\bgroup` to allow `&` to appear in the argument.

```

6504 {\uccode‘\*=\ %
6505 \uppercase{\gdef\TX@verb{%
6506 \leavevmode\null\TX@vwarn
6507 {\ifnum0=‘}\fi\ttfamily\let\\\ignorespaces
6508 \@ifstar{\let~*\TX@vb}{\TX@vb}}}

```

`\TX@vb` Get the ‘almost verbatim’ text using `\meaning`. The ‘!’ is added to the front of the user supplied text, to ensure that the whole argument does not consist of a single `{ }` group. `TeX` would strip the outer braces from such a group. The ‘!’ will be removed later.

Originally I followed Knuth, and had `\def\@tempa{##1}`, however this did not allow `#` to appear in the argument. So I changed this to to use a token register, and `\edef`. This allows `#` appear, but makes each one appear twice!, so later we loop through, replacing `##` by `#`.

```
6509 \def\TX@vb#1{\def\@tempa##1#1{\toks@{##1}\edef\@tempa{\the\toks@}%
6510      \expandafter\TX@v\meaning\@tempa\ \ \ifnum0={\fi}\@tempa!}}
```

`\TX@v` Strip the initial segment of the `\meaning`, including the ‘!’ added earlier.

```
6511 \def\TX@v#1!{\afterassignment\TX@vfirst\let\@tempa= }
```

As explained above we are going to replace `##` pairs by `#`. To do this we need non-special `#` tokens. Make `*` into a parameter token so that we can define macros with arguments. The normal meanings will be restored by the `\endgroup` later.

```
6512 \begingroup
6513 \catcode'\*=\catcode'\#
6514 \catcode'\#=12
```

`\TX@vfirst` As a special case, prevent the first character from being dropped. This makes `\verb*|` produce `␣`. Then call `\TX@v@`. This is slightly tricky as I have to ensure that an actual `#` rather than a command `\let to #` is passed on if the first character is `#`.

```
6515 \gdef\TX@vfirst{%
6516   \if\@tempa#%
6517     \def\@tempb{\TX@v@#}%
6518   \else
6519     \let\@tempb\TX@v@
6520     \if\@tempa\space~\else\@tempa\fi
6521   \fi
6522   \@tempb}
```

`\TX@v@` Loop through the `\meaning`, replacing all spaces by `~`. If the last character is a space it is dropped, so that `\verb*|\LaTeX|` produces `\LaTeX` not `\LaTeX␣`. The rewritten tokens are then further processed to replace `##` pairs.

```
6523 \gdef\TX@v@*1 *2{%
6524   \TX@v@hash*1##\relax\if*2\\\else~\expandafter\TX@v@\fi*2}
```

`\TX@v@hash` The inner loop, replacing `##` by `#`.

```
6525 \gdef\TX@v@hash*1##*2{*1\ifx*2\relax\else#\expandafter\TX@v@hash\fi*2}
```

As promised, we now restore the normal meanings of `#` and `*`.

```
6526 \endgroup
```

`\TX@vwarn` Warn the user the first time this `\verb` is used.

```
6527 \def\TX@vwarn{%
6528   \@warning{\noexpand\verb may be unreliable inside tabularx}%
6529   \global\let\TX@vwarn\@empty}
6530
```

```
6531 %%%%%%%%% end Tabularx package code %%%%%%%%%
6532
```

At this point we had better stop anyone trying to load the several tabular-related packages (as effectively they are already loaded).

```
6533 %\@memfakeusepackage{array}
6534 %\@memfakeusepackage{dcolumn}
6535 %\@memfakeusepackage{delarray}
6536 %\@memfakeusepackage{tabularx}
6537
```

17.6 Fear's rules

Simon Fear disapproves of the default L^AT_EX table rules and wrote the `booktabs` package [Fea03] to provide better horizontal rules. Like many typographers, he abhors vertical rules.

`\bktabrule` The rules in this section are based on `\hrule \@height#1` but this does not work for continuous tabulars. Provide a version of this that can be `\let` when necessary.

```
6538 \newcommand*{\bktabrule}[1]{%
6539   \hrule \@height#1}
6540
```

The following is taken almost verbatim from the `booktabs` package, version 1.618, March 2003.

```
6541 %%%%%%%%% Booktabs package code %%%%%%%%%
6542 %%%%%%%%% slightly modified by PRW %%%%%%%%%
6543 %%%%%%%%% by permission of Simon Fear %%%%%%%%%
6544
```

Danie Els is now (2005) the official maintainer of the `booktabs` package. He provided me with some updates to synchronize `memoir`, `booktabs` and support for `colortbl` with `booktabs`. I made the appropriate changes/extensions on 2005/09/03 as part of version 1.618 of `memoir`.

`\CT@arc@` This supports the `bookhands/colortbl` package combination.

```
6545 \AtBeginDocument{%
6546   \providecommand*{\CT@arc@}{}}
6547
```

```

\heavyrulewidth First we set up some new dimensions.
\lightrulewidth 6548 \newdimen\heavyrulewidth
\cmidrulewidth 6549 \newdimen\lightrulewidth
\belowrulesep 6550 \newdimen\cmidrulewidth
\belowbottomsep 6551 \newdimen\belowrulesep
\aboverulesep 6552 \newdimen\belowbottomsep
\abovetopsep 6553 \newdimen\aboverulesep
\cmidrulesep 6554 \newdimen\abovetopsep
\cmidrulekern 6555 \newdimen\cmidrulesep
\defaultaddspace 6556 \newdimen\cmidrulekern
6557 \newdimen\defaultaddspace
6558 \heavyrulewidth=.08em
6559 \lightrulewidth=.05em
6560 \cmidrulewidth=.03em
6561 \belowrulesep=.65ex
6562 \belowbottomsep=\z@
6563 \aboverulesep=.4ex
6564 \abovetopsep=\z@
6565 \cmidrulesep=\doublerulesep
6566 \cmidrulekern=.5em
6567 \defaultaddspace=.5em
6568

```

\@cmidla And some internal counters and other things of no interest to the end user:

```

\@cmidlb 6569 \newcount\@cmidla
\@aboverulesep 6570 \newcount\@cmidlb
\@belowrulesep 6571 \newdimen\@aboverulesep
\@thisruleclass 6572 \newdimen\@belowrulesep
\@lastruleclass 6573 \newcount\@thisruleclass
\@thisrulewidth 6574 \newcount\@lastruleclass
6575 \@lastruleclass=0
6576 \newdimen\@thisrulewidth
6577

```

which will be described as needed below.

```

\futurenonSPACElet Next we define a very useful macro (more-or-less straight from the The
\@BTfns lone TeXbook's Dirty Tricks chapter; documented there). Use \futurenonSPACElet
\@BTfns ltwo instead of \futurelet when looking for the next (non-space) token after a
\@BTfns lthree macro that has an argument. (After a macro without an argument, space is
ignored anyway, so \futurenonSPACElet wouldn't be needed.) This hack allows
users to type white space between successive rule commands.
6578 \def\futurenonSPACElet#1{\def\@BTcs{#1}%
6579 \afterassignment\@BTfns lone\let\nexttoken= }
6580 \def\@BTfns lone{\expandafter\futurelet\@BTcs\@BTfns ltwo}
6581 \def\@BTfns ltwo{\expandafter\ifx\@BTcs\@sptoken\let\next=\@BTfns lthree
6582 \else\let\next=\nexttoken\fi \next}
6583 \def\@BTfns lthree{\afterassignment\@BTfns lone\let\next= }
6584

```

17.6.1 Full width rules

When we are not in a `longtable` environment, we can simply implement the full width rules as a `\hrule` in a `\noalign{}` group. But within a `longtable`, the rule has to be drawn like a `\cmidrule{1- $\LT@cols$ }` (the rationale for this is explained in the `longtable` documentation).

In order to allow for both, all the rule macros have to open a `\noalign` group immediately, while they work out whether they have been called within a `longtable`; if you don't do this, \TeX 's underlying `\halign` process gets hiccups. I use \LaTeX 's dirty trick (`\ifnum=0'`) to fool the parser that the bracket count is OK. The bracket really gets closed after all the skipping at the end of the `\@BTendrule` macro.

The class 1 rules, and `\specialrule`, really only differ in the defaults for space above and below, and the width, passed to a common routine, `\@BTrule`, described below. The spaces, `\@aboverulesep` and `\@belowrulesep`, are set within the `\noalign` group, so are inherited by `\@BTrule`. Similarly, `\@BTrule` knows as much as it needs to about the routine that called it by examining the inherited `\@thisruleclass`. The optional width argument is parsed by `\@BTrule` after being set to default if absent.

```

\toprule
\midrule 6585 \def\toprule{\noalign{\ifnum0='}\fi
\bottomrule 6586 \@aboverulesep=\abovetopsep
\specialrule 6587 \global\@belowrulesep=\belowrulesep
6588 \global\@thisruleclass=\@ne
6589 \@ifnextchar[{\@BTrule}{\@BTrule[\heavyrulewidth]}}
6590
6591 \def\midrule{\noalign{\ifnum0='}\fi
6592 \@aboverulesep=\aboverulesep
6593 \global\@belowrulesep=\belowrulesep
6594 \global\@thisruleclass=\@ne
6595 \@ifnextchar[{\@BTrule}{\@BTrule[\lightrulewidth]}}
6596
6597 \def\bottomrule{\noalign{\ifnum0='}\fi
6598 \@aboverulesep=\aboverulesep
6599 \global\@belowrulesep=\belowbottomsep
6600 \global\@thisruleclass=\@ne
6601 \@ifnextchar[{\@BTrule}{\@BTrule[\heavyrulewidth]}}
6602
6603 \def\specialrule#1#2#3{\noalign{\ifnum0='}\fi
6604 \@aboverulesep=#2\global\@belowrulesep=#3\global\@thisruleclass=\tw@
6605 \@BTrule[#1]}
6606

```

`\addlinespace` An `\addlinespace` is essentially a zero-width rule with zero space above and argument (or default) space below. But because the rule is not actually drawn, but is just a `\vskip`, there is no need to check if we're in a `longtable`, so we don't need to call `\@BTrule` as for 'real' rules. But we do share the `\@BTendrule`

lookahead and flagsetting code (described below), and the `\vskip` is done there.

```
6607 \def\addlinespace{\noalign{\ifnum0='}\fi
6608 \ifnextchar[{\@addspace}{\@addspace[\defaultaddspace]}}
6609 \def\@addspace[#1]{\global\@belowrulesep=#1\global\@thisruleclass=\tw@
6610 \futurelet\@tempa\@BTendrule}
6611
```

`\@BTrule` All the rules (except `\addlinespace`) share this code.

```
6612 \def\@BTrule[#1]{%
6613 \global\@thisrulewidth=#1\relax
```

Save the width argument (if the user didn't give one, then the calling routine will have called `\@BTrule` with the default) in a global variable for later use when drawing the rule.

```
6614 \ifnum\@thisruleclass=\tw@\vskip\@aboverulesep}else
```

Specialrules always insert specified space above. (Note: `addlinespaces` don't come here).

```
6615 \ifnum\@lastruleclass=\z@\vskip\@aboverulesep}else
6616 \ifnum\@lastruleclass=\ne\vskip\doublerulesep\fi\fi\fi
```

After text (last rule class 0), precede the rule by `\aboverulesep`; but if immediately after a previous rule, insert a `\doublerulesep`.

Now we work out, by a very nasty hack, if we're within a `longtable`. It's easy if `\longtable` isn't even defined: then we can't be. But it is not enough just to check if `longtable` is loaded — we might be within an ordinary table rather than a `longtable`. So we look to see if `\hline` has been re-defined from its `LATEX` definition to be the same as `\LT. (Longtable currently does this redefinition when it opens a longtable environment, but not globally, so it is cleared it when the environment closes.) Another package could potentially do this! And longtable might change the way it implements this! So, it is not entirely safe, but I have found no better way so far.`

We set up `\@BTswitch` to call `\@BTnormal` or `\@BLTrule`, as appropriate, then call it.

```
6617 \ifx\longtable\undefined\let\@BTswitch\@BTnormal}else
6618 \ifx\hline\LT

```

`\@BTnormal` This is when we're *not* within a `longtable`. We are already in a `\noalign` group, all we need do is draw an `\hrule` and gobble any trailing spaces, then call the closing routine with `\@tempa` set equal to the next token in the document.

```
6622 % \def\@BTnormal{\hrule
6623 % \@height \@thisrulewidth\futurenonspacel\@tempa\@BTendrule}
6624 \def\@BTnormal{%
6625 %% \bktabrule{\@thisrulewidth}
6626 {\CT@arc@\bktabrule{\@thisrulewidth}}}%
6627 \futurenonspacel\@tempa\@BTendrule}
```

`\@BLTrule` This is for full width rule within a `longtable`. First we check if a kerning argument has been used; if so let `\@@BLTrule` read it, else call `\@@BLTrule` with an empty string:

```
6628 \def\@BLTrule{\@ifnextchar({\@@BLTrule}{\@@BLTrule{}}}
```

`\@@BLTrule`

```
6629 \def\@@BLTrule(#1){\@setrulekerning{#1}%
6630 \global\@cmidlb\LT@cols
```

The `\@setrulekerning` routine parses the kerning argument tokens and sets global kerning widths accordingly (or to defaults, if user hasn't set them explicitly). The global assignment to `\@cmidlb` sets up the column count for the `\@cmidruleb` macro, which is shared with `cmidrules`.

```
6631 \ifnum0='{ \fi}%
```

Close the currently open `\noalign` group. Within a `longtable`, rules are all to be drawn as leaders within a text box that is `\LT@cols` columns wide.

```
6632 \@cmidruleb
```

Draw the rule. We share the `\@cmidruleb` code with ordinary `\cmidrules`.

```
6633 \noalign{\ifnum0='{ \fi
```

We have to open a new `noalign` immediately else `TeX` will start a new text box where we don't want one. Then, after gobbling any unwanted white space, we call the closing routine.

```
6634 \futurenonspacel\@tempa\@BTendrule}
```

```
6635
```

`\@BTendrule` We look one step ahead (token is in `\@tempa`) to see if another rule follows (shame on user!). If so, we set `\@lastruleclass` equal to `\@thisruleclass` (thus setting it up for the following rule). If there isn't a following rule, we clear `\@lastruleclass` (ie set it to zero), which isn't technically true since we have just drawn a rule, but sets it up correctly for the next rule encountered, which must be following some intervening text.

```
6636 \def\@BTendrule{%
6637   \ifx\@tempa\toprule\global\@lastruleclass=\@thisruleclass
6638   \else\ifx\@tempa\midrule\global\@lastruleclass=\@thisruleclass
6639   \else\ifx\@tempa\bottomrule\global\@lastruleclass=\@thisruleclass
6640   \else\ifx\@tempa\cmidrule\global\@lastruleclass=\@thisruleclass
6641   \else\ifx\@tempa\specialrule\global\@lastruleclass=\@thisruleclass
6642   \else\ifx\@tempa\addlinespace\global\@lastruleclass=\@thisruleclass
6643   \else\global\@lastruleclass=\z@\fi\fi\fi\fi\fi\fi
6644   \ifnum\@lastruleclass=\@ne\relax\else\vskip\@belowrulesep\fi
6645   \ifnum0='{ \fi}}
```

17.6.2 Special subrules

`\@setrulekerning` The following code parses the trimming arguments (if there are any) for `\cmidrule` or a `\BLTrule`. The rule will be trimmed left and right by

`\cmrkern@l` and `\cmrkern@l`, which are zero by default, set to `\cmidrulekern` by the plain (`lr`) arguments, or user set as in (`r{.5em}`). We parse token by token through the arguments. The tokens `r` and `l` cause `\cmrkern@r` or `\cmrkern@l` to be set to `\cmidrulekern`. There is no lookahead to see if a width is the next token; this strategy is efficient for the plain commands, while inefficient for the qualified commands, but more importantly it is much easier to program. Tokens `r` and `l` also set `\cmrswitch` so that if the next token turns out to be `{\wd}` then the kerning will be done on the side currently specified. I have been too lazy to program an error message should one encounter tokens other than `r`, `l` or `{\wd}`.

```

6646 \def\@setrulekerning#1{%
6647   \global\let\cmrkern@l\z@
6648   \global\let\cmrkern@r\z@
6649   \@tfor\@tempa :=#1\do
6650   {\def\@tempb{r}%
6651     \ifx\@tempa\@tempb
6652       \global\let\cmrkern@r\cmidrulekern
6653       \def\cmrswitch{\cmrkern@r}%
6654     \else
6655       \def\@tempb{l}%
6656       \ifx\@tempa\@tempb
6657         \global\let\cmrkern@l\cmidrulekern
6658         \def\cmrswitch{\cmrkern@l}%
6659       \else
6660         \global\expandafter\let\cmrswitch\@tempa
6661       \fi
6662     \fi}}
6663

```

`\cmidrule` The `\cmidrule` re-uses `\@lastruleclass` in an entirely different way from the
`\@cmidrule` full width rules. (Maybe I should have used a different flag; it seemed efficient at
`\@@cmidrule` the time ...). This is (left) set to one if you are in the middle of a row of
`\cmidrul`s, or starting a new one (with `\morecmidrul`s). Otherwise, when
`\@lastruleclass` is zero, we precede the rule with `\aboverulesep`.

```

6664 \def\cmidrule{\noalign{\ifnum0=}\fi
6665   \@ifnextchar[{\@cmidrule}{\@cmidrule[\cmidrulewidth]}}
6666 \def\@cmidrule[#1]{\@ifnextchar[{\@cmidrule[#1]}{\@cmidrule[#1]()}}
6667 \def\@@cmidrule[#1](#2)#3{\@@cmidrule[#3]{#1}{#2}}
6668

```

`\@@@cmidrule` The above is fiddling around to set defaults for missing optional arguments. We
also pass to `\@@@cmidrule` in a different order, namely
`=\a-b\width required\verbkerning commands` (this being the order in
which the arguments are actually processed):

```

6669 \def\@@@cmidrule[#1-#2]#3#4{\global\@cmidla#1\relax
6670   \global\advance\@cmidla\m@ne
6671   \ifnum\@cmidla>0\global\let\@gtempa\@cmidrulea\else
6672     \global\let\@gtempa\@cmidruleb\fi

```



```

6673 \global\@cmidlb#2\relax
6674 \global\advance\@cmidlb-\@cmidla

```

This has set up a switch (`\@gtempa`) to call the relevant routine, `\@cmidrulea` or `\@cmidruleb`, depending on whether we start from column one or not.

```

6675 \global\@thisrulewidth=#3

```

That is, set per default or given argument. Then parse any trimming arguments to set, globally, `\cmrkern@r` and `\cmrkern@l` accordingly:

```

6676 \setrulekerning{#4}

```

Now insert space above if needed, close the `\noalign`, then switch to appropriate rule drawing routine as determined above (`\let` to `\@gtempa`):

```

6677 \ifnum\@lastruleclass=\z@\vskip \aboverulesep\fi
6678 \ifnum0='{ \fi}\@gtempa

```

Having now drawn the rule, open another `\noalign`, and call the closing routine:

```

6679 \noalign{\ifnum0='{ \fi\futurenonspacinglet\@tempa\@xcmidrule}

```

`\@xcmidrule` In this closing routine, see if another `\cmidrule` follows; if so, backspace vertical so it will line up with the one you just drew, and setting `\@lastruleclass` to 1 will suppress adding space above the next. If a `\morecmidrules` follows, we add (positive) `\cmidrulesep` (and again set `\@lastruleclass` to one). Otherwise this is the last rule of the current group and we can just add `\belowrulesep`. Finally, we close the `\noalign`.

```

6680 \def\@xcmidrule{\ifx\@tempa\cmidrule\vskip-\@thisrulewidth
6681 \global\@lastruleclass=\@ne\else
6682 \ifx\@tempa\morecmidrules\vskip \cmidrulesep
6683 \global\@lastruleclass=\@ne\else
6684 \vskip \belowrulesep\global\@lastruleclass=\z@\fi\fi
6685 \ifnum0='{ \fi}}
6686

```

`\@cmidrulea` This code (called below) actually draws the rules. They are drawn as boxes in `\@cmidruleb` text, rather than in a `\noalign` group, which permits the left and right kerning.

```

6687 \def\@cmidrulea{%
6688 \multispan\@cmidla&\multispan\@cmidlb
6689 %% \unskip\hskip \cmrkern@l\leaders\bktabrule{\@thisrulewidth}\hfill
6690 \unskip\hskip \cmrkern@l{%
6691 \CT@arc@\leaders\bktabrule{\@thisrulewidth}\hfill}%
6692 \hskip \cmrkern@r\cr}
6693 \def\@cmidruleb{%
6694 \multispan\@cmidlb
6695 %% \unskip\hskip \cmrkern@l\leaders\bktabrule{\@thisrulewidth}\hfill
6696 \unskip\hskip \cmrkern@l{%
6697 \CT@arc@\leaders\bktabrule{\@thisrulewidth}\hfill}%
6698 \hskip \cmrkern@r\cr}
6699

```

`\morecmidrules` This is really a dummy command; all the work is done above within the `\cmidrule` routine. We look one step ahead there to see if a `\morecmidrules`

follows the current `\cmidrule`, and if so set the flag. Otherwise, `\morecmidrules` itself does nothing.

```
6700 \def\morecmidrules{\noalign{\relax}}
6701
6702 %%%%%%%%% end of Booktabs package code %%%%%%%%%
6703
```

Ensure that the real booktabs package can't get called.

```
6704 %\@memfakeusepackage{booktabs}
6705
```

17.7 Input files into tabulars, and etex

M.J. Williams (CTT *\noalign problem with \input and tabular in memoir class*, 28 Aug 2007) reported that using `\input` in a tabular resulted in errors. His example was:

```
...
\begin{tabular}{c|c} \hline
1 & 2 \\ \hline
\input{data} \hline
5 & 6 \\ \hline
\end{tabular}
```

where `data.tex` contains the single line (and no newline)

```
3 & 4 \\
```

Morten Høgholm noted that the problem was caused by `memoir`'s extension to `\input` and gave a solution that depends on processing via `etex` rather than `tex`. Here's Morten's fix. Quoting, in part:

`memoir` extends the file loading mechanism by providing `\AtBeginFile` and `\AtEndFile` whose arguments are executed as the names indicate. ... after inputting a file the storage bins are emptied. The problem [in your case] is that this emptying is an assignment and so starts a new tabular cell and `\hline` is only allowed after `\\` or other `\hlines`.

... the tabular environment wraps each tabular cell in additional groups so one can do an explicit test for which type of group we are in. If tested to be in an align group *and* in vertical mode it means TeX hasn't found something to start a new cell and then `\noalign` can be used.

```
6706 \ifetex
6707   \renewcommand*{\killm@matf}[1]{%
6708     \ifnum 6=\currentgroupstype
6709       \ifvmode
6710         \expandafter\expandafter\expandafter\@firstoftwo
6711         \expandafter\expandafter\expandafter\noalign
```

```

6712     \fi
6713     \fi
6714     \@firstofone
6715     {\@namelet{#1-m@mf}\relax
6716     \@namelet{#1-m@mfe}\relax
6717     }%
6718   }
6719 \fi
6720

```

17.8 Continuous tabulars

The standard `tabular` environment is wrapped inside a box (see `\@tabular`), which means that it cannot break across pages. Equivalent environments are provided which are not boxed and so can continue across page boundaries.

`\ctableleftskip` These skips are inserted at the left and right of continuous tabulars so that their
`\ctabrightskip` horizontal location can be specified.

```

6721 \newskip\ctableleftskip \ctableleftskip=\fill
6722 \newskip\ctabrightskip \ctabrightskip=\fill
6723

```

`\ctabular*` This starts off the `ctabular*` continuous tabular environment. The default
`\@ctabularstar` location is centered. NOTE: This is not for release.

```

6724 \expandafter\def\csname ctabular*\endcsname{%
6725   \@ifnextchar[ {\@ctabularstar}{\@ctabularstar[c]}}
6726 \def\@ctabularstar[#1]#2{\global\@curtab\@ne
6727   \ctableleftskip\fill
6728   \ctabrightskip\fill
6729   \if l#1% left
6730     \ctableleftskip\z@
6731   \else
6732     \if r#1% right
6733       \ctabrightskip\z@
6734     \fi
6735   \fi
6736   \setlength\dimen@{#2}%
6737   \xdef\@halignto{to\the\dimen@}\NC@tabular}

```

`\ctabular` `\ctabular` is similar to `\ctabular*` except that the width is known to be
`\hsize`.

```

6738 \newcommand*{\ctabular}[1][c]{\global\@curtab\@ne
6739   \ctableleftskip\fill
6740   \ctabrightskip\fill
6741   \if l#1% left
6742     \ctableleftskip\z@
6743   \else
6744     \if r#1% right
6745       \ctabrightskip\z@

```

```

6746     \fi
6747     \fi
6748     \gdef\@halignto{to\hsize}\NC@tabular}
6749
\NC@tabular \NC@tabular
6750 \newcommand*\NC@tabular}{%
6751     \par
6752     \addvspace{\topsep}
6753     \col@sep\tabcolsep
6754     \let\d@llarbegin\begin\group
6755     \let\d@llarend\end\group
6756     \NC@tabarray}
6757
\NC@calign
6758 \newcommand*\NC@calign{\everycr{}\tabskip\ctableftskip\halign}
6759
\NC@tabarray Effectively a modified version of \@array
6760 \newcommand*\NC@tabarray[1]{%
6761     \@tempdima \ht\strutbox
6762     \advance\@tempdima\extrarowheight
6763     \setbox \@arstrutbox \hbox{\vrule
6764         \@height \arraystretch \@tempdima
6765         \@depth \arraystretch \dp\strutbox
6766         \@width\z@}%
6767     \begin\group
6768     %% \mkpream{\@{\hspace{\@totalleftmargin}}\#1@{}}%
6769     \mkpream{\#1}%
6770     \xdef\@preamble{\NC@calign \@halignto
6771         \bgroup & \tabskip\z@
6772         \@arstrut
6773         \@preamble
6774         \tabskip\ctabrightskip
6775         \cr}%
6776     \end\group
6777     \let\@sharp ##\let\protect\relax
6778     \lineskip\z@
6779     \baselineskip\z@
6780     \let\\\@arraycr
6781     \let\tabularnewline\\%
6782     \let\par\@empty
6783     \ctabsetlines
6784     \@preamble
6785 }
6786
\endctabular End the environments.
\endctabular*

```

```

6787 \def\endctabular{%
6788   \crrc \egroup
6789   \gdef\@preamble{%
6790     \addvspace{\topsep}
6791     \noindent}
6792 \expandafter\let\csname endctabular*\endcsname=\endctabular
6793

```

17.8.1 Horizontal lines

The standard `\hline` can produce odd results when used with continuous tabulars, but `\hhline` or `\cline` are OK. I want a version of `\hline` that works with continuous tabulars and has an adjustable width (i.e., thickness). This will be based on a modified `\cline`

For reference, here are the essentials of the kernel's definitions.

```

\def\hline{%
  \noalign{\ifnum0='}\fi\hrule \@height \arrayrulewidth \futurelet
  \reserved@a\@xhline}
\def\@xhline{%
  \ifx\reserved@a\hline
    \vskip\doublerulesep
    \vskip-\arrayrulewidth
  \fi
  \ifnum0='{ \fi}}
\def\cline#1{\@cline#1\@nil}
\def\@cline#1-#2\@nil{%
  \omit
  \@multicnt#1%
  \advance\@multispan\m@ne
  \ifnum\@multicnt=\@ne\@firstofone{&\omit}\fi
  \@multicnt#2%
  \advance\@multicnt-#1%
  \advance\@multispan\@ne
  \leaders\hrule\@height\arrayrulewidth\hfill
  \cr
  \noalign{\vskip-\arrayrulewidth}}

```

`\memcline` This is a version of the kernel `\cline` code that provides a variable width.

```

\m@m@cline 6794 \newcommand*{\memcline}[2]{\m@m@cline[#1]#2\@nil}
6795 \def\m@m@cline[#1]#2-#3\@nil{%
6796   \omit
6797   \@multicnt#2%
6798   \advance\@multispan\m@ne
6799   \ifnum\@multicnt=\@ne\@firstofone{&\omit}\fi
6800   \@multicnt#3%
6801   \advance\@multicnt-#2%
6802   \advance\@multispan\@ne
6803   \leaders\hrule\@height #1\hfill % <- variable \@height value

```

```

6804 \cr
6805 \noalign{\vskip- #1}} % <- variable \@height value
6806

```

`\memhline` A version of `\hline`, based on `\memcline`, taking a rule width as argument. It
`\m@mhline` draws a `\memcline` from the first to the last (`\@curtab`) column.

```

6807 \newcommand*\memhline}[1][\arrayrulewidth]{\memcline{#1}{1-\@curtab}}
6808 \newcommand*\m@mhline{\cline{1-\@curtab}}

```

Fear's rules also do not play well within continuous tabulars.

`\m@m@BTnormal` Special version of `\@BTnormal`

```

6809 \def\m@m@BTnormal{%
6810 \ifnum0='{ \fi} % closes the \noalign
6811 \multispan{\@curtab} \leaders\bkta brule{\@thisrulewidth}\hfill\cr
6812 \noalign{\ifnum0='{ \fi
6813 \futurenon spacelet\@tempa\@BTendrule}
6814

```

`\ctabsetlines` Continuous tabulars cannot use the standard horizontal lines.

```

6815 \def\ctabsetlines{%
6816 \let\hline\m@mhline
6817 \let\@BTnormal\m@m@BTnormal}
6818

```

17.9 Automated tabulations

It can be convenient, especially while drafting a document, to have a list of items put into a tabular without having to mark the ends of the rows.

The following is based on plain T_EX code given on pages 307–308 of *TeX for the Impatient* [ABH90], which provides code for typesetting in columns.

`\abovecolumnspenalty` Counters and such.

```

\@linestogo 6819 \newcount\abovecolumnspenalty
\@cellstogo 6820 \abovecolumnspenalty=10000
\@cellsincolumn 6821 \newcount\@linestogo % lines remaining to be procesed
\crtok 6822 \newcount\@cellstogo % cells remaining in column or row
6823 \newcount\@cellsincolumn % number of lines per column
6824 \newtoks\crtok
6825 \crtok = {\cr}%
6826

```

`\@mincolumnwidth` More things

```

\c@lleftskip 6827 \newdimen\@mincolumnwidth
\c@lrightskip 6828 \let\c@lleftskip\hfil % left skip within a column
6829 \let\c@lrightskip\hfil % right skip within a column
6830

```

```

\preautotab Hooks into the auto tabulations.
\postautotab 6831 \let\preautotab\relax
               6832 \let\postautotab\relax
               6833

\autocols \autocols[<width>]{<pos>}{<num>}{<style>}{<comma separated list>} arranges
the elements in the <comma separated list> into <num> columns, the elements
filling each column before moving to the next. That is, the elements are ordered
top to bottom and left to right.

6834 \newcommand{\autocols}[5][Opt]{\par\begin{group}
6835   \ctabsetlines
      Set the table position
6836   \if l#2
6837     \raggedright
6838   \else
6839     \if r#2
6840       \raggedleft
6841     \else
6842       \centering
6843     \fi
6844   \fi

      Set the column position style
6845   \let\c@lleftskip\hfil
6846   \let\c@lrightskip\hfil
6847   \if l#4
6848     \let\c@lleftskip\relax
6849   \else
6850     \if r#4
6851       \let\c@lrightskip\relax
6852     \fi
6853   \fi

      Count the number of entries and the minimum width (max entry width) for the
      columns.
6854   \@mincolumnwidth\z@
6855   \TX@cols=#3
6856   \@curtab=#3
6857   \@linestogo\z@
6858   \@for\@tempa:=#5\do{
6859     \advance\@linestogo\@ne
6860     \settowidth{\@tempdima}{\@tempa}
6861     \ifdim\@tempdima>\@mincolumnwidth
6862       \@mincolumnwidth=\@tempdima
6863     \fi
6864   }
6865   \advance\@mincolumnwidth\tabcolsep
6866   \linespercol

```

Specify what is to be done after every entry

```

6867 \def\@endcolumnactions{%
6868 \global\advance\@linestogo\m@ne
6869 \ifnum\@cellstogo<\tw@
6870 \global\advance\TX@cols\m@ne
6871 \ifnum\TX@cols>\z@\linespercol\fi
6872 \the\crtok
6873 \else
6874 &\global\advance\@cellstogo\m@ne
6875 \fi}%

```

Calculate the width of the columns

```

6876 \ifdim #1 > \z@
6877 \TX@col@width=#1
6878 \divide\TX@col@width \TX@cols
6879 \else
6880 \TX@col@width=\@mincolumnwidth
6881 \fi
6882 \penalty\abovecolumnspenalty
6883 \noindent% usually not a paragraph

```

Create most of the preamble by looping to add \@cellsincolumn-1 slots, then the last one which is different.

```

6884 \def\@preamble{%
6885 \begingroup
6886 \let\@sharp\relax
6887 \ifnum\@cellsincolumn>\@ne
6888 \loop
6889 \g@addto@macro{\@preamble}{%
6890 \hb@xt@ \TX@col@width{%
6891 \c@lleftskip\strut\@sharp\c@lrightskip} &}%
6892 \advance\@cellsincolumn\m@ne
6893 \ifnum\@cellsincolumn>\@ne
6894 \repeat
6895 \fi
6896 \g@addto@macro{\@preamble}{%
6897 \hb@xt@ \TX@col@width{\c@lleftskip\strut\@sharp\c@lrightskip}}%
6898 \endgroup
6899 \let\@sharp ##

```

Start the \valign

```

6900 \tabskip\ctableftskip
6901 %% \tabskip\z@
6902 \valign \bgroup
6903 \tabskip\z@
6904 \@preamble
6905 \tabskip\ctabrightskip\cr

```

Add all the entries then finish off.

```

6906 \@for\@tempa:=#5\do{
6907 \@tempa\unskip\space\@endcolumnactions}%

```



```

6908   \the\crtok \egroup \par \endgroup}
6909

```

`\linespercol` `\linespercol` calculates the maximum number of lines that go into a column, where there are `\TX@cols` columns and `\@linestogo` lines, so that the columns are balanced as well as possible. The result is `\@cellstogo`

```

6910 \newcommand*{\linespercol}{%
6911   \@cellsincolumn=\@linestogo
6912   \divide\@cellsincolumn \TX@cols
6913   \@cellstogo=\@cellsincolumn
6914   \multiply\@cellstogo \TX@cols
6915   \@tempcnta=\@linestogo
6916   \advance\@tempcnta -\@cellstogo
6917   \ifnum \@tempcnta>\z@
6918     \advance\@cellsincolumn \@ne
6919   \fi
6920   \global\@cellstogo=\@cellsincolumn}
6921

```

`\autorows` `\autorows[$\langle width \rangle$]{ $\langle pos \rangle$ }{ $\langle num \rangle$ }{ $\langle style \rangle$ }{ $\langle comma separated list \rangle$ }` arranges the elements in the *$\langle comma separated list \rangle$* into $\langle num \rangle$ columns, the elements filling each row before moving to the next. That is, the elements are ordered left to right and top to bottom. By default, each column is the same width, enough for the widest entry. If $\langle width \rangle$ is a negative length (e.g., -1pt) the columns are set to their natural widths. If $\langle width \rangle$ is positive (e.g., `\textwidth`), column widths are equal widths so that the overall width is $\langle width \rangle$. The sideways location of the tabular is $\langle pos \rangle$ (l, c, or r), and the column style id $\langle style \rangle$ (l, c, or r).

```

6922 \newcommand{\autorows}[5][0pt]{\par\begin{group}
6923   \ctabsetlines
        Set the table position
6924   \ctableleftskip\fill
6925   \ctabrightskip\fill
6926   \if l#2
6927     \ctableleftskip\z@
6928   \else
6929     \if r#2
6930       \ctabrightskip\z@
6931     \fi
6932   \fi

```

```

        Set the column position style
6933   \let\c@lleftskip\hfil
6934   \let\c@lrightskip\hfil
6935   \if l#4
6936     \let\c@lleftskip\relax
6937   \else
6938     \if r#4

```

```

6939     \let\c@lrightskip\relax
6940     \fi
6941   \fi

```

Count the number of entries and the minimum width (max entry width) for the columns.

```

6942   \TX@cols=#3\relax
6943   \@curtab=#3\relax
6944   \@cellstogo = \TX@cols
6945   \@mincolumnwidth\z@
6946   \@linestogo\z@
6947   \@for\@tempa:=#5\do{%
6948     \advance\@linestogo\@ne
6949     \settowidth{\@tempdima}{\@tempa}
6950     \ifdim\@tempdima>\@mincolumnwidth
6951       \@mincolumnwidth=\@tempdima
6952     \fi}%
6953   \advance\@mincolumnwidth\tabcolsep

```

Specify what is to be done after every entry

```

6954   \def\@endcolumnactions{%
6955     \global\advance\@linestogo\m@ne
6956     \global\advance\@cellstogo\m@ne
6957     \ifnum\@cellstogo<\@ne
6958       \global\@cellstogo=\TX@cols
6959       \the\crtok
6960     \else
6961       &
6962     \fi}%

```

Calculate the width of the columns

```

6963   \ifdim #1>\z@
6964     \TX@col@width=#1
6965   \else
6966     \TX@col@width=\hsize
6967   \fi
6968   \divide\TX@col@width \TX@cols
6969   \ifdim #1=\z@
6970     \TX@col@width=\@mincolumnwidth
6971   \fi
6972   \penalty\abovecolumnspenalty
6973   \noindent % usually not a paragraph
6974   \vskip -\z@ % don't know why we need this, but looks bad without it

```

Create most of the preamble by looping to add \@cellsincolumn-1 slots, then the last one which is different.

```

6975   \def\@preamble{%
6976     \begingroup
6977     \let\@sharp\relax
6978     \ifnum\TX@cols>\@ne
6979       \loop

```

```

6980     \ifdim #1<\z@
6981         \g@addto@macro{\@preamble}{%
6982             \strut\c@lleftskip\@sharp\c@lrightskip &}%
6983     \else
6984         \g@addto@macro{\@preamble}{%
6985             \hb@xt@ \TX@col@width{%
6986                 \strut\c@lleftskip\@sharp\c@lrightskip} &}%
6987     \fi
6988     \advance\TX@cols\m@ne
6989     \ifnum\TX@cols>\@ne
6990     \repeat
6991 \fi
6992 \ifdim #1<\z@
6993     \g@addto@macro{\@preamble}{%
6994         \strut\c@lleftskip\@sharp\c@lrightskip}%
6995 \else
6996     \g@addto@macro{\@preamble}{%
6997         \hb@xt@ \TX@col@width{\strut\c@lleftskip\@sharp\c@lrightskip}}%
6998 \fi
6999 \endgroup
7000 \let\@sharp ##

```

Start the \halign

```

7001 \tabskip\ctableftskip
7002 \halign to \hsize \bgroup
7003 \tabskip\z@
7004 \@preamble
7005 %% \tabskip\ctabrightskip\cr \preautotab
7006 \tabskip\ctabrightskip\cr

```

Add all the entries then finish off.

```

7007 \@for\@tempa:=#5\do{%
7008     \@tempa\unskip\space\@endcolumnactions}%
7009 %% \the\crtok \postautotab \the\crtok \egroup \endgroup \par
7010 \the\crtok \egroup \endgroup \par}
7011

```

18 Floating objects

The file `latex.dtx` only defines a number of tools with which floating objects can be defined. This is done in the document class. It needs to define the following macros for each floating object of type `TYPE` (e.g., `TYPE = figure`).

`\fps@TYPE` The default placement specifier for floats of type `TYPE`.

`\ftype@TYPE` The type number for floats of type `TYPE`. Each `TYPE` has associated a unique positive `TYPE` number, which is a power of two. E.g., figures might have type number 1, tables type number 2, programs type number 4, etc.

`\ext@TYPE` The file extension indicating the file on which the contents list for float type `TYPE` is stored. For example, `\ext@figure = 'lof'`.

`\fnum@TYPE` A macro to generate the figure number for a caption. For example, `\fnum@TYPE == 'Figure \thefigure'`.

`\makecaption<num><text>` A macro to make a caption, with `<num>` the value produced by `\fnum@...` and `<text>` the text of the caption. It can assume it's in a `\parbox` of the appropriate width. This will be used for *all* floating objects.

The actual environment that implements a floating object such as a figure is defined using the macros `\@float` and `\end@float`, which are defined in `latex.dtx`.

An environment that implements a single column floating object is started with `\@float{TYPE}[<placement>]` of type `TYPE` with `<placement>` as the placement specifier. The default value of `<PLACEMENT>` is defined by `\fps@TYPE`. The environment is ended by `\end@float`. E.g., `\figure == \@float{figure}, \endfigure == \end@float`.

18.1 Floats

To define a float environment, say `fenv`, the following macros must be defined:

- `\fps@fenv` The default placement specifier (normally `tbp`).
- `\ftype@fenv` The type number which is an integer and a power of 2.
- `\ext@fenv` The file extension for the contents list.
- `\c@fenv` A counter for the environment (for caption numbering).
- `\fnum@fenv` A macro to generate the caption 'number'.
- `\l@fenv` A macro to produce an entry in a list of...
- `\flegtocfenv` A macro to write a `\namedlegend` title to a listof file.
- `\flegfenv` A macro to typeset the name of a `\namedlegend`.
- `\toclevel@fenv` Holding a bookmark level (required if the `hyperref` package will be used).

`newflo@tctr` A counter for the type number of a new float. Normally figures are of type 1, tables type 2, and the next float type is then 4, and so on.

```

7012 \newcounter{newflo@tctr}
7013 \setcounter{newflo@tctr}{1}
7014

```

`\newfloat` `\newfloat`toargwithin{*fenv*}{*ext*}{*capname*} creates the commands for a new float environment, *fenv* (aka X), using *ext* (aka Z) as the file extension and *capname* for the caption name.

```
7015 \newcommand{\newfloat}[4][\@empty]{%
```

`\ftypeX` Define the float type, set it to the float counter, and double the counter afterwards.

```
7016 %%% \@namedef{ftype@#2}{\value{newflo@tctr}}
7017 %%% \addtocounter{newflo@tctr}{\value{newflo@tctr}}
7018 \expandafter\edef\csname ftype@#2\endcsname{\the\c@newflo@tctr}%
7019 \advance\c@newflo@tctr \c@newflo@tctr
```

`\cX` Create the counter for the caption, which must not have been previously defined.

```
7020 \@ifundefined{c@#2}{% counter is not defined
7021   \ifx \@empty#1\relax
7022     \newcounter{#2}
7023   \else
7024     \newcounter{#2}[#1]
7025     \expandafter\edef\csname the#2\endcsname{%
7026 \expandafter\noexpand\csname the#1\endcsname.\noexpand\arabic{#2}}
7027   \fi}{%
7028   \setcounter{#2}{0}
7029 }
```

`\extX` Define `\extX` for the file extension and set the new Zdepth depth counter to 1.

```
\c@Zdepth 7030 \@namedef{ext@#2}{#3}% file extension
7031 \@ifundefined{c@#3depth}{\newcounter{#3depth}}{}
7032 \setcounter{#3depth}{1}
7033
```

`\fpsX` `\fpsX` is the default float placement specification, `\fnumX` typesets the caption
`\fnumX` name and number, and `\flegX` and `\flegtocX` are for named legends.

```
\flegX 7034 \@namedef{fps@#2}{tbp} % position
\flegtocX 7035 \@namedef{fnum@#2}{#4~\@nameuse{the#2}} % caption naming
7036 \@namedef{fleg#2}{#4} % legend naming
7037 \@namedef{flegtoc#2}##1{} % legend name in ToC
7038
```

X Finally define the new float environment, in both normal and starred forms.

```
X* 7039 \newenvironment{#2}{\@float{#2}}{\end@float}
7040 \newenvironment{#2*}{\@dblfloat{#2}}{\end@dblfloat}
```

This ends the definition of `\newfloat`.

```
7041 } % end \newfloat
7042
```

`\setfloatlocations` `\setfloatlocations{<float>}{<locs>}` sets the default locations for the *float* class of floats (e.g., figure) to *locs* (initially *tbp*). For tables you might want to use:

```

\setfloatlocations{table}{htbp}
7043 \newcommand*\setfloatlocations[2]{\@namedef{fps@#1}{#2}}
7044

```

To define subcaptions for use in a new float environment, say `fenv`, the following macros must be defined [Coc02]:

- A new counter `subfenv` for subcaption numbering.
- A new counter `extdepth`, where `ext` is the file extension for the contents list of `fenv`, for setting the contents depth.
- `\thesubfenv` for the formatting of the subcaption number.
- `\@thesubfenv` for typesetting the number.
- `\@@thesubfenv` for alternative label reference.
- `\p@subfenv` for prepending to the subcaption number when it is referenced.
- `\ext@subfenv` the file extension for the contents list.
- `\l@subfenv` for formatting the contents list entry.
- `\@makesubfloatcaption` for typesetting the subcaption.
- `\toclevel@subfenv` for hyperref bookmarks

`\newsfloat` `\newsfloat{<fenv>}` creates the commands for a new subfloat for `<fenv>` (aka `X`).

```
7045 \newcommand{\newsfloat}[1]{%
```

Call `\newlistentry[X]{subX}{extX}{1}` to get most of the work done.

```
7046 \newlistentry[#1]{sub#1}{\@nameuse{ext@#1}}{1}
```

`\ext@subX` And now for the rest of the commands for subcaptions.

```
\thesubX 7047 \@namedef{ext@sub#1}{\csname ext@#1\endcsname}
```

```
\@thesubX 7048 \@namedef{thesub#1}{(\alph{sub#1})}
```

```
\@@thesubX 7049 \@namedef{@thesub#1}{\@nameuse{thesub#1}}%
```

```
\p@subX 7050 \if@tightsubcap\hskip\subfloatlabelskip\else\space\fi}
```

```
\toclevel@subX 7051 \@namedef{@@thesub#1}{\@nameuse{thesub#1}}
```

```
7052 \@namedef{p@sub#1}{\csname the#1\endcsname}
```

```
7053 \@namedef{toclevel@sub#1}{1}}
```

```
7054
```

By permission of Steven Douglas Cochran the class provides similar functionality as the `subfigure` package [Coc02]. This requires some changes to be made to `\@float` and `\end@float`.

`\ifdonemaincaption` This is set TRUE after the `\(cont)caption` has been called in a float.

```
7055 \newif\ifdonemaincaption
```

```
7056 \donemaincaptionfalse
```

```
7057
```

`\@float` The kernel `\@float` and `\@dblfloat` macros are redefined to set
`\@dblfloat` `\ifdonemaincaption` to FALSE, and also to zero the subfloat counter, if it is defined. Ignasi Furió reported¹⁷ that floats embedded in text created an extra space. This was caused by a missing %.

```
7058 \let\@memoldfloat\@float
7059 \renewcommand{\@float}[1]{\donemaincaptionfalse
7060   \ifundefined{c@sub#1}{\csname c@sub#1\endcsname = 0\relax}%
7061   \@memoldfloat{#1}}
7062 \let\@memolddblfloat\@dblfloat
7063 \renewcommand{\@dblfloat}[1]{\donemaincaptionfalse
7064   \ifundefined{c@sub#1}{\csname c@sub#1\endcsname = 0\relax}%
7065   \@memolddblfloat{#1}}
7066
```

`\end@float` The kernel `\end@float` and `\end@dblfloat` macros are redefined to dump out
`\end@dblfloat` any subcaptions that have not yet been processed.

```
7067 \let\@memoldefloat\end@float
7068 \def\end@float{%
7069   \@memlistsubcaptions{\@capytype}\@memoldefloat}
7070 \let\@memolddblfloat\end@dblfloat
7071 \def\end@dblfloat{%
7072   \@memlistsubcaptions{\@capytype}\@memolddblfloat}
7073
```

Unfortunately the `fixltx2e` package, version 1.1h (current as of 2005/09/03) makes assumptions about `\end@float` which do not hold for memoir. The code has to be reverted back to that in version 1.0b of `fixltx2e`.

```
7074 \AtBeginDocument{\ifpackageloaded{fixltx2e}{%
7075   \def\end@dblfloat{%
7076     \if@twocolumn
7077       \@endfloatbox
7078       \ifnum\@floatpenalty<\z@
7079         \@largefloatcheck
7080         \global\dp\@currbox1sp %
7081         \@cons\@deferlist\@currbox
7082       \fi
7083       \ifnum\@floatpenalty=-\@Mii \@Esphack\fi
7084     \else
7085       \end@float
7086     \fi}}}}
7087
```

18.2 Captions

The caption styling is accomplished by redefining the `\@makecaption` command. First, though, define and initialise the user-level commands.

¹⁷Private email from ignasi.furio@uib.es, 2003/10/17

```

\if@contcw For use when checking caption width and captioning styles styles.
\if@conthang 7088 \newif\if@contcw
\if@contindent 7089 \newif\if@conthang
                7090 \newif\if@contindent
                7091

\captiondelim For the caption delimiter.
\@contdelim 7092 \newcommand{\captiondelim}[1]{\def\@contdelim{#1}}
                7093 \captiondelim{: }
                7094

\captionnamefont The font for the caption name.
\@contnfont 7095 \newcommand{\captionnamefont}[1]{\def\@contnfont{#1}}
                7096 \captionnamefont{}
                7097

\captiontitlefont The font for the caption title.
\@conttfont 7098 \newcommand{\captiontitlefont}[1]{\def\@conttfont{#1}}
                7099 \captiontitlefont{}
                7100

\captionstyle The paragraphing style for the caption.
\@contcstyle 7101 \newcommand*\captionstyle[1]{\def\@contcstyle{#1}}
                7102 \captionstyle{}
                7103

\captionstyle The paragraphing style for the caption.
\@memcshort I had email from Jørgen Larsen (jl@dirac.ruc.dk), 2003/04/09, asking for
\@memcnom separate controls for the short and long captions.
\@contcshortstyle \captionstyle[short]{normal}.
\@contcstyle 7104 \renewcommand{\captionstyle}{%
                7105 \ifnextchar[ {\@memcshort}{\@memcnorm}}
                7106 \def\@memcshort[#1]#2{%
                7107 \def\@contcshortstyle{#1}
                7108 \def\@contcstyle{#2}}
                7109 \def\@memcnorm#1{%
                7110 \def\@contcshortstyle{#1}
                7111 \def\@contcstyle{#1}}
                7112 \captionstyle{}
                7113

\@contcwidth The macros for dealing with the caption width.
\captionwidth 7114 \newlength{\@contcwidth}
\changecaptionwidth 7115 \newcommand{\captionwidth}[1]{\setlength{\@contcwidth}{#1}}
\normalcaptionwidth 7116 \captionwidth{\linewidth}
                7117 \newcommand{\changecaptionwidth}{\@contcwtrue}
                7118 \newcommand{\normalcaptionwidth}{\@contcwfalse}
                7119 \normalcaptionwidth
                7120

```



```

\@contindw The macros for hanging and indented captions.
\hangcaption 7121 \newlength{\@contindw}
\indentcaption 7122 \newcommand{\hangcaption}{\@conthangtrue\@contindentfalse}
\normalcaption 7123 \newcommand{\indentcaption}[1]{\setlength{\@contindw}{#1}%
7124   \@conthangfalse\@contindenttrue}
7125 \newcommand{\normalcaption}{\@conthangfalse\@contindentfalse}
7126 \normalcaption
7127

\precaption The macros for the pre- and post-caption text/commands, and for the
\@contpre mid-caption command for bilingual captions.
\postcaption 7128 \newcommand{\precaption}[1]{\def\@contpre{#1}}
\@contpost 7129 \precaption{}
\midbicapcaption 7130 \newcommand{\postcaption}[1]{\def\@contpost{#1}}
\@contmidbi 7131 \postcaption{}
7132 \newcommand{\midbicapcaption}[1]{\def\@contmidbi{#1}}
7133 \midbicapcaption{}
7134

\captiontitlefinal \captiontitlefinal{<stuff>} will put <stuff> immediately at the end of a
\caption's title text but it will not appear in the LoF/LoT/etc. For example
\captiontitlefinal{.}
The code was supplied by Frederic Connes.
7135 \newcommand*\captiontitlefinal[1]{\def\@contfinal{#1}}
7136 \captiontitlefinal{}
7137

\abovecaptionskip Vertical space above and below a caption. Make them sum to an integral number
\belowcaptionskip of lines.
7138 \newlength{\abovecaptionskip}
7139 \setlength{\abovecaptionskip}{0.5\onelineskip}
7140 \newlength{\belowcaptionskip}
7141 \setlength{\belowcaptionskip}{0.5\onelineskip}
7142

\caption For subfloat support, the (kernel) \caption macro needs to note that it has been
called.
7143 \let\@memoldcaption\caption
7144 \def\caption{\donemaincaptiontrue\@memoldcaption}
7145

\memcaptioninfo \memcaptioninfo{type}{\thetypenum}{short}{long}
7146 \newcommand{\memcaptioninfo}[4]{}
7147

\@caption For title referencing support, the (kernel) \@caption macro needs to store the
title.
7148 \let\@memold@caption\@caption

```

```

7149 \long\def\@caption#1[#2]#3{%
7150   \M@gettitle{#2}%
7151   \memcaptioninfo{#1}{\csname the#1\endcsname}{#2}{#3}%
7152   \@memoldcaption{#1}[#2][#3]}
7153

```

\@makecaption This is a reimplementaion of the kernel **\@makecaption** command. As well as including the caption typesetting commands it enables captions that include forced newlines (e.g., by `\\`).

The first part is due to Donald Arseneau¹⁸ from postings to the CTT newsgroup and Email discussions. The `\topskip` strut is used whenever the caption is the first part of the float. This means, among other things, that if a caption comes at the top of a page, then the first line of the caption will be aligned with the normal first line of a page. The `\abovecaptionskip` is only used when there is something above the caption in the current float.

```

7154 \long\def\@makecaption#1#2{\let\@memtempa\relax
7155   \ifdim\prevdepth>-99\p@ \vskip\abovecaptionskip
7156   \else \def\@memtempa{\vbox to\topskip{}}\fi

```

\@contfnote The caption title will be typeset twice, firstly to measure its width and secondly
\@contfmark to actually typeset it. To avoid problems caused by a footnote in the caption getting processed twice, we temporarily disable the expected relevant commands.

```

7157   \let\@contfnote\footnote \renewcommand{\footnote}[2] [] {}
7158   \let\@contfmark\footnotemark \renewcommand{\footnotemark}[1] [] {}

```

Now measure the width of the total caption, not forgetting to take account of the font specifications, and then restore the footnoting.

```

7159   \sbox\@tempboxa{\@contnfont #1\@contdelim \@conttfont #2\@contfinal}
7160   \let\footnote\@contfnote
7161   \let\footnotemark\@contfmark

```

If the caption is less than one line, then the whole caption needs to be centered on the page (otherwise the short caption may be typeset flushleft).

```

7162   \ifdim\wd\@tempboxa<\linewidth \centering \fi
7163   \if@contcw

```

For typesetting at anything other than the normal width, put the caption into a `\parbox` of the specified width. This must be centered.

```

7164     \centering
7165     \parbox{\@contcwidth}{%

```

Henrik Holm¹⁹ proposed adding the next line to center short, narrow captions.

```

7166     \ifdim\wd\@tempboxa<\@contcwidth \centering \fi
7167     \fi

```

Hanging and indenting doesn't apply to short captions, so do these now.

```

7168   \ifdim\wd\@tempboxa<\linewidth

```

¹⁸Email: asnd@triumf.ca

¹⁹Email from henrik@tele.ntnu.no on 2002/02/10.

```

7169 \contpre
7170 {\contnfont #1\contdelim}\memtempa
7171 {\contcshortstyle \conttfont #2\contfinal\par}
7172 \else
7173 \if@conthang

```

For a hanging caption we have to measure the width of the caption name, then typeset the whole caption in a hanging paragraph.

```

7174 \sbox\tempboxa{\contnfont #1\contdelim}
7175 \contpre%
7176 {\contcstyle\hangindent=\wd\tempboxa
7177 \noindent\box\tempboxa\memtempa \conttfont #2\contfinal\par}
7178 \else
7179 \if@contindent

```

An indented caption is similar, except the amount of indentation is kept in \contindw.

```

7180 \contpre%
7181 {\contnfont #1\contdelim}\memtempa
7182 {\contcstyle\hangindent=\contindw
7183 \hangafter=\@ne\conttfont #2\contfinal\par}% <- v1.4
7184 \else

```

For the normal style, just typeset the caption.

```

7185 \contpre%
7186 {\contnfont #1\contdelim}\memtempa
7187 {\contcstyle \conttfont #2\contfinal\par}
7188 \fi
7189 \fi
7190 \fi

```

Finish off the typesetting by processing the post-text, and if not using the normal width then close off the \parbox, and lastly put in some vertical space.

```

7191 \contpost
7192 \if@contcw
7193 \par
7194 } % end of the \parbox
7195 \fi
7196 \vskip\belowcaptionskip}
7197

```

18.2.1 Continuation captions and legends

`\contcaption` `\contcaption{<text>}` is a user-level command. It is a simplified version of the normal `\caption` command as it doesn't have to deal with numbering or list of ... entries.

However, Brent Lievers²⁰ requested that `\label` should pick up the correct caption number after a `\contcaption`.

```

7198 \newcommand{\contcaption}{%

```

²⁰(lieversb@post.queensu.ca) in CTT thread *Figures*, 2003/11/14.

```

7199 \addtocounter{\@capttype}{\m@ne}\refstepcounter{\@capttype}%
7200 \@contcaption\@capttype}
7201

```

`\@@contcaption` `\@contcaption` is the workhorse for the `\contcaption` command. In turn, it uses the `\makecaption` command (defined in the usual classes) to do most of its work. It uses the number of the previous `\caption` command in the same type of float and its implementation includes much of the code used in the L^AT_EX `\caption` command.

First specify `\@@contcaption{<type>}{<title>}` to save some repetitive code.

```

7202 \long\def\@@contcaption#1#2{%
7203   \par
7204   \begingroup
7205     \@parboxrestore
7206     \if@minipage
7207       \setminipage
7208     \fi
7209     \normalsize
7210     \makecaption{\csname fnum@#1\endcsname}{\ignorespaces #2}\par
7211   \endgroup}
7212

```

Now for `\@contcaption{<type>}{<title>}`. It has to flush out any subcaptions at the appropriate time, as well as typesetting the caption.

```

7213 \long\def\@contcaption#1#2{%
7214   \if@contbotsub
7215     \@memlistsubcaptions{#1}%
7216     \@@contcaption{#1}{#2}%
7217   \else
7218     \@@contcaption{#1}{#2}%
7219     \@memlistsubcaptions{#1}%
7220   \fi}
7221

```

`\memlegendinfo` The macro `\legend{<text>}` is intended to be used in a float environment for an ‘anonymous’ caption, but can be used anywhere.

The implementation is similar to the `\caption` command but we have to eliminate printing of a delimiter.

```

7222 \newcommand{\memlegendinfo}[1]{}
7223 \newcommand{\legend}[1]{%
7224   \M@gettitle{#1}%
7225   \memlegendinfo{#1}%
7226   \par
7227   \begingroup
7228     \@parboxrestore
7229     \if@minipage
7230       \setminipage
7231     \fi
7232     \normalsize

```

```

7233     \captiondelim{\mbox{}}
7234     \@makecaption{}{\ignorespaces #1}\par
7235 \endgroup
7236

```

`\namedlegend` `\namedlegend[<short-title>]{<long-title>}` is like the `\caption` command except `\memmamedlegendinfo` that it does not number the caption.

```

\memmamedlegendinfo{<type>}{<short-title>}{<long-title>}
7237 \newcommand{\namedlegend}{\@dblarg{\@legend\@capttype}}
7238 \newcommand{\memmamedlegendinfo}[3]{}
7239

```

`\@legend` `\@legend{<type>}[<short-title>]{<long-title>}` is the workhorse for the `\namedlegend` command. In turn, it calls `\@makelegend`. It requires two commands to have been defined, namely `\flegtotype` and `\flegtype`. The command `\flegtotype{<text>}` is responsible for writing a title text to the appropriate listof file. `\flegtype` is responsible for typesetting the name of the legend.

```

7240 \long\def\@legend#1[#2]#3{%
7241   \M@getttitle{#2}%
7242   \memmamedlegendinfo{#1}{#2}{#3}%
7243   \par
7244   \csname flegtoc#1\endcsname{#2}%
7245   \begingroup
7246     \@parboxrestore
7247     \if@minipage
7248       \@setminipage
7249     \fi
7250     \normalsize
7251     \@makecaption{\csname fleg#1\endcsname}{\ignorespaces #3}\par
7252   \endgroup}
7253

```

18.2.2 Non-float captions

`\newfixedcaption` These commands are defined in terms of their `\...command` counterparts.
`\renewfixedcaption` Call as `\...fixedcaption[<capcommand>]{<command>}{<env>}`
`\providfixedcaption`

```

7254 \newcommand{\newfixedcaption}[3][\caption]{%
7255   \newcommand{#2}{\def\@capttype{#3}#1}}
7256 \newcommand{\renewfixedcaption}[3][\caption]{%
7257   \renewcommand{#2}{\def\@capttype{#3}#1}}
7258 \newcommand{\providfixedcaption}[3][\caption]{%
7259   \providecommand{#2}{\def\@capttype{#3}#1}}
7260

```

18.2.3 Bilingual captions

The bilingual caption commands all use internal grouping so that any changes are kept local. This has the unfortunate side-effect that any `\label` command

must be within the grouping otherwise the wrong number is picked up. To make the coding, if not necessarily the use, of the commands simpler, I have not used the traditional style of square brackets for optional caption text arguments. Instead, empty ‘required’ arguments are used as the implementation means.

```

\membitwounumcaptioninfo \membitwounumcaptioninfo{type}{\thetypenum}{\langle short1\rangle}{\langle long1\rangle}
\membionenumcaptioninfo {\langle name2\rangle}{\langle short2\rangle}{\langle long2\rangle}
\membicaptioninfo \membionenumcaptioninfo{type}{\thetypenum}{\langle short1\rangle}{\langle long1\rangle}
{\langle name2\rangle}{\langle short2\rangle}{\langle long2\rangle}
\membicaptioninfo{type}{\thetypenum}{\langle short1\rangle}{\langle long1\rangle}
{\langle name2\rangle}{\langle long2\rangle}

7261 \newcommand{\membitwounumcaptioninfo}[7]{}
7262 \newcommand{\membionenumcaptioninfo}[7]{}
7263 \newcommand{\membicaptioninfo}[6]{}
7264

```

`\bitwounumcaption` The 6 arguments are: optional label, short and long in language 1, name in language 2, and short and long in language 2. Both texts are put into the List of as numbered entries.

```

7265 \newcommand{\bitwounumcaption}[6][\@empty]{%
7266   \begingroup
7267   \let\memcaptioninfo\gobblefour

   Check if the first language argument is vacuous, then call the normal \caption
   for language 1.

7268   \@ifmtarg{#2}{\def\m@mmscapi{#3}\caption{#3}}%
7269   {\def\m@mmscapi{#2}\caption[#2]{#3}}%

   Do the optional labeling.

7270   \ifx \@empty #1\else
7271     \label{#1}%
7272   \fi

```

Remove any extra spacing between the captions, and set the NAME for the second caption. Use a command to transfer the NAME to the renewal code to avoid circularity if for example, we are trying to redefine `\tablename` as `\tablename`. Decrement the caption counter.

```

7273   \setlength{\abovecaptionskip}{0pt}%
7274   \setlength{\belowcaptionskip}{0pt}%
7275   \edef\@memtempc{#4}%
7276   \expandafter\renewcommand\csname \@capttype name\endcsname{\@memtempc}%
7277   \addtocounter{\@capttype}{-1}%

   Now repeat for the second language caption.

7278   \@contmidbi
7279   \@ifmtarg{#5}{\def\m@mmscapii{#6}\caption{#6}}%
7280   {\def\m@mmscapii{#5}\caption[#5]{#6}}%
7281   \membitwounumcaptioninfo{\@capttype}{\@nameuse{the\@capttype}}%
7282   {\m@mmscapi{#3}{#4}{\m@mmscapii{#6}}%
7283   \endgroup}

```

7284

\bionenumcaption The 6 arguments are: optional labelling, short and long in language 1, name in language 2, and short and long in language 2. Both texts are put into the List of, but only the first is numbered.

```
7285 \newcommand{\bionenumcaption}[6][\@empty]{%
```

```
7286   \begingroup
```

```
7287   \let\memcaptioninfo\@gobblefour
```

Check if the first language argument is vacuous, then call the normal `\caption` for language 1.

```
7288   \@ifmtarg{#2}{\def\m@mscapi{#3}\caption{#3}}%
```

```
7289           {\def\m@mscapi{#2}\caption[#2]{#3}}%
```

Do the optional labeling.

```
7290   \ifx \@empty #1\else
```

```
7291     \label{#1}%
```

```
7292   \fi
```

Do the between captions code.

```
7293   \setlength{\abovecaptionskip}{0pt}%
```

```
7294   \setlength{\belowcaptionskip}{0pt}%
```

```
7295   \edef\@memtempc{#4}%
```

```
7296   \expandafter\renewcommand\csname \@captype name\endcsname{\@memtempc}
```

Use a continuation caption for the second language, not forgetting to add the appropriate unnumbered text to the List.

```
7297   \@contmidbi
```

```
7298   \contcaption{#6}%
```

```
7299   \@ifmtarg{#5}{%
```

```
7300     \def\m@mscapii{#6}%
```

```
7301     \addcontentsline{\csname ext@\@captype\endcsname}{\@captype}%
```

```
7302       {\protect\numberline}{\ignorespaces #6}}}%
```

```
7303     \def\m@mscapii{#5}%
```

```
7304     \addcontentsline{\csname ext@\@captype\endcsname}{\@captype}%
```

```
7305       {\protect\numberline}{\ignorespaces #5}}}%
```

```
7306   \membionenumcaptioninfo{\@captype}{\@nameuse{the\@captype}}%
```

```
7307           {\m@mscapi}{#3}{#4}{\m@mscapii}{#6}%
```

```
7308   \endgroup}
```

7309

\bicaption The 5 arguments are: optional labelling, short and long in language 1, name in language 2, and long in language 2. Only the first text is put into the List.

```
7310 \newcommand{\bicaption}[5][\@empty]{%
```

```
7311   \begingroup
```

```
7312   \let\memcaptioninfo\@gobblefour
```

Check if the first language argument is vacuous, then call the normal `\caption` for language 1.

```
7313   \@ifmtarg{#2}{\def\m@mscapi{#3}\caption{#3}}%
```

```
7314           {\def\m@mscapi{#2}\caption[#2]{#3}}%
```

Do the optional labeling.

```
7315 \ifx \@empty #1\else
7316   \label{#1}%
7317 \fi
```

Do the between captions code and finally just use `\contcaption` for the second language.

```
7318 \setlength{\abovecaptionskip}{0pt}%
7319 \setlength{\belowcaptionskip}{0pt}%
7320 \edef\@memtempc{#4}
7321 \expandafter\renewcommand\csname \@capttype name\endcsname{\@memtempc}%
7322 \@contmidbi
7323 \contcaption{#5}%
7324 \membicaptioninfo{\@capttype}{\@nameuse{the\@capttype}}%
7325                  {\m@mscapi}{#3}{#4}{#5}%
7326 \endgroup}
7327
```

`\bicontcaption` The 3 arguments are long in language 1, name in language 2, and long in language 2.

```
7328 \newcommand{\bicontcaption}[3]{%
7329   \begingroup
```

Call `\contcaption` for language 1.

```
7330   \contcaption{#1}%
```

Do the between captions code and use `\contcaption` for the second language.

```
7331   \setlength{\abovecaptionskip}{0pt}%
7332   \setlength{\belowcaptionskip}{0pt}%
7333   \edef\@memtempc{#2}%
7334   \expandafter\renewcommand\csname \@capttype name\endcsname{\@memtempc}%
7335   \@contmidbi
7336   \contcaption{#3}%
7337   \endgroup}
7338
```

18.2.4 Support for the `subfigure` package functionality

Much of the code in this section is based on the `subfigure` package code, by kind permission of its author, Steven Douglas Cochran. To try and avoid clashes with the real `subfigure` code I have used different macro names, especially when I have copied the code.

`\subcaptionstyle` The paragraphing style for subcaptions.

```
\@contsubcstyle 7339 \newcommand{\subcaptionstyle}[1]{\def\@contsubcstyle{#1}}
7340 \subcaptionstyle{}
7341
```

`\if@shortsubcap` For dealing with short and hanging subcaptions. Analogous to the `subfigure`
`\if@hangsubcap` `nooneline` and `hang` options. The default is normal subcaptions.
`\shortsubcaption`
`\hangsubcaption`
`\normalsubcaption`


```

7342 \newif\if@shortsubcap
7343 \newif\if@hangsubcap
7344 \newcommand*{\shortsubcaption}{\@shortsubcaptrue}
7345 \newcommand*{\hangsubcaption}{\@hangsubcaptrue}
7346 \newcommand*{\normalsubcaption}{\@shortsubcapfalse\@hangsubcapfalse}
7347 \normalsubcaption
7348

```

`\subfloattopskip` These `\subfloat...` lengths are analagous to the subfigure `\subfig...` lengths.

```

\subfloatcapskip 7349 \newskip\subfloattopskip
\subfloatcaptopadj 7350 \newskip\subfloatcapskip
\subfloatbottomskip 7351 \newskip\subfloatcaptopadj
\subfloatlabelskip 7352 \newskip\subfloatbottomskip
\subfloatcapmargin 7353 \newskip\subfloatlabelskip
7354 \newdimen\subfloatcapmargin

```

`\if@tightsubcap` Unlike the subfigure package the class provides no options for subcaptions. These
`\loosesubcaptions` macros provide the subfigure loose/tight option functions. Set the default to
`\tightsubcaptions` tight.

```

7355 \newif\if@tightsubcap
7356 \newcommand{\loosesubcaptions}{%
7357   \subfloattopskip = 10\p@
7358   \subfloatcapskip = 10\p@
7359   \subfloatcaptopadj = \z@
7360   \subfloatbottomskip = 10\p@
7361   \subfloatlabelskip = 0.33em
7362   \subfloatcapmargin = 10\p@
7363   \@tightsubcapfalse
7364 }
7365
7366 \newcommand{\tightsubcaptions}{%
7367   \subfloattopskip = 5\p@
7368   \subfloatcapskip = \z@
7369   \subfloatcaptopadj = 3\p@
7370   \subfloatbottomskip = 5\p@
7371   \subfloatlabelskip = 0.33em \@plus 0.07em \@minus 0.03em
7372   \subfloatcapmargin = \z@
7373   \@tightsubcaptrue
7374 }
7375 \tightsubcaptions
7376

```

`\subcaptionsize` These macros set the size and fonts for the subcaptions. Set the defaults to

`\@subcapsize` `\footnotesize`, and the normal roman font.

```

\subcaptionlabelfont 7377 \newcommand*{\subcaptionsize}[1]{\def\@subcapsize{#1}}
\@subcaplabelfont 7378 \newcommand*{\subcaptionlabelfont}[1]{\def\@subcaplabelfont{#1}}
\subcaptionfont 7379 \newcommand*{\subcaptionfont}[1]{\def\@subcapfont{#1}}
\@subcapfont 7380 \subcaptionsize{\footnotesize}
7381 \subcaptionlabelfont{\normalfont}

```

```

7382 \subcaptionfont{\normalfont}
7383

\@contkeep  \@contkeep stores the current subfloat number in counter @contsubnum and
\@contset   \@contset sets the subfloat number to the value of @contsubnum.
\subconcluded \subconcluded sets the subfloat number to zero.

7384 \newcounter{@contsubnum}
7385 \newcommand{\@contkeep}{%
7386   \setcounter{@contsubnum}{\value{sub\@capttype}}}
7387 \newcommand{\@contset}{%
7388   \setcounter{sub\@capttype}{\value{@contsubnum}}}
7389 \newcommand{\subconcluded}{%
7390   \setcounter{sub\@capttype}{0}}

\if@contbotsub A flag indicating whether the subcaption is to be at the bottom or top of the
                subfloat; TRUE for the subcaption at the bottom.

7391 \newif\if@contbotsub
7392 \@contbotsubtrue
7393

\subcaption \subcaption[list-entry]{caption} is a generic subcaption. There is no
                subfigure equivalent.

7394 \newcommand{\subcaption}{%
7395   \bgroup
7396   \let\label=\memsub@label
7397   \ifdonemaincaption\else
7398     \advance\cscname c@\@capttype\endcscname\@ne
7399   \fi
7400   \refstepcounter{sub\@capttype}\@contkeep
7401   \ifnextchar [%
7402     {\@memsubcap{sub\@capttype}}%
7403     {\@memsubcap{sub\@capttype}[\@empty]}}

\@memsubcap This handles the optional argument to \subcaption. It sets \@tempdima to
                \hsize as later on \@makesubfloatcaption uses this.

7404 \long\def\@memsubcap#1[#2]#3{%
7405   \@tempdima=\hsize
7406   \vskip\subfloatcapskip
7407   \ifx \@empty #2
7408     \@memsubcaption{#1}{#3}{#3}%
7409   \else
7410     \@memsubcaption{#1}{#2}{#3}%
7411   \fi
7412   \vskip\subfloatcapskip
7413   \egroup}

\@memsubcaption \@memsubcaption{type}{list-entry}{caption} typesets a subcaption. This
                is a copy of the subfigure \@subcaption macro.

```

```

7414 \newcommand{\@memsubcaption}[3]{%
7415   \ifx \relax#2\relax \else
7416     \bgroup
7417     \let\label\@gobble
7418     \let\protect\string
7419     \def\@memsubcaplabel{\@nameuse{@the#1}}%
7420     \xdef\@memsubfigcaptionlist{%
7421       \@memsubfigcaptionlist,%
7422       {\protect\numberline{\@memsubcaplabel}\noexpand{\ignorespaces #2}}}%
7423     \egroup
7424   \fi
7425   \@makesubfloatcaption{\@nameuse{@the#1}}{#3}}
7426

```

`\contsubcaption` `\contsubcaption[<caption>]` is the continued version of `\subcaption`.

```

7427 \newcommand{\contsubcaption}{%
7428   \bgroup
7429   \let\label=\memsub@label
7430   \@contset
7431   \refstepcounter{sub\@capttype}\@contkeep
7432   \@ifnextchar [%
7433     {\@memsubcap{sub\@capttype}}%
7434     {\@memsubcap{sub\@capttype}[\@empty]}}

```

subfloat The subfigure documentation suggests a way of defining a `subfloat` environment. This is a trivial implementation because the `\subcaption` and `\contsubcaption` commands are provided by the class, and can be used within a `subfloat`.

```

7435 \newenvironment{subfloat}{}{}
7436

```

`\subbottom` `\subbottom[<list-entry>][<subcaption>]{<text>}` typesets a subcaption below the `\@memsubbody` *<text>*. Most of the work is performed by the `\@memsubbody` macro.

```

7437 \newcommand{\subbottom}{%
7438   \@contbotsubtrue
7439   \@memsubbody}
7440
7441 \newcommand{\@memsubbody}{%
7442   \bgroup
7443   \let\label=\memsub@label
7444   \ifdonemaincaption\else
7445     \advance\csname c@\@capttype\endcsname\@ne
7446   \fi
7447   \refstepcounter{sub\@capttype}\@contkeep%
7448   \leavevmode
7449   \@ifnextchar [%
7450     {\@memsubfig}%
7451     {\@memsubfig[\@empty]}}
7452

```

`\contsubbottom` These are the continued versions of `\subbottom` and `\@memsubbody`.

```

\@memcontsubbody 7453 \newcommand{\contsubbottom}{%
7454   \@contbotsubtrue
7455   \@memcontsubbody}
7456
7457 \newcommand{\@memcontsubbody}{%
7458   \bgroup
7459   \let\label=\memsub@label
7460   \@contset
7461   \refstepcounter{sub\@capytype}\@contkeep%
7462   \leavevmode
7463   \@ifnextchar [%
7464     {\@memsubfig}%
7465     {\@memsubfig[\@empty]}}
7466
```

`\subtop` These are similar to `\subbottom` and `\contsubbottom` except that they put the subcaption on top of the *text*.

```

\contsubtop 7467 \newcommand{\subtop}{%
7468   \@contbotsubfalse
7469   \@memsubbody}
7470
7471 \newcommand{\contsubtop}{%
7472   \@contbotsubfalse
7473   \@memcontsubbody}
7474
```

`\@memsubfig` This is a revised version of the subfigure `\@subfigure` command — just the called macro names are changed.

```

7475 \def\@memsubfig[#1]{%
7476   \@ifnextchar [%
7477     {\@memsubfloat{sub\@capytype}[#1]}%
7478     {\@memsubfloat{sub\@capytype}[\@empty #1][#1]}}
7479
```

`\@memsubfloat` This is a modified version of the subfigure `\@subfloat` command. Essentially the `\csname if#1topcap\endcsname` constructs are replaced by `\if@contbotsub`.

```

7480 \def\@memsubfloat#1[#2][#3]#4{%
7481   \@tempcnta=\@ne
7482   \if@tightsubcap
7483     \if@minipage
7484       \@tempcnta=\z@
7485     \else
7486       \ifdim\lastskip=\z@
7487         \@tempcnta=\@ne
7488       \else
7489         \@tempcnta=\tw@
7490       \fi
7491     \fi

```

```

7492 \fi
7493 \if@contbotsub
7494   \def\subfig@top{\subfloattopskip}%
7495   \def\subfig@bottom{\subfloatbottomskip}%
7496 \else
7497   \def\subfig@top{\subfloatbottomskip}%
7498   \def\subfig@bottom{\subfloattopskip}%
7499 \fi
7500 \setbox\@tempboxa \hbox{#4}%
7501 \@tempdima=\wd\@tempboxa
7502 \vtop\bgroup
7503   \vbox\bgroup
7504     \ifcase\@tempcnta
7505       \@minipagefalse
7506     \or
7507       \vspace{\subfig@top}
7508     \or
7509       \ifdim \lastskip=\z@ \else
7510         \@tempskipb\subfig@top\@xaddvskip
7511       \fi
7512     \fi
7513     \if@contbotsub
7514       \box\@tempboxa\egroup
7515       \ifx \@empty#3\relax \else
7516         \vskip\subfloatcapskip
7517         \@memsubcaption{#1}{#2}{#3}%
7518       \fi
7519     \else
7520       \ifx \@empty#3\relax \else
7521         \@memsubcaption{#1}{#2}{#3}%
7522         \vskip\subfloatcapskip
7523         \vskip\subfloatcaptopadj
7524       \fi\egroup
7525       \box\@tempboxa
7526     \fi
7527     \vspace{\subfig@bottom}
7528   \egroup
7529 \egroup}
7530

```

The following series of macros, from subfigure, control the typesetting of the subcaptions.

```

\@memsubfigcaptionlist A copy of \@subfigcaptionlist.
7531 \newcommand*\@memsubfigcaptionlist{}

\memlistsubcaptions A copy of \listsubcaptions.
7532 \newcommand*\memlistsubcaptions{%
7533   \@ifstar
7534   {\gdef\@memsubfigcaptionlist{}}%

```

```

7535     {\@memlistsubcaptions{\@capttype}}
7536

```

`\@memlistsubcaptions` A copy of `\@listsubcaptions`.

```

7537 \newcommand*{\@memlistsubcaptions}[1]{%
7538   \@ifundefined{\@capttype}{\%
7539     \@ifundefined{ext@sub#1}{\%
7540       \@for \@tempa:=\@memsubfigcaptionlist \do {%
7541         \ifx \@empty\@tempa\relax \else
7542           \addcontentsline{\@nameuse{ext@sub#1}}{sub#1}{\@tempa}%
7543         \fi}}%
7544   \gdef\@memsubfigcaptionlist{}}
7545

```

`\@makesubfloatcaption` This is a copy of `\@makesubfigurecaption`.

```

7546 \newcommand{\@makesubfloatcaption}[2]{%
7547   \setbox\@tempboxa\hbox{%
7548     \@subcapsize
7549     {\@subcaplabelfont #1}{\@subcapfont\ignorespaces #2}}%
7550   \@tempdimb=-\subfloatcapmargin
7551   \multiply\@tempdimb\tw@
7552   \advance\@tempdimb\@tempdima
7553   \hb@xt@\@tempdima{%
7554     \hss
7555     \ifdim \wd\@tempboxa >\@tempdimb
7556       \memsubfig@caption{#1}{#2}%
7557     \else
7558       \if@shortsubcap
7559         \memsubfig@caption{#1}{#2}%
7560       \else
7561         \box\@tempboxa
7562       \fi
7563     \fi
7564     \hss}}
7565

```

`\@memsubfig@caption` This is a copy of `\@subfig@caption`.

```

7566 \newcommand{\@memsubfig@caption}[2]{%
7567   \if@hangsubcap
7568     \sbox{\@tempboxa}{\@subcapsize\@subcaplabelfont #1}%
7569     \addtolength{\@tempdimb}{-\wd\@tempboxa}%
7570     \usebox{\@tempboxa}%
7571     \memsubfig@captionpar{\@tempdimb}{%
7572       {\@subcapfont\ignorespaces #2}}%
7573   \else
7574     \memsubfig@captionpar{\@tempdimb}{\@subcaplabelfont #1}%
7575     {\@subcapfont\ignorespaces #2}}%
7576   \fi}
7577

```

`\memsubfig@captionpar` This replaces `\subfig@captionpar`.

```
7578 \newcommand{\memsubfig@captionpar}[2]{%
7579   \parbox[t]{#1}{\@subcapsize\@contsubcstyle #2}}
7580
```

`\memsub@label` These are copies of `\sub@label` and `\subref`.

```
7581 \newcommand{\memsub@label}{%
7582   \@ifnextchar (%
7583     {\sf@memsub@label}%
7584     {\sf@memsub@label(Sub\@capttype\space
7585                          \@nameuse{p@sub\@capttype}%
7586                          \@nameuse{thesub\@capttype}})}%
```

`\sf@memsub@label` This is a copy of `\sf@sub@label`.

```
7587 \def\sf@memsub@label(#1)#2{%
7588   \protected@edef\mem@currentlabelname{#1}%
7589   \sf@memsub@label{#2}}
7590
```

`\sf@memsub@label` This is an expanded copy of `\sf@sub@label` processed after any packages may have been loaded.

```
7591 \AtBeginDocument{%
7592   \@ifpackageloaded{nameref}{%
7593     \newcommand*\sf@memsub@label[1]{%
7594       \bsphack
7595       \protected@write\@auxout{}{%
7596         \string\newlabel{#1}%
7597           {\@nameuse{p@sub\@capttype}\@nameuse{@@thesub\@capttype}}%
7598           {\thepage}%
7599           {\mem@currentlabelname\relax}%
7600           {\@currentHref}}}%
7601       \protected@write\@auxout{}{%
7602         \string\newlabel{sub#1}%
7603           {\@nameuse{@@thesub\@capttype}}%
7604           {\thepage}%
7605           {\mem@currentlabelname\relax}%
7606           {\@currentHref}}}%
7607       \@esphack}
7608   }{\@ifpackageloaded{hyperref}{%
```

The `hyperref` package is loaded, but not `nameref`.

```
7609   \newcommand*\sf@memsub@label[1]{%
7610     \bsphack
7611     \protected@write\@auxout{}{%
7612       \string\newlabel{#1}%
7613         {\@nameuse{p@sub\@capttype}\@nameuse{@@thesub\@capttype}}%
7614         {\thepage}%
7615         {\mem@currentlabelname\relax}%
7616     }
```

```

7616         {\@currentHref}{}}}%
7617     \protected@write\@auxout{}{%
7618         \string\newlabel{sub@#1}%
7619         {\@nameuse{@@thesub\@captive}}%
7620         {\thepage}%
7621         {\mem@currentlabelname\relax}%
7622         {\@currentHref}{}}}%
7623     \@esphack}
7624 }{%

```

Neither the hyperref nor the nameref package is loaded.

```

7625     \let\@memoldlabel\label
7626     \newcommand*{\sf@memsub@label}[1]{%
7627         \@bsphack
7628         \@memoldlabel{#1}%
7629         \protected@write\@auxout{}{%
7630             \string\newlabel{sub@#1}%
7631             {\@nameuse{@@thesub\@captive}}%
7632             {\thepage}}}%
7633     \@esphack}
7634 }{%
7635 }
7636 }
7637

```

`\subcaptionref` This is a copy of the `\subref` macro.

```

7638 \DeclareRobustCommand{\subcaptionref}{%
7639     \@ifstar{\ssc@ref}{\sc@ref}}

```

`\ssc@ref` The implementation of the starred and unstarred forms of `\subcaptionref`.

```

\sc@ref 7640 \newcommand*{\ssc@ref}[1]{\ref{sub@#1}}
7641 \newcommand*{\sc@ref}[1]{\@subcaplabelfont\ref{sub@#1}}
7642

```

18.2.5 Side captions

`\m@mscap@capbox` We need two save boxes, one to hold the caption and the other for the float material.

```

7643 \newsavebox{\m@mscap@capbox}
7644 \newsavebox{\m@mscap@fbox}
7645

```

`\sidecapsep` `\sidecapsep` is the space between the text and the caption, which is set in a box
`\sidecapwidth` `\sidecapwidth` wide. These are initialized to the `\marginpar...` values.

```

7646 \newdimen\sidecapsep
7647 \sidecapsep=\marginparsep
7648 \newdimen\sidecapwidth
7649 \sidecapwidth=\marginparwidth

```


`\setsidecaps` `\setsidecaps{⟨sep⟩}{⟨width⟩}` sets the `\sidecapsep` and `\sidecapwidth` (Ivars Finvers noted that the initial values don't change if the `\marginpar...` values change, such as at `\checkandfixthelayout`).

```
7650 \newcommand*{\setsidecaps}[2]{%
7651   \setlength{\sidecapsep}{#1}\@memznegtest{\sidecapsep}%
7652   \setlength{\sidecapwidth}{#2}\@memznegtest{\sidecapwidth}}
7653
```

`\m@m@tempdima` Lengths for internal use. `\m@m@tempdima` is meant for general temporary use.

```
\m@m@tempdima 7654 \newdimen\m@m@tempdima
7655 \newdimen\m@m@tempdima
```

`\sidecapraise` Length to make (small) adjustments to the position of the caption wrt the float.

```
7656 \newdimen\sidecapraise
7657 \sidecapraise \z@
7658
```

`\setsidecappos` `\setsidecappos{⟨pos⟩}`, where `⟨pos⟩` is one of `t`, `c`, or `b`, sets the vertical position of the caption in relation to the float and the result is saved as `\m@m@tempcappos`. The default is `c`.

```
7659 \newcommand*{\setsidecappos}[1]{%
7660   \def\m@m@tempcappos{#1}\def\@tempb{t}%
7661   \ifx\@tempb\m@m@tempcappos
7662   \else
7663     \def\@tempb{b}%
7664     \ifx\@tempb\m@m@tempcappos
7665     \else
7666       \def\@tempb{c}%
7667       \ifx\@tempb\m@m@tempcappos
7668       \else
7669         \@memerror{Argument to \string\setsidecappos\space is not t or c or b.
7670           \MessageBreak Set to c}{\@ehc}%
7671       \def\m@m@tempcappos{c}%
7672     \fi
7673   \fi
7674 \fi}
7675 \setsidecappos{c}
7676
```

`\sidecapmargin` `\sidecapmargin{⟨margin⟩}`, where `⟨margin⟩` is one of `left`, `right`, `inner`, or `outer`, controls the margin where the caption will be put. The result is saved as `\m@m@tempcapmargin` as a number. The default is `left`.

```
7677 \newcommand{\sidecapmargin}[1]{%
7678   \def\@tempa{#1}\def\@tempb{left}%
7679   \ifx\@tempb\@tempa
7680     \def\m@m@tempcapmargin{0}% left
7681   \else
7682     \def\@tempb{right}%

```

```

7683     \ifx\@tempb\@tempa
7684         \def\m@mscapmarg{1}%   right
7685     \else
7686         \def\@tempb{outer}%
7687         \ifx\@tempb\@tempa
7688             \def\m@mscapmarg{2}%   outer
7689         \else
7690             \def\@tempb{inner}%
7691             \ifx\@tempb\@tempa
7692                 \def\m@mscapmarg{3}%   inner
7693             \else
7694                 \@memerror{Unrecognized argument for \string\sidecapmargin}%
7695                     {\@ehc}%
7696                 \def\m@mscapmarg{-1}% error
7697             \fi
7698         \fi
7699     \fi
7700 \fi}
7701 \sidecapmargin{left}
7702

```

`\ifscapmargleft` `\ifscapmargleft` is TRUE the caption should be in the left hand margin, otherwise in the right hand margin.

`\scapmarglefttrue` `\scapmargleftfalse` 7703 `\newif\ifscapmargleft`

7704

`\sidecapfloatwidth` `\sidecapfloatwidth` is the width of the box holding the float. Note that this is a macro, not a length, so must be changed using `\renewcommand*`. The default is `\linewidth`. Later, `\m@mscapmainwidth` will be set to the current length specification from `\sidecapfloatwidth`.

7705 `\def\sidecapfloatwidth{\linewidth}`

7706 `\newdimen\m@mscapmainwidth`

7707

`\m@mscaplkern` `\setm@mscaplkern` is a utility macro to calculate the kern (`\m@mscaplkern`) required when the caption is in the left margin.

```

7708 \newdimen\m@mscaplkern
7709 \newcommand*\setm@mscaplkern{%
7710     \m@mscaplkern=\sidecapwidth
7711     \advance\m@mscaplkern \sidecapsep
7712     \advance\m@mscaplkern \m@mscapmainwidth}
7713

```

`\sidecapstyle` `\sidecapstyle` is called just before the caption is set. It can be redefined to set different caption style parameters. The default is `raggedleft` for left margin captions and `raggedright` for right margin captions.

```

7714 \newcommand*\sidecapstyle{%
7715 %%% \captionnamefont{\bfseries}%
7716 \ifscapmargleft

```

```

7717 \captionstyle{\raggedleft}%
7718 \else
7719 \captionstyle{\raggedright}%
7720 \fi}
7721

```

`\sidecaption` The whole shebang is in the `sidecaption` environment but it is more convenient to specify this via macros rather than directly as an environment.

`\sidecaption[<fortoc>]{<title>}[<label>]` is what it looks like to the user, but internally further macros handle all the arguments.

```

7722 \newcommand*{\sidecaption}{%
7723 \ifnextchar [{\@sidecaption}{\@sidecaption[]}}

```

`\@sidecaption` `\@sidecaption[<fortoc>]{<title>}` grabs the first two arguments.

```

7724 \def\@sidecaption[#1]#2{%
7725 \ifnextchar [{\@@sidecaption{#1}{#2}}{\@@sidecaption{#1}{#2} []}}

```

`\@mem@scap@beforehook` Two hooks into the 'scap' internals. By default they do nothing, but can be used
`\@mem@scap@afterhook` make say sidecaption align towards the spine in twosided documents. This is done by redefining *before* to `\checkoddpagel\ifoddpagel\else\raggedleft\fi` and redefining *after* to `\par`. Note that since 'sidecontcaption', 'sidenamedlegend' and 'sidelegend' all share the same basic internals redefining the two hooks will affect those as well. The user might want to create their own special environment, say, like this:

```

\makeatletter
\newenvironment{sidecaption*}{
  \renewcommand\@mem@scap@beforehook{\checkoddpagel\ifoddpagel\else\raggedleft\fi}
  \renewcommand\@mem@scap@afterhook{\par}
  \begin{sidecaption}}
  {\end{sidecaption}}
\makeatother

```

```

7726 \newcommand\@mem@scap@beforehook{}
7727 \newcommand\@mem@scap@afterhook{}
7728

```

`\@@sidecaption` `\@@sidecaption{<fortoc>}{<title>}[<label>]` is the last macro in the chain and handles all three arguments. This does all the work for `\begin{sidecaption}`

`\m@mscap@fortoc` First, save all the arguments as macros.

```

\m@mscap@forcap 7729 \def\@@sidecaption#1#2[#3]{%
\m@mscaplabel 7730 \ifx\@empty#1\@empty
7731 \def\m@mscap@fortoc{#2}%
7732 \else
7733 \def\m@mscap@fortoc{#1}%
7734 \fi
7735 \def\m@mscap@forcap{#2}%
7736 \ifx\@empty#3\@empty

```

```

7737 \def\m@m@scaplabel{}%
7738 \else
7739 \def\m@m@scaplabel{\@bsphack\label{#3}\@esphack}%
7740 \fi

```

Set the float width, calculate the left margin kern, and start a minipage to hold the float, saving it in box \m@m@scap@fbox.

```

7741 \m@m@scapstart@fbox}
7742

```

\m@m@scapstart@fbox \m@m@scapstart@fbox is the macro that actually sets the float width, calculates the left margin kern, and starts the float's minipage. The macro \m@m@scapend@fbox ends the box.

```

7743 \newcommand*{\m@m@scapstart@fbox}{%
7744 \m@m@scap@beforehook%
7745 \setlength{\m@m@scapmainwidth}{\sidecapfloatwidth}%
7746 \setm@m@scaplkern
7747 \begin{lrbox}{\m@m@scap@fbox}%
7748 \begin{minipage}[c]{\m@m@scapmainwidth}}
7749 \newcommand*{\m@m@scapend@fbox}{%
7750 \end{minipage}%
7751 \end{lrbox}}
7752

```

\endsidecaption \endsidecaption does the work for \end{sidecaption}. Finish the float minipage, then increment the caption counter and call \label via \m@m@scaplabel.

```

7753 \def\endsidecaption{%
7754 \m@m@scapend@fbox
7755 \refstepcounter\@captype
7756 \m@m@scaplabel

```

Set the caption inside a minipage, saving it in box \m@m@scap@capbox.

```

7757 \begin{lrbox}{\m@m@scap@capbox}%
7758 \begin{minipage}[c]{\sidecapwidth}%
7759 \sidecapstyle
7760 \@caption\@captype[\m@m@scap@fortoc]{\m@m@scap@forcap}
7761 \end{minipage}%
7762 \end{lrbox}%

```

Output the float and caption.

```

7763 \m@m@scapopboxes}

```

\m@m@scapopboxes Having determined how high the caption box must be raised with respect to the float box, output the boxes.

```

7764 \newcommand*{\m@m@scapopboxes}{%
7765 \m@m@calcs@capraise

```

Set the float (from box `\m@mscap@fbox`) then the caption (from box `\m@mscap@capbox`) kerning it to the left or right as appropriate.

```

7766 \usebox{\m@mscap@fbox}\m@mscapcheckside
7767 \ifscapmargleft%
7768 \rlap{\kern-\m@mscaplkern
7769 \raisebox{\m@mscapraise}{\usebox{\m@mscap@capbox}}}%
7770 \else%
7771 \rlap{\kern\sidecapsep
7772 \raisebox{\m@mscapraise}{\usebox{\m@mscap@capbox}}}%
7773 \fi

```

Finally, make `\m@mscapthisside` a no-op.

```

7774 \gdef\m@mscapthisside{}%
7775 \@mem@scap@afterhook%
7776 }
7777

```

`\m@mcalscapraise` Calculate the amount the caption might have to be raised wrt the float. This depends on the position:

`t`: raise by the difference in heights

`c`: shouldn't have to do anything as the minipages are meant to center aligned

`b`: lower by the difference in depths

However, experiments showed that a little bit of tweaking might help. The final adjustment, `\sidecapraise` is controlled by the user.

```

7778 \newcommand*{\m@mcalscapraise}{%
7779 \def\@tempb{t}%
7780 \ifx\m@mscappos\@tempb
7781 \settoheight{\m@tempdima}{\strut\usebox{\m@mscap@capbox}}%
7782 \settoheight{\m@mscapraise}{\usebox{\m@mscap@fbox}}%
7783 \advance\m@mscapraise -\m@tempdima
7784 \advance\m@mscapraise 0.5ex
7785 \else
7786 \def\@tempb{b}%
7787 \ifx\m@mscappos\@tempb
7788 \settodepth{\m@tempdima}{\usebox{\m@mscap@fbox}}%
7789 \settodepth{\m@mscapraise}{\strut\usebox{\m@mscap@capbox}}%
7790 \advance\m@mscapraise -\m@tempdima
7791 \else
7792 \m@mscapraise=\z@
7793 \advance\m@mscapraise 0.25ex
7794 \fi
7795 \fi
7796 \advance\m@mscapraise \sidecapraise}
7797

```

`\m@mscapcheckside` This macro determines whether the caption should be set in the left or right margin. In twocolumn documents the caption for a single column float is always set in the adjacent margin. Starred floats are treated as regular floats as in a onecolumn document.

```

7798 \newcommand*{\m@mscapcheckside}{%
7799   \if@twocolumn
7800     \ifdim\hsize=\textwidth% float*
7801       \m@mscapcheckregside
7802     \else
7803       \if@firstcolumn
7804         \scapmarglefttrue
7805       \else
7806         \scapmargleftfalse
7807       \fi
7808     \fi
7809   \else
7810     \m@mscapcheckregside
7811   \fi

```

Finally apply any user's override.

```

7812 \m@mscapthisside}

```

`\m@mscapcheckregside` This performs the margin calculation for onecolumn documents.

```

7813 \newcommand*{\m@mscapcheckregside}{%
7814   \if@twoside
7815     \checkoddpages
7816     \ifnum\m@mscapmarg<\@ne% % left
7817       \scapmarglefttrue
7818     \else
7819       \ifnum\m@mscapmarg=\@ne% % right
7820         \scapmargleftfalse
7821       \else
7822         \ifnum\m@mscapmarg=\tw@% % outer
7823           \scapmarglefttrue
7824         \ifoddpages
7825           \scapmargleftfalse
7826         \fi
7827       \else% % inner
7828         \scapmargleftfalse
7829         \ifoddpages
7830           \scapmarglefttrue
7831         \fi
7832       \fi
7833     \fi
7834   \fi
7835   \else% onside
7836     \scapmarglefttrue
7837     \ifnum\m@mscapmarg>\@ne
7838       \ifnum\m@mscapmarg<\thr@@
7839         \scapmargleftfalse
7840       \fi
7841     \fi
7842   \fi}
7843

```

`\overridescapmargin` User macro to override the calculated caption margin. Call as either:
`\m@mscapthisside \overridescapmargin{left}` or `\overridescapmargin{right}`

```

7844 \newcommand*\overridescapmargin[1]{%
7845   \def\@tempb{#1}\def\@tempa{left}%
7846   \ifx\@tempa\@tempb
7847     \def\m@mscapthisside{\scapmarglefttrue}%
7848   \else
7849     \def\@tempa{right}%
7850     \ifx\@tempa\@tempb
7851       \def\m@mscapthisside{\scapmargleftfalse}%
7852     \else
7853       \@memerror{Argument to \string\overridescapmargin\space neither
7854                 left nor right}{\@ehc}%
7855       \def\m@mscapthisside{}%
7856     \fi
7857   \fi}
7858 \newcommand*\m@mscapthisside{}
7859
```

Now for the other kinds of captions.

`\sidecontcaption` `\sidecontcaption{<title>}[<label>]` is for a continuation sidecaption.

```

7860 \newcommand*\sidecontcaption{%
7861   \@sidecontcaption}

```

`\@sidecontcaption` `\@sidecontcaption{<title>}` grabs the first argument.

```

7862 \def\@sidecontcaption#1{%
7863   \@ifnextchar [{\@@sidecontcaption{#1}}{\@@sidecontcaption{#1} []}]

```

`\@@sidecontcaption` `\@@sidecontcaption{<title>}[<label>]` is the last macro in the chain and handles all two arguments. This does all the work for `\begin{sidecontcaption}`

```

7864 \def\@@sidecontcaption#1[#2]{%
7865   \def\m@mscap@forcap{#1}%
7866   \ifx\@empty#2\@empty
7867     \def\m@mscaplabel{}%
7868   \else
7869     \def\m@mscaplabel{\@bsphack\label{#2}\@esphack}%
7870   \fi
7871   \m@mscapstart@fbox}
7872
```

`\endsidecontcaption` `\endsidecontcaption` does the work for `\end{sidecontcaption}`.

```

7873 \def\endsidecontcaption{%
7874   \m@mscapend@fbox
7875   \addtocounter{\@captype}{\m@ne}\refstepcounter{\@captype}
7876   \m@mscaplabel
7877   \begin{lrbox}{\m@mscap@capbox}%
7878     \begin{minipage}[c]{\sidecapwidth}%
7879     \sidecapstyle

```

```

7880      \@contcaption\@captype{\m@mscap@for cap}
7881      \end{minipage}%
7882      \end{lrbox}%
7883      \m@mscapopboxes}
7884

```

`\sidenamedlegend` `\sidenamedlegend[fortoc]{title}` is for a namedlegend sidecaption.

```

\@sidenamedlegend 7885 \newcommand*{\sidenamedlegend}{%
7886   \@ifnextchar [{\@sidenamedlegend}\@sidenamedlegend[]}%
7887   \def\@sidenamedlegend[#1]#2{%
7888     \@@sidenamedlegend{#1}{#2}}

```

`\@@sidenamedlegend` `\@@sidenamedlegend{fortoc}{title}` is the last macro in the chain and handles all arguments. This does all the work for `\begin{sidenamedlegend}`

```

7889 \def\@@sidenamedlegend#1#2{%
7890   \ifx\@empty#1\@empty
7891     \def\m@mscap@fortoc{#2}%
7892   \else
7893     \def\m@mscap@fortoc{#1}%
7894   \fi
7895   \def\m@mscap@for cap{#2}%
7896   \def\m@mscaplabel{}%
7897   \m@mscapstart@fbox}
7898

```

`\endsidenamedlegend` `\endsidenamedlegend` does the work for `\end{sidenamedlegend}`.

```

7899 \def\endsidenamedlegend{%
7900   \m@mscapend@fbox
7901   \begin{lrbox}{\m@mscap@capbox}%
7902     \begin{minipage}[c]{\sidecapwidth}%
7903       \sidecapstyle
7904       \@legend\@captype[\m@mscap@fortoc]{\m@mscap@for cap}
7905     \end{minipage}%
7906   \end{lrbox}%
7907   \m@mscapopboxes}
7908

```

`\sidelegend` `\sidelegend{title}` is for a legend sidecaption.

```

7909 \newcommand*{\sidelegend}{%
7910   \@@sidelegend}

```

`\@@sidelegend` `\@@sidelegend{title}` is the last macro in the chain and handles all arguments. This does all the work for `\begin{sidelegend}`

```

7911 \def\@@sidelegend#1{%
7912   \def\m@mscap@for cap{#1}%
7913   \m@mscapstart@fbox}
7914

```


`\endsidelegend` `\endsidelegend` does the work for `\end{sidelegend}`.

```

7915 \def\endsidelegend{%
7916   \m@mscapend@fbox
7917   \begin{lrbox}{\m@mscap@capbox}%
7918     \begin{minipage}[c]{\sidecapwidth}%
7919       \sidecapstyle
7920       \legend{\m@mscap@forcap}
7921     \end{minipage}%
7922   \end{lrbox}%
7923   \m@mscapopboxes}
7924

```

19 Epigraphs

This code comes from the `epigraph` package [Wil00a].

`\beforeepigraphskip` The several length commands, which can be changed by the user with

`\afterepigraphskip` `\setlength`.

```

\epigraphwidth 7925 \newlength{\beforeepigraphskip}
\epigraphrule 7926 \setlength{\beforeepigraphskip}{.5\baselineskip}
7927 \newlength{\afterepigraphskip}
7928 \setlength{\afterepigraphskip}{.5\baselineskip}
7929 \newlength{\epigraphwidth}
7930 \setlength{\epigraphwidth}{.4\textwidth}
7931 \newlength{\epigraphrule}
7932 \setlength{\epigraphrule}{.4\p@}

```

`\epigraphsize` The size of the font to be used.

```
7933 \newcommand{\epigraphsize}{\small}
```

`\epigraphflush` The three commands to position epigraphs in the textblock and to position the

`\textflush` components of the epigraph.

```

\sourceflush 7934 \newcommand{\epigraphflush}{flushright}
7935 \newcommand{\textflush}{flushleft}
7936 \newcommand{\sourceflush}{flushright}

```

`\epigraphfontsize` These are declarative forms of the above. It's a bit late now, but the previous

`\epigraphposition` macros should have been internal.

```

\epigraphtextposition 7937 \newcommand{\epigraphfontsize}[1]{\def\epigraphsize{#1}}
\epigraphsourceposition 7938 \newcommand{\epigraphposition}[1]{\long\def\epigraphflush{#1}}
7939 \newcommand{\epigraphtextposition}[1]{\def\textflush{#1}}
7940 \newcommand{\epigraphsourceposition}[1]{\def\sourceflush{#1}}
7941

```

`\@epirule` The internal command to draw a rule between text and source.

```
7942 \newcommand{\@epirule}{\rule[.5ex]{\epigraphwidth}{\epigraphrule}}
```

`\@epitext` The internal command to typeset the $\langle text \rangle$. Put it into a minipage of the right size and typeset per `\textflush`.

```

7943 \newcommand{\@epitext}[1]{%
7944   \begin{minipage}{\epigraphwidth}\begin{\textflush} #1\par
      Draw a rule if it will be visible, otherwise add some extra vertical space.
7945   \ifdim\epigraphrule>\z@ \@epirule \else \vspace*{1ex} \fi
7946   \end{\textflush}\end{minipage}}
```

`\@episource` The internal command for typesetting the $\langle source \rangle$, which is put into a minipage and typeset according to `\sourceflush`.

```

7947 \newcommand{\@episource}[1]{%
7948   \begin{minipage}{\epigraphwidth}
7949     \begin{\sourceflush} #1\par
7950   \end{\sourceflush}\end{minipage}}
7951
```

`\epigraph` Having got the preliminaries out of the way, here's the user command for a single epigraph. This is set in a minipage to prevent breaking across a page. Position it according to `\epigraphflush`.

```

7952 \newcommand{\epigraph}[2]{\vspace{\beforeepigraphskip}
7953   {\epigraphsize\begin{\epigraphflush}\begin{minipage}{\epigraphwidth}
7954     \@epitext{#1}\@ \@episource{#2}
7955   \end{minipage}\end{\epigraphflush}
7956   \vspace{\afterepigraphskip}}}
```

`\qitem` `\qitem` is the epigraph list version of `\item`. Set everything inside a minipage.

`\qitemlabel`

```

7957 \newcommand{\qitem}[2]{%
7958   \raggedright\item \begin{minipage}{\epigraphwidth}
7959     \@epitext{#1}\@ \@episource{#2}
7960   \end{minipage}}}
```

`\qitemlabel` is needed for a list as well. It is not going to typeset anything.

```

7961 \newcommand{\qitemlabel}[1]{\hfill}
```

`epigraphs` Now for the epigraph list. This is defined in terms of a `list` environment.

```

7962 \newenvironment{epigraphs}{%
      Do the vertical space, set the font size, position according to \epigraphflush,
      and put everything into a minipage.
7963   \vspace{\beforeepigraphskip}\begin{\epigraphflush}
7964   \epigraphsize
7965   \begin{minipage}{\epigraphwidth}
7966   \list{}%
      Make the list just fit the minipage (i.e., no indents).
7967   {\itemindent\z@ \labelwidth\z@ \labelsep\z@
7968     \leftmargin\z@ \rightmargin\z@
7969     \let\makelabel\qitemlabel}}%
7970   {\endlist\end{minipage}\end{\epigraphflush}
7971   \vspace{\afterepigraphskip}}
```

19.1 Epigraphs before a chapter title

`\@epichapapp` Commands to drop and restore positions of chapter titles. Dropping is accomplished by inserting vertical space before the `\@chapapp` command.

```
\undodrop 7972 \newcommand{\dropchapter}[1]{%
7973   \let\@epichapapp\@chapapp
7974   \renewcommand{\@chapapp}{\vspace*{#1}\@epichapapp}}
7975 \newcommand{\undodrop}{\let\@chapapp\@epichapapp}
```

Placing an epigraph before a chapter title uses the scheme outlined by Piet van Oostrum [Oos96]. This is to put a zero sized picture into the page header.

`\if@epirhs` Two booleans for testing whether an epigraph is to be at the RH margin, `\if@epicenter` centered, or at the LH margin. The default is RH margin.

```
7976 \newif\if@epirhs      \@epirhstrue
7977 \newif\if@epicenter   \@epicentertrue
```

`\@epipos` This routine sets the `\if@epi...` booleans according to the value of `\epigraphflush`. If `\epigraphflush` is neither `center` nor `flushleft` then it defaults to `flushright`. We have to use this to be upward compatible with `\epigraphflush` being set by the user with `\renewcommand`.

```
7978 \newcommand{\@epipos}{%
7979   \long\def\@ept{flushleft}
7980   \ifx\epigraphflush\@ept
7981     \@epirhsfalse \@epicenterfalse
7982   \else
7983     \long\def\@ept{center}
7984     \ifx\epigraphflush\@ept
7985       \@epirhsfalse \@epicentertrue
7986     \else
7987       \@epirhstrue  \@epicenterfalse
7988     \fi
7989   \fi}
```

`\epigraphhead` `\epigraphhead[<distance>]{<text>}` puts *<text>* at *<distance>* (a number, not a length) below the header at the page position specified by `\epigraphflush`.

```
7990 \newcommand{\epigraphhead}[2][95]{%
```

We have to use `\def` instead of the normal L^AT_EX definition commands as we will keep on (re)defining things. For reasons that are not fully clear to me L^AT_EX doesn't seem to like me using a `\savebox` for storing the epigraph text, so I'll use a command instead.

```
7991   \def\@epitemp{\begin{minipage}{\epigraphwidth}#2\end{minipage}}
```

Define an epigraph page style.

```
7992   \def\ps@epigraph{\let\@mkboth\@gobbletwo
```

There are three possible definitions for `\@oddhead` depending on the value of `\epigraphflush`. We call `\@epipos` to decide which one to do.

```
7993     \@epipos
```

```

7994 \if@epirhs
7995 \def\@oddhead{\hfil\begin{picture}(0,0)
7996 \put(0,-#1){\makebox(0,0)[r]{\@epitemp}}
7997 \end{picture}}
7998 \else
7999 \if@epicenter
8000 \def\@oddhead{\hfil\begin{picture}(0,0)
8001 \put(0,-#1){\makebox(0,0)[b]{\@epitemp}}
8002 \end{picture}\hfil}
8003 \else
8004 \def\@oddhead{\begin{picture}(0,0)
8005 \put(0,-#1){\makebox(0,0)[l]{\@epitemp}}
8006 \end{picture}\hfil}
8007 \fi
8008 \fi
8009 \let\@evenhead\@oddhead
8010 \def\@oddfoot{\reset@font\hfil\thepage\hfil}
8011 \let\@evenfoot\@oddfoot

Make epigraph be the page style for this page.
8012 \thispagestyle{epigraph}}
8013

```

The above produces a plain pagestyle with the epigraph. Life is more complex if someone wants a fancy style with the epigraph. They will have to do some work, though.

`\the@epigraph` `\the@epigraph` is a macro to store the contents of an epigraph, and `\@epidrop` stores the $\langle distance \rangle$ number. `\epigraphforheader` $[\langle distance \rangle]{\langle text \rangle}$ defines the internal macros appropriately.

```

8014 \newcommand{\the@epigraph}{%
8015 \newcommand{\@epidrop}{95}
8016 \newcommand{\epigraphforheader}[2][95]{%
8017 \def\@epidrop{#1}\long\def\the@epigraph{#2}}
8018

```

`\epigraphpicture` `\epigraphpicture` puts `\the@epigraph` into a zero-sized picture at location $(0,-\@epidrop)$. This can then be used as part of a fancy chapter header. The coding is similar to `\epigraphhead`.

```

8019 \newcommand{\epigraphpicture}{%
8020 \def\@epitemp{%
8021 \begin{minipage}{\epigraphwidth}\the@epigraph\end{minipage}}%
8022 \@epipos
8023 \if@epirhs
8024 \begin{picture}(0,0)%
8025 \put(0,-\@epidrop){\makebox(0,0)[r]{\@epitemp}}%
8026 \end{picture}%
8027 \else
8028 \if@epicenter
8029 \begin{picture}(0,0)%

```

```

8030      \put(0,-\@epidrop){\makebox(0,0)[b]{\@epitemp}}%
8031      \end{picture}%
8032  \else
8033      \begin{picture}(0,0)%
8034      \put(0,-\@epidrop){\makebox(0,0)[l]{\@epitemp}}%
8035      \end{picture}%
8036  \fi
8037  \fi}
8038

```

20 The deprecated font commands

The class does not support the old font changing commands unless the `oldfontcommands` option is used.

```

\@memoldfonterr  Macros for old font class error and warning. E.g.,
\@memoldfontwarn \@memoldfonterr{\tt}{\ttfamily}{\texttt}

8039 \newcommand*{\@memoldfonterr}[3]{%
8040   \memerror{Font command \protect#1\space is not supported}{%
8041     Use \protect#2, or \protect#3{...}, or the oldfontcommands option}}
8042 \newcommand*{\@memoldfontwarn}[3]{%
8043   \@memwarn{The \protect#1\space font command is deprecated.
8044     \MessageBreak Use \protect#2{...} or {\protect#3... } instead}}
8045

```

Just give a single warning when an old font command is used following the `oldfontcommands` option.

`\@mem@rmwarn` The old command for roman font.

```

\rm 8046 \if@memoldfont
8047   \def\@mem@rmwarn{\@memoldfontwarn{\rm}{\textrm}{\rmfamily}}
8048   \DeclareOldFontCommand{\rm}{\@mem@rmwarn\gdef\@mem@rmwarn}{%
8049     \normalfont\rmfamily}{\mathrm}
8050 \else
8051   \def\rm{\@memoldfonterr{\rm}{\textrm}{\rmfamily}}
8052 \fi
8053

```

`\@mem@sffwarn` The old sans font command.

```

\sff 8054 \if@memoldfont
8055   \def\@mem@sffwarn{\@memoldfontwarn{\sf}{\textsf}{\sffamily}}
8056   \DeclareOldFontCommand{\sf}{\@mem@sffwarn\gdef\@mem@sffwarn}{%
8057     \normalfont\sffamily}{\mathsf}
8058 \else
8059   \def\sff{\@memoldfonterr{\sf}{\textsf}{\sffamily}}
8060 \fi
8061

```

`\@mem@ttwarn` The old typewriter font command.

```
\tt 8062 \if@memoldfont
      8063 \def\@mem@ttwarn{\@memoldfontwarn{\tt}{\texttt}{\ttfamily}}
      8064 \DeclareOldFontCommand{\tt}{\@mem@ttwarn\gdef\@mem@ttwarn}{%
      8065 \normalfont\ttfamily}{\mathtt}
      8066 \else
      8067 \def\tt{\@memoldfonterr{\tt}{\texttt}{\ttfamily}}
      8068 \fi
      8069
```

`\@mem@bfwarn` The old bold font command.

```
\bf 8070 \if@memoldfont
      8071 \def\@mem@bfwarn{\@memoldfontwarn{\bf}{\textbf}{\bfseries}}
      8072 \DeclareOldFontCommand{\bf}{\@mem@bfwarn\gdef\@mem@bfwarn}{%
      8073 \normalfont\bfseries}{\mathbf}
      8074 \else
      8075 \def\bf{\@memoldfonterr{\bf}{\textbf}{\bfseries}}
      8076 \fi
      8077
```

`\@mem@itwarn` The old italic font command.

```
\it 8078 \if@memoldfont
      8079 \def\@mem@itwarn{\@memoldfontwarn{\it}{\textit}{\itshape}}
      8080 \DeclareOldFontCommand{\it}{\@mem@itwarn\gdef\@mem@itwarn}{%
      8081 \normalfont\itshape}{\mathit}
      8082 \else
      8083 \def\it{\@memoldfonterr{\it}{\textit}{\itshape}}
      8084 \fi
      8085
```

`\@mem@slwarn` The old slanted font command.

```
\sl 8086 \if@memoldfont
      8087 \def\@mem@slwarn{\@memoldfontwarn{\sl}{\textsl}{\slshape}}
      8088 \DeclareOldFontCommand{\sl}{\@mem@slwarn\gdef\@mem@slwarn}{%
      8089 \normalfont\slshape}{\@nomath\sl}
      8090 \else
      8091 \def\sl{\@memoldfonterr{\sl}{\textsl}{\slshape}}
      8092 \fi
      8093
```

`\@mem@scwarn` The old small caps font command.

```
\sc 8094 \if@memoldfont
      8095 \def\@mem@scwarn{\@memoldfontwarn{\sc}{\textsc}{\scshape}}
      8096 \DeclareOldFontCommand{\sc}{\@mem@scwarn\gdef\@mem@scwarn}{%
      8097 \normalfont\scshape}{\@nomath\sc}
      8098 \else
      8099 \def\sc{\@memoldfonterr{\sc}{\textsc}{\scshape}}
      8100 \fi
      8101
```

`\@mem@calwarn` The old calligraphic font command.

```

\cal 8102 \if@memoldfont
      8103 \def\@mem@calwarn{%
      8104 \@memwarn{The \protect\cal\space font command is deprecated.
      8105 \MessageBreak Try to use \protect\mathcal\space instead}}
      8106 \DeclareRobustCommand*\cal{\@mem@calwarn\gdef\@mem@calwarn{}}%
      8107 \@fontswitch\relax\mathcal}
      8108 \else
      8109 \def\cal{%
      8110 \@memerror{Font command \protect\cal\space is not supported}{%
      8111 Use \protect\mathcal, or the oldfontcommands option}}
      8112 \fi
      8113

```

`\@mem@mitwarn` The old math italic font command.

```

\mit 8114 \if@memoldfont
      8115 \def\@mem@mitwarn{%
      8116 \@memwarn{The \protect\mit\space font command is deprecated.
      8117 \MessageBreak Try to use \protect\mathnormal\space instead}}
      8118 \DeclareRobustCommand*\mit{\@mem@mitwarn\gdef\@mem@mitwarn{}}%
      8119 \@fontswitch\relax\mathnormal}
      8120 \else
      8121 \def\mit{%
      8122 \@memerror{Font command \protect\mit\space is not supported}{%
      8123 Use \protect\mathnormal, or the oldfontcommands option}}
      8124 \fi
      8125

```

`\em` The old emphasis font command (the original `\em` is defined in the kernel file `\eminnershape ltfssini.dtx`, coded here as `\@m@m@m`).

```

\emph 8126 \DeclareRobustCommand{\em}{%
      8127 \@nomath\em
      8128 \ifdim\fontdimen\@ne\font > \z@
      8129 \eminnershape
      8130 \else
      8131 \itshape
      8132 \fi}
      8133 \providecommand{\eminnershape}{\upshape}
      8134 \DeclareTextFontCommand{\emph}{\em}
      8135

```

21 Cross Referencing

21.1 Label referencing

`\fref` These are named references to labeled figures, tables and pages. I find these
`\tref` useful to ensure consistency throughout the document — I don't have to
`\pref` remember whether it is 'see Figure ...' or 'figure' or 'Fig.' or ...

```

8136 \newcommand*{\fref}[1]{\figurerefname~\ref{#1}}
8137 \newcommand*{\tref}[1]{\tablerefname~\ref{#1}}
8138 \newcommand*{\pref}[1]{\pagerefname~\pageref{#1}}

```

`\Aref` These are named references to labeled Part, Chapter and Sectional divisions.

```

\Bref 8139 \newcommand*{\Aref}[1]{\appendixrefname~\ref{#1}}
\Pref 8140 \newcommand*{\Bref}[1]{\bookrefname~\ref{#1}}
\Cref 8141 \newcommand*{\Pref}[1]{\partrefname~\ref{#1}}
\Sref 8142 \newcommand*{\Cref}[1]{\chapterrefname~\ref{#1}}
      8143 \newcommand*{\Sref}[1]{\sectionrefname~\ref{#1}}
8144

```

21.2 Title referencing

This is based on DA's `titleref` package [Ars01a]. The following remarks are taken from that package.

Titles for numbered sectioning units and for floats with captions are the same as the respective TOC, LOF, or LOT entry (even when no TOC is printed). When a short title is provided (`\section[short]{long}`) it will be used for the `\titleref`. This is especially useful for figure captions. Unnumbered sections take their title reference from the printed title. Beware! This turns the title into a moving argument when it normally is not, and will cause weird errors if there are 'fragile' commands present.

Enumerated lists do not have titles and simply inherit the title of their section.

The format of the title reference is controlled by the command

`\theTitleReference`, which can be redefined with `\renewcommand`. It takes two parameters: the number and the title. The number is just the regular `\ref` and it is **WRONG** in unnumbered sections. Beware! The default definition is the unadorned title. You could do, for example,

```
\renewcommand{\theTitleReference}[2]{#1\ \emph{#2}}.
```

The title of the current section is also available without `\label`: Use the command `\currenttitle` to generate a `\titleref` to the current section. If you have redefined `\theTitleReference` to print the number with the title, be aware that the two may not correspond: In a numbered list the number will show the current item but the title will show the current section, but in an unnumbered section the number will show some previous section number.

The big problem with the `[usetoc]` method is that `\titleref` will not refer to a `\label` which was given in the title or caption itself; the label must be placed after the sectioning command or the caption. If you make a title-reference to a label given in a title, you will get a warning message.

```

\ifheadnameref \ifheadnameref: FALSE use ToC entry, TRUE use header entry
\headnameref 8145 \newif\ifheadnameref
\tocnameref 8146 \newcommand*{\headnameref}{\headnamereftrue}
      8147 \newcommand*{\tocnameref}{\headnamerfalse}
8148 \tocnameref

```



```

\theTitleReference \theTitleReference{<num>}{<title>} is the style for typesetting a referenced
(number and) title.

8149 \newcommand{\theTitleReference}[2]{#2}
8150

\label This redefinition of \label is intended to work with other redirections of
\label, if they record extra information in a similar way.

8151 \let\@mem@old@label\label
8152 \def\label#1{\@bsphack\begingroup
8153   \protected@edef\@currentlabel{\protect\M@TitleReference
8154     {\@currentlabel}{\M@currentTitle}}%
8155   \@mem@old@label{#1}%
8156   \endgroup \@esphack}%
8157

\@mem@@getttitle \@mem@@getttitle{<title>} grabs a title text.

8158 \def\@mem@@getttitle#1{\begingroup \let\protect\@unexpandable@protect
8159   \let\label\@mem@nestwarn
8160   \let\index\@gobble \let\glossary\@gobble
8161   \let\markboth\@gobbletwo \let\@mkboth\@gobbletwo
8162   \let\markright\@gobble
8163   \edef\@tempa{\noexpand\def\noexpand\M@currentTitle{#1}}%
8164   \expandafter\endgroup\@tempa}
8165

\@mem@nestwarn
\M@TitleReference 8166 \let\@mem@nestwarn\@gobble
8167 \let\M@TitleReference\@firstoftwo
8168

\titleref \titleref{<key>} prints the title corresponding to \label{key}.
\currenttitle \currenttitle prints the latest title. \titleref extended to support a starred
\@mem@titleref version that will not provide hyperlinks when using hyperref. The hyperref
\@mem@titlerefno link definition of \@mem@titlerefno link is found in memhfixc.

8169 \newcommand*\@mem@titleref[1]{\begingroup
8170   \let\numberline\@gobble
8171   \let\M@TitleReference\@mem@theTR % interrupt recursion of \ref
8172   \ref{#1}\endgroup}
8173 \let\@mem@titlerefno link\@mem@titleref
8174
8175 \DeclareRobustCommand\titleref{\@ifstar{\@mem@titlerefno link}{\@mem@titleref}}
8176
8177 \DeclareRobustCommand\currenttitle{\begingroup
8178   \let\numberline\@gobble
8179   \theTitleReference\@currentlabel\M@currentTitle\endgroup}
8180

% \DeclareRobustCommand\@mem@nestwarn[1]{\@mem@warn%
% {Label \string"#1\string" was put in a title,\MessageBreak

```

```

% so the \noexpand\titleref is incorrect}}

\M@currentTitle
8181 \let\M@currentTitle\@empty
8182

\@mem@theTR proper definition:
8183 \def\@mem@theTR{\let\M@TitleReference\@firstoftwo \theTitleReference}
8184

\namerefon Named references has turned some arguments (e.g., \legend{text}) into moving
\namerefoff ones (Sven.Hartrumpf@FernUni-Hagen.de reported a problem but not a cause,
March 2003). Not everyone needs named references.
8185 \newcommand*\namerefon{\let\M@getttitle\@mem@getttitle}
8186 \newcommand*\namerefoff{\let\M@getttitle\@gobble}
8187 \namerefon

```

22 Table of Contents, etc.

A `\section` command writes a `\contentsline{section}{\langle title \rangle}{\langle page \rangle}` command on the `.toc` file, where `\langle title \rangle` contains the contents of the entry and `\langle page \rangle` is the page number. If sections are being numbered, then `\langle title \rangle` will be of the form `\numberline{\langle num \rangle}{\langle heading \rangle}` where `\langle num \rangle` is the number produced by `\thesection`. Other sectioning commands work similarly.

A `\caption` command in a ‘figure’ environment writes `\contentsline{figure}{\numberline{\langle num \rangle}{\langle caption \rangle}}{\langle page \rangle}` on the `.lof` file, where `\langle num \rangle` is the number produced by `\thefigure` and `\langle caption \rangle` is the figure caption. It works similarly for a ‘table’ environment. The command `\contentsline{\langle name \rangle}` expands to `\l@{\langle name \rangle}`. So, to specify the table of contents, we must define `\l@chapter`, `\l@section`, `\l@subsection`, ...; to specify the list of figures, we must define `\l@figure`; and so on. Most of these can be defined with the `\@dottedtocline` command, which works as follows.

```
\@dottedtocline{\langle level \rangle}{\langle indent \rangle}{\langle numwidth \rangle}{\langle title \rangle}{\langle page \rangle}
```

`\langle level \rangle` An entry is produced only if `\langle level \rangle <=` value of the `tocdepth` counter.

Note, `\chapter` is level 0, `\section` is level 1, etc.

`\langle indent \rangle` The indentation from the outer left margin of the start of the contents line.

`\langle numwidth \rangle` The width of a box in which the section number is to go, if `\langle title \rangle` includes a `\numberline` command.

```

\@pnumwidth This command uses the following three parameters, which are set with a
\@tocrmarg \newcommand (so em’s can be used to make them depend upon the font).
\@dotsep

```

`\@pnumwidth` The width of a box in which the page number is put.

`\@tocrmarg` The right margin for multiple line entries. One wants `\@tocrmarg`
 \geq `\@pnumwidth`

`\@dotsep` Separation between dots, in mu units. Should be defined as a number
 like 2 or 1.7

```
8188 \newcommand{\@pnumwidth}{1.55em}
8189 \newcommand{\@tocrmarg}{2.55em}
8190 \newcommand{\@dotsep}{4.5}
```

`\tocentryskip` We define two lengths and a utility command.

```
\tocbaseline 8191 \newlength{\tocentryskip} \setlength{\tocentryskip}{1em}
\tocskip 8192 \newlength{\tocbaseline} \setlength{\tocbaseline}{20pt}
8193 \newcommand{\tocskip}[1]{%
8194   \addtocontents{toc}{\protect\vspace{#1}}}
```

22.1 New List of ...

It is apparent that users want to do at least two things that are not readily provided by the standard classes: (a) change the appearance of the Table of Contents, etc., headings, and (b) create new List of ... This class provides a means of creating new Lists whose headings are parameterized, thereby killing two birds with one stone.

In the standard classes the tables of contents, figures etc. are always set in single-column style. In this class you can choose one- or two-column ToCs, etc. The titles are added to the ToC, unless the starred versions of the commands are used.

`\ensureonecol` These two macros cooperate to switch from two-columns to one column, and
`\restorefromonecol` back again.

`\onecoltocetc` These macros define `\ensureonecol` and `\restorefromonecol` such that

- `\twocoltocetc` • `\onecoltocetc` ToCs, etc., will be set in one column (the default)
- `\doccoltocetc` • `\twocoltocetc` ToCs, etc., will be set in two columns
- `\doccoltocetc` ToCs, etc., will be set in column(s) corresponding to the document's (onecolumn/twocolumn) option.

```
8195 \newcommand*{\onecoltocetc}{%
8196   \def\ensureonecol{%
8197     \if@twocolumn
8198       \@restonecoltrue\onecolumn
8199     \else
8200       \@restonecolfalse
8201     \fi}%
8202   \def\restorefromonecol{\if@restonecol\twocolumn\fi}}
```

```

8203 \newcommand*{\twocoltocetc}{%
8204   \def\ensureonecol{%
8205     \if@twocolumn
8206       \@restonecoltrue
8207     \else
8208       \@restonecolfalse\twocolumn
8209     \fi}%
8210   \def\restorefromonecol{\if@restonecol\else\onecolumn\fi}}
8211 \newcommand*{\doccoltocetc}{%
8212   \let\ensureonecol\relax
8213   \let\restorefromonecol\relax}

```

Set the default

```

8214 \onecoltocetc
8215

```

`\cftparskip` The `\parskip` local to the ToC, etc, is set to the length `\cftparskip`.

```

8216 \newlength{\cftparskip}
8217 \setlength{\cftparskip}{0pt}
8218

```

`\newlistof` `\newlistof{<listofcmmid>}{<ext>}{<listofname>}` creates the command `\listofcmmid` to typeset a new List of, where the external file has the extension `.ext` and the heading title is `<listofname>`. The code for this is a heavily modified part of the `tocloft` package.

```

8219 \newcommand{\newlistof}[3]{%

```

In the following, X stands for the value of `<listofcmmid>` and Z stands for the value of `<ext>`.

`\ext@Z` The file extension and listing depth, which is set to level 1.

```

\Zdepth 8220   \@namedef{ext@#2}{#2}
8221   \@ifundefined{c@#2depth}{\newcounter{#2depth}}{}
8222   \setcounter{#2depth}{1}

```

`\Zmark` The heading marks for the listing.

```

8223   \@namedef{#2mark}{\markboth{#3}{#3}}

```

`\X` Typeset the listing title and entries, with both a normal and starred version.

```

8224   \@namedef{#1}{\@ifstar{\@nameuse{mem@#1}{01}}{\@nameuse{mem@#1}{00}}}

```

`\mem@X` In earlier version of the class, we used two macros `\@starZ` and `\@plainZ`, to reduce code we now only use one, the argument of it takes care of differentiating between the stuff that are different. Two hooks is added just before and after importing the list file contents.

```

8225   \@namedef{cft#2beforelisthook}{}%
8226   \@namedef{cft#2afterlisthook}{}%
8227   \@namedef{mem@#1}##1{%
8228     \ensureonecol

```

```

8229 \par
8230 \begingroup
8231 \nameuse{@#2maketitle}
8232 \if##1
8233 \ifmem@em@starred@listof\else
8234 \phantomsection
8235 \addcontentsline{toc}{chapter}{#3}
8236 \fi
8237 \fi
8238 \parskip\cftparskip
8239 \nameuse{cft#2beforelisthook}%
8240 \@starttoc{#2}%
8241 \nameuse{cft#2afterlisthook}%
8242 \endgroup
8243 \restorefromonecol}

```

`\@Zmaketitle` This macro typesets the title.

```

8244 \@namedef{@#2maketitle}{%
8245 \nameuse{#2headstart}
8246 {\parindent\z@
8247 %%% \parskip\cftparskip
8248 \interlinepenalty\@M
8249 \nameuse{print#2nonum}%
8250 \nameuse{print#2title}{#3}%
8251 \nameuse{#2mark}%
8252 \thispagestyle{chapter}%
8253 \nameuse{after#2title}
8254 }
8255 \@afterheading}

```

`\Zheadstart` The macros `\Zheadstart` and `\afterZtitle` control what goes before and after the title. They default to the corresponding macros for chapters.

```

8256 \@namedef{#2headstart}{\chapterheadstart}
8257 \@namedef{after#2title}{\afterchaptertitle}

```

`\printZnonum` This typesets something before the title and defaults to `\printchapternonum`.

```

8258 \@namedef{print#2nonum}{\printchapternonum}

```

`\printZtitle` The title is typeset by the macro `\printZtitle{<title>}` which defaults to `\printchaptertitle`.

```

8259 \@namedef{print#2title}##1{\printchaptertitle{##1}}

```

This is the end of the definition of `\newlistof`.

```

8260 } % end \newlistof
8261

```

`\ifmem@em@starred@listof` The class handles the created ‘new lists’ a little different than other classes. We automatically add it to the main table of contents. This is usually what one

wants. If not, one can always use the starred version. But in some cases other packages will use `\tableofcontents` to typeset their own ‘list of...’, in which case it becomes a little hard to add a star. Therefore we add a switch that can be used to emulate the starred list of. After the macro `\KeepFromToc` the memoir created (non-starred) ‘list of’ will behave as their starred counterpart.

```
8262 \newif\ifmem@em@starred@listof
8263 \newcommand\KeepFromToc{\mem@em@starred@listoftrue}
```

Tip: use `\KeepFromToc` as an environment, i.e.

```
\begin{KeepFromToc}
\listoffigures
\end{KeepFromToc}
```

to locally remove a list from the TOC.

`\@starttoc` A list of macro calls `\@starttoc` to read the appropriate file. I have changed the kernel definition to allow a file to be read multiple times by delaying killing the file until the end of the document.

```
8264 \renewcommand{\@starttoc}[1]{%
8265   \begingroup\makeatletter
8266     \@input{jobname.#1}%
8267     \if@files
8268       \AtEndDocument{%
```

If a ToC, or other ListOf, is called more than once then we have to stop opening yet another, redundant, output file. The check against `\relax` seems to meet the requirement.

```
8269       \expandafter\ifx\csname tf@#1\endcsname\relax
8270       \expandafter\newwrite\csname tf@#1\endcsname
8271       \immediate\openout \csname tf@#1\endcsname \jobname.#1\relax
8272       \fi
8273     }%
8274   \fi
8275   \@nobreakfalse
8276   \endgroup}
8277
```

22.2 Table of Contents

`\tableofcontents` These macros request that L^AT_EX produces a table of contents. The ToC heading `\tableofcontents*` is added to the ToC unless the starred version is used.

```
8278 \newlistof{tableofcontents}{toc}{\contentsname}
```

The List of Figures and List of Tables are defined later.

22.3 List entries

Each command in the body of the text that makes an entry in the ToC, or LoF or LoT needs an additional macro to format the entry, as described above. Users often want to change the formatting of the entries but it is not immediately obvious how to do that.

I have borrowed and modified more of the code from the `tocloft` package to enable easy creation and modification the formatting of the entries.

```
\setpnumwidth User commands for setting \@pnumwidth and \@tocrmarg.
\setrmarg 8279 \newcommand*\setpnumwidth[1]{\renewcommand{\@pnumwidth}{#1}}
8280 \newcommand*\setrmarg[1]{\renewcommand{\@tocrmarg}{#1}}

\cftdot In the default ToC, a dotted line can be used to provide a leader between a title
\cftdotfill and the page number. The definition of this leader is buried in the
\@dottedtocline command. The \cftdotfill{<sep>} command provides a
parameterised version of the leader code, where <sep> is the separation between
the dots in mu units. The symbol used for the ‘dots’ in the leader is given by the
value of \cftdot.
8281 \providecommand{\cftdot}{.}
8282 \providecommand{\cftdotfill}[1]{%
8283 \leaders\hbox{$\m@th\mkern #1 mu\hbox{\cftdot}\mkern #1 mu$}\hfill}

\cftdotsep \cftdotsep holds the default dot separation. If the kerns in \cftdotfill are
\cftnodots large enough, then no dots will be printed. \cftnodots should be ‘large enough’.
8284 \providecommand{\cftdotsep}{4.5}
8285 \newcommand{\cftnodots}{2000}
```

Now for the trickier bits regarding the typesetting of the ToC entries.

A `.toc` (also `.lof` and `.lot`) file consists of a list of

```
\contentsline{<kind>}{<title>}{<page>}
```

commands, where `<kind>` is the kind of heading (e.g., `part` or `section` or `figure`), `<title>` is the title text (including the number), and `<page>` is the page number. The entries are inserted into the file by calling the

```
\addcontentsline{<file>}{<kind>}{<title>}
```

command, where `<file>` is the file extension (e.g., `toc`, `lot`) and the other arguments are the same as for the `\contentsline` command. (Arbitrary stuff may also be put into the file via the `\addtocontents{<file>}{<text>}` command). The typesetting of the `\contentsline` entries is performed by commands of the form `\l@kind`. The sectioning and captioning commands call `\addcontentsline` to insert their titles into the `.toc` etc., files.

For the purposes at hand it is generally impossible to treat the typesetting of a title and its number separately, as both are bundled into the `<title>` argument within `\contentsline`. They could be handled separately if the `\contentsline` command was suitably modified. If this was done, then the `\addtocontentsline` command would also need to be changed which would then require the sectioning and captioning commands to be modified as well. This is certainly possible, but

would cause problems if any other package also modified the sectioning or captioning commands, and there are several packages which do this. I provide modified versions of the `\l@kind` commands. Essentially, my new definitions consist of inlined versions of the code for `\@dottedtocline`.

```
\cftparrfillskip The \l@kind commands modify (locally) the value of \parfillskip.
\cftparrfillskip is a copy of the default The TEXbook \parfillskip definition.
8286 \newcommand*\cftparrfillskip{\parfillskip=0pt plus1fil}

\@cftn@me Lars Madsen suggested that macros like \cftfigurename be added to entries in
the LoF, and so on.
8287 \newcommand*\@cftn@me{}
8288

\numberline The purpose of the \numberline{<secnum>} command is to typeset <secnum>
\numberlinehook left justified in a box of width \@tempdima. I redefine it to add three additional
parameters, namely \@cftbsnum, \@cftasnum and \@cftasnumb (see
ltsect.dtx for the original definition). We also add a hook, that initially does
nothing, but might be redefined to record say withs of <secnum>s.
8289 \newcommand*\numberlinehook[1]{}
8290 \renewcommand*\numberline[1]{%
8291   \numberlinehook{#1}%
8292   \hb@xt@\@tempdima{\@cftn@me\@cftbsnum #1\@cftasnum\hfil}\@cftasnumb}

\@cftbsnum
\@cftasnum 8293 \newcommand*\@cftbsnum{}
\@cftasnumb 8294 \newcommand*\@cftasnum{}
8295 \newcommand*\@cftasnumb{}

\newlistentry \newlistentry[<within>]{<counter>}{<ext>}{<level-1>} creates a set of
commands for typesetting a new kind of entry in a List of. <counter> is the name
of the counter for the entry and must be the same as the name of the entry (e.g.,
subsection). The optional <within> is the name of a counter within which
counter is defined. The file extension for the List of is <ext> and <level-1> is one
less than the level of the entry in the List of.
8296 \newcommand{\newlistentry}[4][\@empty]{%
  In the following, X is used as the value of <counter> and Z as the value of <ext>.

  \c@X Check if <within> and <counter> have been defined. It is an error if <within> has
  \theX not been defined. <counter> will be created if it has not been previously defined.
  Set the default counter values.
8297   \ifundefined{c@#2}{%      check & set the counter
8298     \ifx \@empty#1\relax
8299       \newcounter{#2}
8300     \else
8301       \newcounter{#2}[#1]%
8302       \expandafter\edef\csname the#2\endcsname{%
```



```

8303 \expandafter\noexpand\csname the#1\endcsname.\noexpand\arabic{#2}}
8304 \fi}{
8305 \setcounter{#2}{0}

```

That finishes off the error checking. No matter what the result, the rest of the new commands are defined.

`\l@X \l@X{<title>}{<page>}` typesets the entry.

```
8306 \@namedef{l@#2}##1##2{%
```

Only typeset if the `\Zdepth` is greater than `<level-1>`.

```
8307 \ifnum \@nameuse{c@#3depth} > #4\relax
```

Add some vertical space.

```
8308 \vskip \@nameuse{cftbefore#2skip}
```

Start a group to keep paragraphing changes local. Set the `\leftskip` to the entry's indentation.

`\cftwhatismyname` `\cftwhatismyname` locally stores the type of toc entry. Useful for hooks into general macros like `\numberline`.

```
8309 {\leftskip \@nameuse{cft#2indent}\relax
```

```
8310 \newcommand*\cftwhatismyname{#2}%
```

```
8311 \memRTLleftskip \@nameuse{cft#2indent}\relax
```

Set the `\rightskip` to `\@tocrmarg` to leave room for the page number.

```
8312 %%% \rightskip \@tocrmarg
```

```
8313 \memRTLrightskip \@tocrmarg
```

Ensure that the last line of an entry will be filled. Setting `\parfillskip` to a negative value prevents 'overfull box' messages.

```
8314 %%% \parfillskip -\rightskip
```

```
8315 \parfillskip -\memRTLrightskip
```

Set the paragraph's indentation to the entry's indentation.

```
8316 \parindent \@nameuse{cft#2indent}\relax\@afterindenttrue
```

Try and prevent breaks between lines in a multiline entry.

```
8317 \interlinepenalty\@M
```

Make sure we have left vertical mode.

```
8318 \leavevmode
```

Our version of `\numberline` expects that the width of the number box is in `\@tempdima`, and that the three macros `\cftbsnum`, `\cftaqsnum`, and `\cftasnumb` are defined. We set all these to the values for this entry.

```
8319 \settowidth{\@tempdima}{\@nameuse{cft#2font}\@nameuse{cft#2name}}%
```

```
8320 \addtolength{\@tempdima}{\@nameuse{cft#2numwidth}}%
```

```
8321 \expandafter\let\expandafter\cftbsnum\csname cft#2presnum\endcsname
```

```
8322 \expandafter\let\expandafter\cftasnum\csname cft#2aftersnum\endcsname
```

```
8323 \expandafter\let\expandafter\cftasnumb\csname cft#2aftersnumb\endcsname
```

```
8324 \expandafter\let\expandafter\cftn@me\csname cft#2name\endcsname
```

Arrange that the (entry number and) first line of the title is set at the current indent, and that any subsequent lines will be further indented.

```

8325 %%%          \advance\leftskip\@tempdima \null\nobreak\hskip -\leftskip
8326          \advance\memRTLleftskip\@tempdima \null\nobreak\hskip -\memRTLleftskip

Print the (number and) title, prohibiting any breaking.
8327          {\@nameuse{cft#2font}##1}\nobreak

Print the leader and the page number, and then close the group.
8328          \@nameuse{cft#2fillnum}{##2}}
8329      \fi
8330 }% end of \l@#2
8331

```

Now define all the layout commands used by \l@X. The default values of these print the entry in a normal font with a dotted line between the title and the page number.

\cftbeforeXskip The skip before the title.

```

8332 \expandafter\newlength\csname cftbefore#2skip\endcsname
8333 \setlength{\@nameuse{cftbefore#2skip}}{\z@ \@plus .2\p@}

```

\cftXindent The indent and width for the number.

```

\cftXnumwidth 8334 \expandafter\newlength\csname cft#2indent\endcsname
8335 \expandafter\newlength\csname cft#2numwidth\endcsname

```

Set the default values for the indent and numwidth depending on the entry's level. A level of 1 corresponds to a figure entry (no indent, and space for a number like N.N).

```

8336 \ifcase #4\relax % 0 (level 1)
8337 \setlength{\@nameuse{cft#2indent}}{0em}
8338 \setlength{\@nameuse{cft#2numwidth}}{2.3em}
8339 \or % 1 (level 2)
8340 \setlength{\@nameuse{cft#2indent}}{2.3em}
8341 \setlength{\@nameuse{cft#2numwidth}}{3.2em}
8342 \or % 2 (level 3)
8343 \setlength{\@nameuse{cft#2indent}}{5.5em}
8344 \setlength{\@nameuse{cft#2numwidth}}{4.1em}
8345 \or % 3 (level 4)
8346 \setlength{\@nameuse{cft#2indent}}{8.5em}
8347 \setlength{\@nameuse{cft#2numwidth}}{5.0em}
8348 \else % anything else
8349 \setlength{\@nameuse{cft#2indent}}{10.5em}
8350 \setlength{\@nameuse{cft#2numwidth}}{6.0em}
8351 \fi

```

\cftXfont And the remaining commands; the only ones that are not null are for the dotsep,

\cftXname the font and the leader

```

\cftXpresnum 8352 \@namedef{cft#2font}{\normalfont}
\cftXaftersnum 8353 \@namedef{cft#2name}{}
\cftXaftersnumb
\cftXdotsep
\cftXleader
\cftXpagefont
\cftXafterpnum

```

```

8354 \namedef{cft#2presnum}{%
8355 \namedef{cft#2aftersnum}{%
8356 \namedef{cft#2aftersnumb}{%
8357 \namedef{cft#2dotsep}{\cftdotsep}
8358 \namedef{cft#2leader}{\normalfont\cftdotfill{\@nameuse{cft#2dotsep}}}%
8359 \namedef{cft#2pagefont}{\normalfont}%
8360 \namedef{cft#2afterpnum}{%

```

`\toclevel@X` The hyperref package needs a command `\toclevel@X`, holding the *<level-1>* value.

```

8361 \namedef{toclevel@#2}{#4}

```

`\cftXformatpnum` Typeset the leader and page number. We add a hook into the formatting of the page number. This might later be used to record the widths of the numbers
`\cftXformatpnumhook` used. It will be given the page number as its argument. Does nothing by default.
`\cftXfillnum`

```

8362 \namedef{cft#2formatpnumhook}##1{%
8363 \namedef{cft#2formatpnum}##1{%
8364 \@nameuse{cft#2formatpnumhook}{##1}%
8365 \hb@xt@{\pnumwidth}{\hfil\@nameuse{cft#2pagefont}##1}}
8366 \namedef{cft#2fillnum}##1{%
8367 {\@nameuse{cft#2leader}}\nobreak
8368 %%% \hb@xt@{\pnumwidth}{%
8369 %%% \hfil\@nameuse{cft#2pagefont}##1}
8370 \@nameuse{cft#2formatpnum}{##1}%
8371 \@nameuse{cft#2afterpnum}\par}

```

This ends the definition of `\newlistentry`.

```

8372 } % end \newlistentry
8373

```

`\cftsetindents` `\cftsetindents{<entry>}{<indent>}{<numwidth>}` sets the *indent* and *numwidth* for entry *<entry>*.

```

8374 \newcommand*\cftsetindents[3]{%
8375 \setlength{\@nameuse{cft#1indent}}{#2}
8376 \setlength{\@nameuse{cft#1numwidth}}{#3}}
8377

```

`\cftbookname` If you want a book entry in the ToC like:

BOOK I Title ...

then `\renewcommand*\cftbookname}{BOOK~}`

```

8378 \newcommand*\cftbookname{}
8379

```

`\cftbookbreak` Now for the ToC entry.

```

\l@book 8380 \newcommand*\cftbookbreak{\addpenalty{-\@highpenalty}%

```

```

\booknumberline 8381 \addvspace{\cftbeforebookskip}}

```

```

\booknumberlinehook 8382 \newcommand*\l@book[2]{%

```

```

8383 \ifnum\c@tocdepth >-3\relax

```

```

8384 \cftbookbreak
8385 \begingroup
8386 {%\leftskip \cftbookindent\relax
8387 \memRTLleftskip \cftbookindent\relax
8388 %%% \rightskip \@toCRMarg
8389 \memRTLrightskip \@toCRMarg
8390 %%% \parfillskip -\rightskip
8391 \parfillskip -\memRTLrightskip
8392 \parindent \cftbookindent\relax\@afterindenttrue
8393 \interlinepenalty\@M
8394 \leavevmode
8395 \settowidth{\@tempdima}{\cftbookfont\cftbookname}%
8396 \addtolength{\@tempdima}{\cftbooknumwidth}%
8397 \let\@cftbsnum \cftbookpresnum
8398 \let\@cftasnum \cftbookaftersnum
8399 \let\@cftasnumb \cftbookaftersnumb
8400 %%% \advance\leftskip \@tempdima \null\nobreak\hskip -\leftskip
8401 \advance\memRTLleftskip \@tempdima \null\nobreak\hskip -\memRTLleftskip
8402 {\cftbookfont #1}%
8403 \cftbookfillnum{#2}}
8404 \nobreak
8405 \global\@nobreaktrue
8406 \everypar{\global\@nobreakfalse\everypar{}}%
8407 \endgroup
8408 \fi}
8409 \newcommand*\booknumberlinehook[1]{}
8410 \newcommand{\booknumberline}[1]{%
8411 \booknumberlinehook{#1}%
8412 \hb@xt@\@tempdima{%
8413 \cftbookname\@cftbsnum #1\@cftasnum\hfil}\@cftasnumb}}%\space}
8414

\cftbeforebookskip
\cftbookindent 8415 \newlength{\cftbeforebookskip}
\cftbooknumwidth 8416 \setlength{\cftbeforebookskip}{2.25em \@plus\p@}
\cftbookfont 8417 \newdimen\cftbookindent
\cftbookpresnum 8418 \setlength{\cftbookindent}{0em}
\cftbookaftersnum 8419 \newdimen\cftbooknumwidth
\cftbookaftersnumb 8420 \setlength{\cftbooknumwidth}{1.5em}
\cftbookleader 8421 \newcommand*\cftbookfont{\large\bfseries}
\cftbookdotsep 8422 \newcommand*\cftbookpresnum{}
\cftbookpagefont 8423 \newcommand*\cftbookaftersnum{}
\cftbookafterpnum 8424 \newcommand*\cftbookaftersnumb{}
\cftbookfillnum 8425 \newcommand*\cftbookleader{%
8426 \large\bfseries\cftdotfill{\cftbookdotsep}}
\cftbookformatpnum 8427 \newcommand*\cftbookdotsep{\cftnodots}
\cftbookformatpnumhook 8428 \newcommand*\cftbookpagefont{\large\bfseries}
8429 \newcommand{\cftbookafterpnum}{}
8430 \newcommand{\cftbookfillnum}[1]{%
8431 {\cftbookleader}%

```

```

8432 %%% {\hb@xt@{\pnumwidth{\hss {\cftbookpagefont #1}}}%
8433 \cftbookformatpnum{#1}%
8434 \cftbookafterpnum\par}
8435 \newcommand{\cftbookformatpnumhook}[1]{}
8436 \newcommand{\cftbookformatpnum}[1]{%
8437 \cftbookformatpnumhook{#1}%
8438 \hb@xt@{\pnumwidth{\hss {\cftbookpagefont #1}}}
8439

```

`\cftpartname` If you want a part entry in the ToC like:
 PART I Title ...
 then `\renewcommand*{\cftpartname}{PART~}`

```

8440 \newcommand*{\cftpartname}{}
8441

```

`\cftpartbreak` Can't use `\newlistentry` for `\l@part` because of the initial penalty and the `\l@part` final `\nobreak` code.

```

8442 \newcommand*{\cftpartbreak}{\addpenalty{-\@highpenalty}%
8443 \addvspace{\cftbeforepartskip}}
8444 \newcommand*{\l@part}[2]{%
8445 \ifnum \c@tocdepth >-2\relax
8446 \cftpartbreak
8447 \begingroup
8448 {\leftskip \cftpartindent\relax
8449 \memRTLleftskip \cftpartindent\relax
8450 %%% \rightskip \@tocrmarg
8451 \memRTLrightskip \@tocrmarg
8452 %%% \parfillskip -\rightskip
8453 \parfillskip -\memRTLrightskip
8454 \parindent \cftpartindent\relax\@afterindenttrue
8455 \interlinepenalty\@M
8456 \leavevmode
8457 \settowidth{\@tempdima}{\cftpartfont\cftpartname}%
8458 \addtolength{\@tempdima}{\cftpartnumwidth}%
8459 \let\@cftbsnum \cftpartpresnum
8460 \let\@cftasnum \cftpartaftersnum
8461 \let\@cftasnumb \cftpartaftersnumb
8462 %%% \advance\leftskip \@tempdima \null\nobreak\hskip -\leftskip
8463 \advance\memRTLleftskip \@tempdima \null\nobreak\hskip -\memRTLleftskip
8464 {\cftpartfont #1}%
8465 \cftpartfillnum{#2}}
8466 \nobreak
8467 \global\@nobreaktrue
8468 \everypar{\global\@nobreakfalse\everypar{}}%
8469 \endgroup
8470 \fi}
8471

```

`\toclevel@part` Needed if the `hyperref` package is used.

```

8472 \newcommand*{\toclevel@part}{-1}
8473

\partnumberline \partnumberline is a special version of \numberline output by \part. Its
\partnumberlinehook default definition is identical to \numberline.

8474 \newcommand*\partnumberlinehook[1]{}
8475 \newcommand{\partnumberline}[1]{%
8476   \partnumberlinehook{#1}%
8477   \hb@xt@{\@tempdima{%
8478     \cftpartname\@cftbsnum #1\@cftasnum\hfil}\@cftasnumb}%%\space}
8479

\cftbeforepartskip These are the user commands to control the typesetting of Part entries. They
\cftpartnumwidth are initialised to give the standard appearance.
\cftpartfont 8480 \newlength{\cftbeforepartskip}
\cftpartpresnum 8481 \setlength{\cftbeforepartskip}{2.25em \@plus\p@}
\cftpartaftersnum 8482 \newlength{\cftpartindent}
\cftpartaftersnumb 8483 \setlength{\cftpartindent}{0em}
\cftpartleader 8484 \newlength{\cftpartnumwidth}
\cftpartdotsep 8485 \setlength{\cftpartnumwidth}{1.5em}
\cftpartpagefont 8486 \newcommand{\cftpartfont}{\large\bfseries}
\cftpartafterpnum 8487 \newcommand{\cftpartpresnum}{}
\cftpartindent 8488 \newcommand{\cftpartaftersnum}{}
\cftpartformatpnum 8489 \newcommand{\cftpartaftersnumb}{}
\cftpartformatpnumhook 8490 \newcommand{\cftpartleader}{%
8491   \large\bfseries\cftdotfill{\cftpartdotsep}}
\cftpartfillnum 8492 \newcommand{\cftpartdotsep}{\cftnodots}
8493 \newcommand{\cftpartpagefont}{\large\bfseries}
8494 \newcommand{\cftpartafterpnum}{}
8495 \newcommand{\cftpartformatpnumhook}[1]{}
8496 \newcommand*{\cftpartformatpnum}[1]{%
8497   \cftpartformatpnumhook{#1}%
8498   \hb@xt@{\@pnumwidth{\hss {\cftpartpagefont #1}}}{
8499     \newcommand{\cftpartfillnum}[1]{%
8500       {\cftpartleader}%
8501       {\cftpartformatpnum{#1}}%
8502       \cftpartafterpnum\par}
8503

\cftchaptername If you want a chapter entry in the ToC like:
Chapter 1. A title ...
then \renewcommand*{\cftchaptername}{\chaptername~}
\renewcommand*{\cftchapteraftersnum}{.}

8504 \newcommand*{\cftchaptername}{}
8505

\l@chapapp Generic chapter/appendix ToC entry typesetting.
8506 \newcommand*\l@chapapp[3]{%
8507   \ifnum \c@tocdepth >\m@ne

```

```

8508 \cftchapterbreak
8509 \vskip \cftbeforechapterskip
8510 {%\leftskip \cftchapterindent\relax
8511 \memRTLleftskip \cftchapterindent\relax
8512 %%\rightskip \@toctrmarg
8513 \memRTLrightskip \@toctrmarg
8514 %%\parfillskip -\rightskip
8515 \parfillskip -\memRTLrightskip
8516 \parindent \cftchapterindent\relax
8517 \@afterindenttrue
8518 \interlinepenalty\@M
8519 \leavevmode
8520 \let\@cftbsnum \cftchapterpresnum
8521 \let\@cftasnum \cftchapteraftersnum
8522 \let\@cftasnumb \cftchapteraftersnumb
8523 \def\@chapapp@head{#3}%
8524 \settowidth{\@tempdima}{\cftchapterfont\@chapapp@head}%
8525 \addtolength{\@tempdima}{\cftchapternumwidth}%
8526 %%\advance\leftskip \@tempdima \null\nobreak\hskip -\leftskip
8527 \advance\memRTLleftskip \@tempdima \null\nobreak\hskip -\memRTLleftskip
8528 {\cftchapterfont #1}\nobreak
8529 \cftchapterfillnum{#2}}
8530 \fi}
8531

```

`\l@chapter` `\l@chapter{<title>}{<page>}` typesets the ToC entry for a chapter heading. It is a parameterised copy of the default `\l@chapter` (see `classes.dtx` for the original definition). Can't use `\newlistentry` for this because of the initial penalty.

```

8532 \newcommand*{\l@chapter}[2]{%
8533 \l@chapapp{#1}{#2}{\cftchaptername}}
8534

```

`\toclevel@chapter` Needed if the `hyperref` package is used.

```

8535 \newcommand*{\toclevel@chapter}{0}
8536

```

`\cftappendixname` If you want a appendix chapter entry in the ToC like:
Appendix A A title ...
then `\renewcommand*{\cftappendixname}{\appendixname~}`

```

8537 \newcommand*{\cftappendixname}{}
8538

```

`\l@appendix` Similar to `\l@chapter` but for an appendix.

```

\toclevel@appendix 8539 \newcommand*{\l@appendix}[2]{%
8540 \l@chapapp{#1}{#2}{\cftappendixname}}
8541 \newcommand{\toclevel@appendix}{0}
8542

```

`\chapternumberline` `\chapternumberline` is a special version of `\numberline` output by `\chapter`.
`\chapternumberlinehook` It's default definition is identical to `\numberline`.

```
8543 \newcommand*\chapternumberlinehook[1]{}
8544 \newcommand{\chapternumberline}[1]{%
8545   \chapternumberlinehook{#1}%
8546   \hb@xt@{\@tempdima{\@chapapp@head\@cftbsnum #1\@cftasnum\hfil}}%
8547   \@cftasnum}
8548
```

`\cftbeforechapterskip` These are the user commands to control the typesetting of Chapter entries.

`\cftchapterindent` They are initialised to give the standard appearance.

```
\cftchapternumwidth 8549 \newlength{\cftbeforechapterskip}
\cftchapterfont      8550 \setlength{\cftbeforechapterskip}{1.0em \@plus\p@}
\cftchapterpresnum   8551 \newlength{\cftchapterindent}
\cftchapteraftersnum 8552 \setlength{\cftchapterindent}{0em}
\cftchapteraftersnumb 8553 \newlength{\cftchapternumwidth}
\cftchapterleader     8554 \setlength{\cftchapternumwidth}{1.5em}
\cftchapterdotsep     8555 \newcommand{\cftchapterfont}{\bfseries}
\cftchapterpresnum     8556 \newcommand{\cftchapterpresnum}{}
\cftchapterpagefont   8557 \newcommand{\cftchapteraftersnum}{}
\cftchapterafterpnum   8558 \newcommand{\cftchapteraftersnumb}{}
\cftchapterformatpnum 8559 \newcommand{\cftchapterleader}{%
\cftchapterformatpnumhook 8560   \bfseries\cftdotfill{\cftchapterdotsep}}
\cftchapterfillnum    8561 \newcommand{\cftchapterdotsep}{\cftnodots}
                        8562 \newcommand{\cftchapterpagefont}{\bfseries}
                        8563 \newcommand{\cftchapterafterpnum}{}
                        8564 \newcommand{\cftchapterformatpnumhook}[1]{}
                        8565 \newcommand*\cftchapterformatpnum[1]{%
                        8566   \cftchapterformatpnumhook{#1}%
                        8567   \hb@xt@{\@pnumwidth{\hfil\cftchapterpagefont #1}}
                        8568   \newcommand*\cftchapterfillnum[1]{%
                        8569     {\cftchapterleader}\nobreak
                        8570     \cftchapterformatpnum{#1}%
                        8571     \cftchapterafterpnum\par}
```

`\cftchapterbreak` Another parameter for `\l@chapter`.

```
8572 \newcommand{\cftchapterbreak}{\addpenalty{-\@highpenalty}}
8573
```

`\l@section` `\l@section{<title>}{<page>}` typesets the ToC entry for a `\section` heading.

`\l@subsection` Similarly `\l@subsection` and `\l@subsubsection` for `\subsection` and

`\l@subsubsection` `\subsubsection` entries, and so on.

```
\l@paragraph 8574 \newlistentry[chapter]{section}{toc}{0}
\l@subparagraph 8575 \cftsetindents{section}{1.5em}{2.3em}
                        8576 \newlistentry[section]{subsection}{toc}{1}
                        8577 \cftsetindents{subsection}{3.8em}{3.2em}
                        8578 \newlistentry[subsection]{subsubsection}{toc}{2}
                        8579 \cftsetindents{subsubsection}{7.0em}{4.1em}
                        8580 \newlistentry[subsubsection]{paragraph}{toc}{3}
```



```

8581 \cftsetindents{paragraph}{10.0em}{5.0em}
8582 \newlistentry[paragraph]{subparagraph}{toc}{4}
8583 \cftsetindents{subparagraph}{12.0em}{6.0em}
8584

```

The typesetting for Figures and Tables is defined later.

22.4 Support for the subfigure package

The code for supporting the subfigure package is, in all essentials, the same as that for the figure and table captions; only the names are changed. However, the code need only be executed if the subfigure package is actually loaded.

```

@cftl@subfigtab This command redefines the \l@subfigure and \l@subtable commands.
8585 \newcommand*{\@cftl@subfigtab}{

\l@subfigure \l@subfigure{\(title)\}(page)} typesets the LoF entry for a subfigure caption
\l@subtable heading, and \l@subtable does the same for subtables.

8586 \newlistentry[figure]{subfigure}{lof}{1}
8587 \cftsetindents{subfigure}{2.3em}{2.5em}
8588 \newlistentry[table]{subtable}{lot}{1}
8589 \cftsetindents{subtable}{2.3em}{2.5em}}
8590 \renewcommand*{\@cftl@subfigtab}{}
8591

```

Call the subfigure package setup code only if the subfigure package has been used.

```

8592 \AtBeginDocument{\ifpackageloaded{subfigure}{\@cftl@subfigtab}}{}
8593

```

22.5 Switching page numbering

It can, at times, be useful to be able to have ToC entries that have no printed page numbers.

```

\cftpagenumbersoff The user level command for switching off page numbers is
\cftpagenumbersoff{<entry>} where <entry> is the name of the entry. The
macro redefines the \cftXfillnum command so that there is no leader and the
page number is ignored.

```

```

8594 \DeclareRobustCommand{\cftpagenumbersoff}[1]{%
8595 \namedef{cft#1fillnum}##1{%
8596 \cftparfillskip\@nameuse{cft#1afterpnum}\par}}
8597

```

```

\cftpagenumberon \cftpagenumberon{<entry>} is the user level command for reversing the
corresponding \cftpagenumbersoff. The macro defines the \cftXfillnum
command to correspond to the default definition.

```

```

8598 \DeclareRobustCommand{\cftpagenumberon}[1]{%
8599 \namedef{cft#1fillnum}##1{%

```

```

8600 \cft#1leader}\nobreak
8601 \cft#1formatpnum}{##1}%
8602 \cft#1afterpnum}\par}}
8603

```

22.6 Chapter precis

`\chapterprecis` The command `\chapterprecis{<text>}` typesets *<text>* at the point where it is called, and also adds *<text>* to the .toc file. It expects to be called immediately after a `\chapter` command.

```

8604 \newcommand{\chapterprecis}[1]{%
8605   \chapterprecishere{#1}
8606   \chapterprecistoc{#1}}

```

`\chapterprecishere` `\chapterprecishere{<text>}` typesets *<text>*. It expects to be called immediately after a `\chapter` command.

```

8607 \newcommand{\chapterprecishere}[1]{%
8608   \prechapterprecis #1\postchapterprecis}

```

`\prechapterprecis` The `\pre...` and `\post...` macros put code before and after
`\prechapterprecisshift` `\chapterprecishere` text. By default `\prechapterprecis` adds some
`\precisfont` (negative) space (defined by `\prechapterprecisshift` whose value depends on
`\postchapterprecis` whether or not the article option is used, as discovered by Lars Madsen) and
starts a quote environment using the `\precisfont`. `\postchapterprecis` ends
the quote environment.

```

8609 \newdimen\prechapterprecisshift
8610 \ifartopt
8611   \prechapterprecisshift=0pt
8612 \else
8613   \prechapterprecisshift=-2\baselineskip
8614 \fi
8615 \newcommand*{\precisfont}{\normalfont\itshape}
8616 \newcommand{\prechapterprecis}{%
8617   \vspace*{\prechapterprecisshift}%
8618   \begin{quote}\precisfont}
8619 \newcommand*{\postchapterprecis}{\end{quote}}
8620

```

`\precistocfont` Font for typesetting chapter precis in the ToC.

```

8621 \newcommand{\precistocfont}{\normalfont\itshape}

```

`\chapterprecistoc` `\chapterprecistoc{<text>}` effectively adds *<text>* to the .toc file. The *<text>*
`\precistocfont` will be typeset within the same margins as the title text of a `\chapter`
heading, using the `\precistocfont` font.

```

8622 \newcommand{\chapterprecistoc}[1]{%
8623   \addtocontents{toc}{\precistocfont{#1}}}
8624 \DeclareRobustCommand{\precistocfont}[1]{%

```

Start a group to localize changes to the paragraphing. Set the left margin to the chapter indent plus the chapter number width.

```
8625 {%\nopagebreak\leftskip \cftchapterindent\relax
8626 \nopagebreak\memRTLleftskip \cftchapterindent\relax
8627 %%% \advance\leftskip \cftchapternumwidth\relax
8628 \advance\memRTLleftskip \cftchapternumwidth\relax
```

Set the right hand margin to \@tocrmarg.

```
8629 %%% \rightskip \@tocrmarg\relax
8630 \memRTLrightskip \@tocrmarg\relax
```

Typeset *<text>* using an italic font, then ensure that the paragraph is finished (to use the local skips). Finally close the group and we are done.

```
8631 \precistocfont #1\par}}
```

22.7 Adding things to the ToC

`\cftlocalchange` `\cftmakelocalchange{<file>}{<pnumwidth>}{<tocrmarg>}` makes an entry into `<file>` to change the `\@pnumwidth` and the `\@tocrmarg` values.

```
8632 \newcommand{\cftlocalchange}[3]{%
8633 \addtocontents{#1}{\protect\setpnumwidth{#2} \protect\setrmarg{#3}}}
```

`\cftaddtitleline` `\cftaddtitleline{<file>}{<kind>}{<title>}{<page>}` adds a `\contentsline` entry to `<file>` with the given information.

```
8634 \newcommand{\cftaddtitleline}[4]{%
8635 \addtocontents{#1}{\protect\contentsline{#2}{#3}{#4}}}
```

`\cftaddnumtitleline` `\cftaddnumtitleline{<file>}{<kind>}{<num>}{<title>}{<page>}` adds a `\contentsline` entry to `<file>` with the given information.

```
8636 \newcommand{\cftaddnumtitleline}[5]{%
8637 \addtocontents{#1}%
8638 {\protect\contentsline{#2}{\protect\numberline{#3}%
8639 {\protect\ignorespaces #4}}{#5}}}
```

`\cftinsertcode` This is a generalisation of a suggestion by Lars Madsen (private email, 2007/12/14).

`\cftinserthook` `\cftinsert` `\cftinsertcode{<id>}{<code>}` creates a hook to be executed in a ‘List of’. Since we use `\@nameuse` it does not matter if the hook does not exist `\cftinserthook{<list>}{<id>}` inserts the hook `<id>` into the `<list>` ‘List of’ file (in the form of `\cftinsert{<id>}`).

Use like this:

```
\cftinsertcode{A}{%
\renewcommand*{\cftchapterfont}{\normalfont\scshape}
...
}
\cftinsertcode{F}{...}
\cftinsertcode{G}{...}
...
```

```

\frontmatter
\tableofcontents
\cftinserthook{lof}{G}
\listoffigures
\chapter{...}
...
\mainmatter
\cftinserthook{lof}{F}
\cftinserthook{toc}{A}
\chapter{...}
...

8640 \newcommand*\cftinsert}[1]{\@nameuse{cftinsert#1}}
8641 \newcommand\cftinsertcode}[2]{\@namedef{cftinsert#1}{#2}}
8642 \newcommand*\cftinserthook}[2]{%
8643   \addtocontents{#1}{\protect\cftinsert\protect{#2\protect}}}
8644

```

22.8 ToC and divisional numbering

Commands are provided, based on the `tocvsec2` package, for changing the section numbering level and the ToC entry level.

`\@setclcnt` Helper macro to set a sectioning-related counter. Use as
`\@setclcnt{<sec>}{<counter>}` to set *counter* to the level of *<sec>*.

```

8645 \newcommand*\@setclcnt}[2]{%
8646   \def\@setclcntok{0}% = false
8647   \nametest{#1}{none}%
8648   \ifsamename
8649     \setcounter{#2}{-10}%
8650     \def\@setclcntok{1}% = true
8651   \fi
8652   \nametest{#1}{book}%
8653   \ifsamename
8654     \setcounter{#2}{-2}%
8655     \def\@setclcntok{1}%
8656   \fi
8657   \nametest{#1}{part}%
8658   \ifsamename
8659     \setcounter{#2}{-1}%
8660     \def\@setclcntok{1}%
8661   \fi
8662   \nametest{#1}{chapter}%
8663   \ifsamename
8664     \setcounter{#2}{0}%
8665     \def\@setclcntok{1}%
8666   \fi
8667   \nametest{#1}{section}%
8668   \ifsamename

```

```

8669     \setcounter{#2}{1}%
8670     \def\@setclcntok{1}%
8671 \fi
8672 \nametest{#1}{subsection}%
8673 \ifsamename
8674     \setcounter{#2}{2}%
8675     \def\@setclcntok{1}%
8676 \fi
8677 \nametest{#1}{subsubsection}%
8678 \ifsamename
8679     \setcounter{#2}{3}%
8680     \def\@setclcntok{1}%
8681 \fi
8682 \nametest{#1}{paragraph}%
8683 \ifsamename
8684     \setcounter{#2}{4}%
8685     \def\@setclcntok{1}%
8686 \fi
8687 \nametest{#1}{subparagraph}%
8688 \ifsamename
8689     \setcounter{#2}{5}%
8690     \def\@setclcntok{1}%
8691 \fi
8692 \nametest{#1}{all}%
8693 \ifsamename
8694     \setcounter{#2}{50}%
8695     \def\@setclcntok{1}%
8696 \fi
8697 %% \if@tempswa\else
8698 \ifnum \@setclcntok = 0\relax
8699 \memerror{%
8700     Unknown document division name (#1)
8701 }{%
8702     I'll ignore it.
8703     Type \space <return> and I'll continue.\MessageBreak
8704     If you haven't mistyped the name then use
8705     \protect\setcounter\space instead.}%
8706 \fi}
8707

```

`\settocdepth` `\settocdepth{<sec>}` is the user command for setting *tocdepth* in the .toc file to the value corresponding to `<sec>`. .

```

8708 \newcommand*\settocdepth[1]{%
8709 \def\@chtodok{0}% false
8710 \nametest{#1}{none}%
8711 \ifsamename
8712 \addtocontents{toc}{\changetocdepth{-10}}%
8713 \def\@chtodok{1}% true
8714 \fi
8715 \nametest{#1}{book}%

```

```

8716 \ifsamename
8717   \addtocontents{toc}{\changetocdepth{-2}}%
8718   \def\@chtodok{1}%
8719 \fi
8720 \nametest{#1}{part}%
8721 \ifsamename
8722   \addtocontents{toc}{\changetocdepth{-1}}%
8723   \def\@chtocdok{1}%
8724 \fi
8725 \nametest{#1}{chapter}%
8726 \ifsamename
8727   \addtocontents{toc}{\changetocdepth{0}}%
8728   \def\@chtocdok{1}%
8729 \fi
8730 \nametest{#1}{section}%
8731 \ifsamename
8732   \addtocontents{toc}{\changetocdepth{1}}%
8733   \def\@chtocdok{1}%
8734 \fi
8735 \nametest{#1}{subsection}%
8736 \ifsamename
8737   \addtocontents{toc}{\changetocdepth{2}}%
8738   \def\@chtocdok{1}%
8739 \fi
8740 \nametest{#1}{subsubsection}%
8741 \ifsamename
8742   \addtocontents{toc}{\changetocdepth{3}}%
8743   \def\@chtocdok{1}%
8744 \fi
8745 \nametest{#1}{paragraph}%
8746 \ifsamename
8747   \addtocontents{toc}{\changetocdepth{4}}%
8748   \def\@chtocdok{1}%
8749 \fi
8750 \nametest{#1}{subparagraph}%
8751 \ifsamename
8752   \addtocontents{toc}{\changetocdepth{5}}%
8753   \def\@chtocdok{1}%
8754 \fi
8755 \nametest{#1}{all}%
8756 \ifsamename
8757   \addtocontents{toc}{\changetocdepth{50}}%
8758   \def\@chtocdok{1}%
8759 \fi
8760 % \if@tempwa
8761 \ifnum\@chtocdok=1\relax

```

The next bit is from Heiko Oberdiek (CTT *Re: Memoir*, `\settocdepth` and `pdflatex => problem with PDF bookmarks`, 2006/07/21) as `hyperref` needs `\tocdepth` set in the body not just in the ToC.

```

8762 \ifundefined{toclevel@#1}{%
8763 \@memwarn{Unknown toplevel for #1}%
8764 }{%
8765 \setcounter{tocdepth}{\@nameuse{toclevel@#1}}%
8766 }
8767 \else
8768 \@memerror{%
8769 Unknown document division name (#1)
8770 }{%
8771 I'll ignore it.
8772 Type \space <return> and I'll continue.}%
8773 \fi}
8774

```

`\toclevel@none` Couple of extras for hypperef to cater for all and none as ‘division levels’.

```

\toclevel@all 8775 \newcommand*\toclevel@none{-10}
8776 \newcommand*\toclevel@all{50}
8777

```

`\changetocdepth` Changes the `tocdepth` counter. Make it robust as it will be written to the *.toc file.

```

8778 \DeclareRobustCommand{\changetocdepth}[1]{\setcounter{tocdepth}{#1}}
8779

```

`\maxtocdepth` `\maxtocdepth{<sec>}` can be used to initialise `tocdepth` to the value corresponding to `<sec>`. This can only be used between the end of the preamble and the `\tableofcontents` command.

```

8780 \newcommand{\maxtocdepth}[1]{%
8781 \setclcnt{#1}{tocdepth}}

```

`\maxsecnumdepth` `\maxsecnumdepth{<sec>}` can be used to initialise `secnumdepth` to the value corresponding to `<sec>`.

```

8782 \newcounter{maxsecnumdepth}
8783 \newcommand{\maxsecnumdepth}[1]{%
8784 \setclcnt{#1}{secnumdepth}\setclcnt{#1}{maxsecnumdepth}}
8785

```

`\setsecnumdepth` `\setsecnumdepth{<sec>}` is the user command for setting `secnumdepth` to the value for `<sec>`. In the preamble it sets both the `secnumdepth` and `maxsecnumdepth` to `<sec>` while in the body it only sets `secnumdepth`.

```

8786 \newcommand{\setsecnumdepth}[1]{%
8787 \ifx\@nodocument\relax% after the preamble
8788 \setclcnt{#1}{secnumdepth}%
8789 \else
8790 \setclcnt{#1}{secnumdepth}%
8791 \setclcnt{#1}{maxsecnumdepth}%
8792 \fi}
8793 \setsecnumdepth{section}
8794

```

23 Bibliography

`\bibindent` The open bibliography uses an indentation of `\bibindent`.

```
8795 \newdimen\bibindent
8796 \setlength\bibindent{1.5em}
```

`\bibitemsep` The vertical separation between items in the bibliography list.

```
8797 \newlength{\bibitemsep}
8798 \setlength{\bibitemsep}{\itemsep}
```

`\biblistextra` A hook into the `bibitemlist`.

```
8799 \newcommand{\biblistextra}{\itemsep=\bibitemsep}
8800
```

`bibitemlist` The `thebibliography` environment starts a new document division. Internally it tweaks some typesetting aspects; principally it uses `\sloppy` because good linebreaking is hard in a bibliography, and `\sfcode‘\.=1000\relax` causes a full stop not to produce an end-of-sentence space. The implementation of the environment is based on the generic `list` environment, and uses the `\c@enumiv` count for the labels. The following code is extracted from the `book` class, plus some additions.

```
8801 \newenvironment{bibitemlist}[1]{%
8802   \typeout{bibitemlist}
8803   \list{\@biblabel{\@arabic\c@enumiv}}%
8804     {\settowidth\labelwidth{\@biblabel{#1}}%
8805      \leftmargin\labelwidth
8806      \advance\leftmargin\labelsep
8807      \@openbib@code
8808      \usecounter{enumiv}%
8809      \let\p@enumiv\@empty
8810      \renewcommand\theenumiv{\@arabic\c@enumiv}%
8811      \biblistextra}%
8812   \sloppy
8813   \clubpenalty4000
8814   \@clubpenalty \clubpenalty
8815   \widowpenalty4000%
8816   \sfcode‘\.\@m}%
8817   {\def\@noitemerr
8818    {\@latex@warning{Empty ‘thebibliography’ environment}}}%
8819   \endlist}
8820
```

`\newblock` The default is for `\newblock` to provide a small space.

```
8821 \newcommand{\newblock}{\hskip .11em\@plus.33em\@minus.07em}
```

`\@openbib@code` This is an empty hook. It will be modified if the `openbib` option is used.

```
8822 \let\@openbib@code\@empty
```


`\setbiblabel` This is the user command for setting the label for a `\bibitem`. The following `\@biblabel` sets the default definition.

```
8823 \newcommand*\setbiblabel}[1]{%
8824   \renewcommand*\@biblabel}[1]{#1}}
8825 \setbiblabel{[#1]\hfill}
```

`\@memb@bchap` In a CTT thread *memoir*, *natbib*, and *chapterbib* in January 2003 there was a discussion on how to get all three to work together. Donald Arseneau suggested that `\bibsection` be used as a ‘standard’ way of controlling the sectioning command of a bibliography. Here is an implementation of that idea. `\@memb@bchap` is a copy of *memoir*’s original code for the start of the `thebibliography` environment which used a `\chapter*` title. `\@memb@bsec` is the much simpler version for a `\section` title.

```
8826 \newcommand{\@memb@bchap}{%
8827   \chapter*\bibname}%
8828   \bibmark
8829   \ifnobibintoc\else
8830     \phantomsection
8831     \addcontentsline{toc}{chapter}{\bibname}%
8832   \fi
8833   \prebibhook}
8834 \newcommand{\@memb@bsec}{\section{\bibname}\prebibhook}
```

`\bibsection` Normally treat the bibliography heading as a chapter.

```
8835 \newcommand{\bibsection}{\@memb@bchap}
8836
```

`thebibliography` The definition of the `thebibliography` environment in this class is not quite the same as in the standard classes!

```
8837 \newenvironment{thebibliography}[1]{%
8838   \bibsection
8839   \begin{bibitemlist}{#1}}{\end{bibitemlist}\postbibhook}
```

`\ifnobibintoc` Flag to control whether or not to add the bibliography title to the ToC, and `\bibintoc` declarations to set the flag. Default is to put the title into the ToC.

```
\nobibintoc 8840 \newif\ifnobibintoc
8841 \newcommand*\bibintoc{\nobibintocfalse}
8842 \newcommand*\nobibintoc{\nobibintoctrue}
8843 \bibintoc
8844
```

`\prebibhook` These two macros are called just before starting the bib items and just after `\postbibhook` finishing them. By default they do nothing but can be changed by the user to give, say, some introductory information.

```
8845 \newcommand{\prebibhook}{}
8846 \newcommand{\postbibhook}{}
8847
```

`\@cite` The output of the `\cite` command is produced by this macro. The default is used. The `cite` package is a good way of changing this.

```
8848 % \renewcommand*{\@cite}[1]{[#1]}
```

23.1 Use with the `natbib` and `chapterbib` packages

The `natbib` package [Dal99] uses its own definition for the `thebibliography` environment, which knows nothing about adding the Bibliography to the ToC. The following makes appropriate changes to `natbib` code to support the class. The `chapterbib` package by Donald Arseneau also may make changes to the `thebibliography` environment — perhaps memoir’s version or `natbib`’s version. As packages get loaded after classes, I have to try and patch any non-memoir modifications at begin document time. The fixes have to be firstly for `natbib` and only after those can `chapterbib` be considered.

`\bibsection` Natbib provides `\bibsection` for titling the bibliography. I also have to extend `\endthebibliography` the end of the `thebibliography` environment to cater for `\postbibhook`.

```
8849 \AtBeginDocument{%
```

```
8850   \@ifpackageloaded{natbib}{% natbib is loaded
```

As `natbib` is used, change `\endthebibliography` to the class definition.

```
8851     \addtodef{\endthebibliography}{}{\vskip-\lastskip\postbibhook}
```

```
8852     \@ifpackagewith{natbib}{sectionbib}{% with sectionbib option
```

`natbib`’s `sectionbib` option is used,

```
8853       \renewcommand{\bibsection}{\@memb@bsec}}%
```

The `sectionbib` option is not used, so we have a chapter title.

```
8854       {\renewcommand{\bibsection}{\@memb@bchap}}}%
```

Finished with any `natbib` related changes.

```
8855   }
```

Now deal with `chapterbib` if necessary.

```
8856   \@ifpackagewith{chapterbib}{sectionbib}{%
```

`chapterbib` is used with its `sectionbib` option. This is the only case to worry about. Kill `chapterbib`’s `\sectionbib` macro which it calls at begin document to do its patch, then make sure the class definition is used.

```
8857     \renewcommand{\sectionbib}[2]{}%
```

```
8858     \renewcommand{\bibsection}{\@memb@bsec}}{}%
```

And we’ve finished with this bunch of `\AtBeginDocument` code.

```
8859 }
```

```
8860
```

24 The index

I allow for a single column index as well as the default double column.

`\ifonecolindex` TRUE for a one column index.

```
8861 \newif\ifonecolindex
8862 \onecolindexfalse
8863
```

`\onecolindex`

```
\twocolindex 8864 \newcommand*\{onecolindex}\{onecolindextrue}
8865 \newcommand*\{twocolindex}\{onecolindexfalse}
8866
```

`theindex` The environment `theindex` can be used for indices. It makes an index with two columns, with each entry a separate paragraph. At the user level the commands `\item`, `\subitem` and `\subsubitem` are used to produce index entries of various levels. When a new letter of the alphabet is encountered an amount of `\indexspace` white space can be added.

```
8867 \newenvironment{theindex}{%
8868   \clearforchapter
8869   \if@twocolumn
8870     \@restonecolfalse
8871   \else
8872     \@restonecoltrue
8873   \fi
8874   \ifonecolindex
8875     \onecolumn
8876     \chapter*\{indexname}
8877     \preindexhook
8878   \else
8879     \setlength{\columnseprule}\{indexrule}%
8880     \setlength{\columnsep}\{indexcolsep}%
8881     \twocolumn[\@makeschapterhead\{indexname}
8882               \preindexhook]%
8883   \fi
8884   \indexmark
8885   \ifnoindexintoc\else
8886     \phantomsection
8887     \addcontentsline{toc}{chapter}\{indexname}%
8888   \fi
8889   \thispagestyle{indextitlepagestyle}\parindent\z@
8890   \parskip\z@ \@plus .3\p@ \relax
8891   \let\item\@idxitem}%
8892   {\if@restonecol\onecolumn\else\twocolumn\fi}
8893
```

`\ps@indextitlepagestyle`

```
8894 \aliaspagestyle{indextitlepagestyle}{chapter}
8895
```

`\ifnoindexintoc` Flag to control whether or not to add the index title to the ToC, and
`\indexintoc` declarations to set the flag. Default is to put the title into the ToC.

```

\ifnoindexintoc 8896 \newif\ifnoindexintoc
                  8897 \newcommand*\indexintoc{\noindexintocfalse}
                  8898 \newcommand*\noindexintoc{\noindexintoctrue}
                  8899 \indexintoc
                  8900

```

`\indexcolsep` These two lengths control the column separation and the thickness of the
`\indexrule` inter-column rule.

```

                  8901 \newlength{\indexcolsep} \setlength{\indexcolsep}{35pt}
                  8902 \newlength{\indexrule} \setlength{\indexrule}{0pt}
                  8903

```

`\preindexhook` A macro that is called between the index heading and the start of the two
columns. The user can modify it to add something.

```

                  8904 \newcommand{\preindexhook}{}

```

`\l@index` Format the index entry in the table of contents.

```

                  8905 \newcommand{\l@index}{\@dottedtocline{1}{0em}{0pt}}

```

`\@idxitem` These macros are used to format the entries in the index.

```

\subitem 8906 \newcommand{\@idxitem} {\par\hangindent 40\p@}
\subsubitem 8907 \newcommand{\subitem} {\par\hangindent 40\p@ \hspace*{20\p@}}
                  8908 \newcommand{\subsubitem}{\par\hangindent 40\p@ \hspace*{30\p@}}

```

`\indexspace` The amount of white space that is inserted between ‘letter blocks’ in the index.

```

                  8909 \newcommand{\indexspace}{\par \vskip 10\p@ \@plus5\p@ \@minus3\p@\relax}
                  8910

```

`\makeindex` This is a modified version of the kernel `\makeindex` to allow for multiple indexes.

`\index` It also defines `\index` (which is a modified version of the standard `\index`) and

`\specialindex` `\specialindex`.

`\makememindexhook` `\makememindexhook` is a null op but can be redefined to add extra code into
`\makeindex`. For example, to incorporate `ledmac`’s `\edindex` into the scheme:

```

\renewcommand*\makememindexhook{%
  \def\edindex{\@bsphack%
    \@ifnextchar [{\l@d@index}{\l@d@index[\jobname]}}

```

```

8911 \newcommand*\makememindexhook{}
8912 \providecommand*\makeindex{}
8913 \renewcommand*\makeindex[1][\jobname]{%
8914   \if@filesw
8915     \def\index{\@bsphack%
8916       \@ifnextchar [{\@index}{\@index[\jobname]}}
8917     \def\specialindex{\@bsphack\@spindex}%
8918     \makememindexhook

```

```

8919 \expandafter\newwrite\csname #1@idxfile\endcsname
8920 \expandafter\immediate\openout \csname #1@idxfile\endcsname #1.idx\relax
8921 \typeout{Writing index file #1.idx }%
8922 \fi}
8923

```

Initially define, but emasculate, `\index` and `\specialindex` which are defined properly by the user calling `\makeindex`.

```

8924 \renewcommand{\index}[2][\jobname]{\@bsphack\@esphack}
8925 \newcommand{\specialindex}[3]{\@bsphack\@esphack}
8926

```

`\printindex` The command to read an ind file.

```

8927 \newcommand{\printindex}[1][\jobname]{\@input@{#1.ind}}
8928

```

`\ifreportnoidxfile` Two booleans to control reporting on unknown idx files and displaying indexed items in the margin.

```

\reportnoidxfile
\ignorenoidxfile 8929 \newif\ifreportnoidxfile
\ifshowindexmark 8930 \newcommand*{\reportnoidxfile}{\reportnoidxfiletrue}
\showindexmarks 8931 \newcommand*{\ignorenoidxfile}{\reportnoidxfilefalse}
\hideindexmarks 8932 \ignorenoidxfile
8933 \newif\ifshowindexmark
8934 \newcommand*{\showindexmarks}{\showindexmarktrue}
8935 \newcommand*{\hideindexmarks}{\showindexmarkfalse}
8936 \hideindexmarks
8937

```

`\@index` `\@index[file]` is first stage of `\index`, handling the idx file.

```

8938 \def\@index[#1]{%
8939 \ifundefined{#1@idxfile}%
8940 {\ifreportnoidxfile
8941 \@memwarn{Undefined index file #1}%
8942 \fi
8943 \begingroup
8944 \@sanitize
8945 \@nowrindex}%
8946 {\def\@idxfile{#1}%
8947 \begingroup
8948 \@sanitize
8949 \@wrindexm@m}}

```

`\@nowrindex` Called when there is no idx file to throw away the indexed item.

```

8950 \newcommand{\@nowrindex}[1]{%
8951 \ifshowindexmark\@showidx{#1}\fi\endgroup\@esphack}
8952

```

`\@wrindexm@m` The next stage in index processing is `\@wrindexm@m{item}`, which writes the idx file name and indexed item to the aux file. The `\@wrindexhyp` macro provides hyperlinks in case the `hyperref` package is used.

```

8953 \newcommand{\@wrindexm@m}[1]{\@wrindexhyp#1||\}
8954 \def\@wrindexhyp#1|#2|#3\{\%
8955   \ifshowindexmark\@showidx{#1}\fi
8956   \ifx\#2\%
8957     \protected@write\@auxout{\%
8958       {\string\@wrindexm@m{\@idxfile}{#1|hyperpage}{\thepage}}\%
8959   \else
8960     \def\Hy@temp@A{#2}\%
8961     \ifx\Hy@temp@A\HyInd@ParenLeft
8962       \protected@write\@auxout{\%
8963         {\string\@wrindexm@m{\@idxfile}{#1|#2hyperpage}{\thepage}}\%
8964     \else
8965       \protected@write\@auxout{\%
8966         {\string\@wrindexm@m{\@idxfile}{#1|#2}{\thepage}}\%
8967     \fi
8968   \fi
8969 \endgroup
8970 \@esphack}

```

`\hyperpage` These are defined in the `hyperref` package but we need them. Other hyperstuff is
`\hyperlink` only used if the package itself is used.

```

8971 \newcommand{\hyperpage}[1]{#1}
8972 \newcommand{\hyperlink}[2]{#2}
8973

```

`\@wrindexm@m` The macro `\@wrindexm@m{file}{item}{page}` has been written into the aux file. It is normally defined so that it calls `\@@wrindexm@m{item}{page}` to finally write to the idx file.

```

8974 \newcommand{\@wrindexm@m}[1]{\begingroup
8975   \def\@idxfile{\@nameuse{#1@idxfile}}
8976   \@sanitize
8977   \@@wrindexm@m}
8978

```

`\@@wrindexm@m` `\@@wrindexm@m{<item>}{<page>}` writes the `\indexentry` to the idx file. This file is read both at the beginning and end of a document so `\@@wrindexm@m` must be disabled for one of these reads otherwise the index entries will be doubled up. Initially I disabled it after the first read. It eventually dawned on me that this meant two LaTeX runs to update the idx. Enabling it for the second read means that the idx file is updated at the end of each run.

```

8979 \newcommand{\@@wrindexm@m}[2]{\endgroup}
8980 \AtBeginDocument{\%
8981   \def\@@wrindexm@m#1#2{\%
8982     \if@filesw
8983       \immediate\write \@idxfile{\string\indexentry{#1}{#2}}\%
8984     \fi
8985     \endgroup}\%
8986 }

```

\@spindex Now do similar things for \@specialindex.

```

8987 \newcommand{\@spindex}[2]{%
8988   \@ifundefined{#1@idxfile}%
8989   {\ifreportnoidxfile
8990     \@memwarn{Undefined index file #1}%
8991     \fi
8992     \begingroup
8993     \@sanitize
8994     \@nowrindex}%
8995   {\def\@idxfile{#1}%
8996     \def\@sptheidx{#2}%
8997     \begingroup
8998     \@sanitize
8999     \@wrspindex}}
9000

```

\@wrspindex

```

\@@wrspindexhyp 9001 \newcommand{\@wrspindex}[1]{\@@wrspindexhyp#1||\}
9002 \def\@@wrspindexhyp#1|#2|#3\{%
9003   \ifshowindexmark\@showidx{#1}\fi
9004   \ifx\#2\%
9005     \protected@write\@auxout{%
9006       {\string\@@wrindexm@{\@idxfile}%
9007        {#1|hyperspindexpage(\thepage)}}%
9008       {\@nameuse{the\@sptheidx}}}%
9009   \else
9010     \def\Hy@temp@A{#2}%
9011     \ifx\Hy@temp@A\HyInd@ParenLeft
9012       \protected@write\@auxout{%
9013         {\string\@@wrindexm@{\@idxfile}%
9014          {#1|#2hyperspindexpage(\thepage)}}%
9015         {\@nameuse{the\@sptheidx}}}%
9016     \else
9017       \protected@write\@auxout{%
9018         {\string\@@wrindexm@{\@idxfile}{#1|#2}%
9019          {\@nameuse{the\@sptheidx}}}%
9020     \fi
9021   \fi
9022   \endgroup
9023   \@esphack}

```

\hyperspindexpage

```

9024 \def\hyperspindexpage(#1)#2{\hyperlink{page.#1}{#2}}
9025

```

\ifmemhyperindex Flag to turn off \hyperindexfalse hyperrefing the index (apparently xindy
\memhyperindextrue can't cope with this. Email from Frederic Connes, 2005/07/13:
\memhyperindexfalse ...You use |hyperspindex(\thepage) which xindy doesn't
recognise as a valid markup-locref. And I don't know 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. ...

The default is `\memhyperindextrue`. Setting `\memhyperindexfalse` prohibits a hyper index — setting whatever option in `hyperref` to disable hyper indexing will (probably) not work as far as xindy is concerned.

```
9026 \newif\ifmemhyperindex
9027   \memhyperindextrue
9028
```

`\ifm@mxindy` Use `\xindyindex` when you are going to use the xindy program rather than `makeindex`. `hyperref`ed entries won't work with xindy. The code was supplied by `\m@mxindyfalse` Frederic Connes²¹ in an email to me on 2006/01/08.

```
\xindyindex 9029 \newif\ifm@mxindy
\@wrspxindexhyp 9030 \m@mxindyfalse
9031 \newcommand*{\xindyindex}{\m@mxindytrue}
9032 \def\@wrspxindexhyp#1|#2|#3\{ \{ %
9033   \ifshowindexmark\@showidx{#1}\fi
9034   \ifx\#2\%
9035     \protected@write\@auxout{}%
9036       {\string\@wrsindexm@{\@idxfile}%
9037       \ifm@mxindy{#1}\else{#1|hyperspxindexpage(\thepage)}\fi
9038       {\@nameuse{the\@spxtheidx}}}%
9039   \else
9040     \def\Hy@temp@A{#2}%
9041     \ifx\Hy@temp@A\HyInd@ParenLeft
9042       \protected@write\@auxout{}%
9043         {\string\@wrsindexm@{\@idxfile}%
9044         \ifm@mxindy{#1|#2}\else{#1|#2hyperspxindexpage(\thepage)}\fi
9045         {\@nameuse{the\@spxtheidx}}}%
9046     \else
9047       \protected@write\@auxout{}%
9048         {\string\@wrsindexm@{\@idxfile}{#1|#2}%
9049         {\@nameuse{the\@spxtheidx}}}%
9050     \fi
9051   \fi
9052 \endgroup
9053 \@esphack}
9054
```

`\@wrsindexhyp` If the `hyperref` package is not being used, or `\ifmemhyperindex` is false, there is
`\@wrspxindexhyp` no need to clutter up the index files.

```
9055 \AtBeginDocument{%
9056   \@ifpackageloaded{hyperref}{\memhyperindexfalse}%
```

²¹frederic@connes.org

If the hyperref package is not being used, or a hyperindex is not required, simplify!

```

9057 \ifmemhyperindex\else
9058 \def\@wrindexhyp#1||\{\%
9059 \ifshowindexmark\@showidx{#1}\fi
9060 \protected@write\@auxout{\%
9061 {\string\@wrindexm\@{\@idxfile}{#1}{\thepage}}\%
9062 \endgroup
9063 \esphack}\%
9064 \def\@wrspindexhyp#1||\{\%
9065 \ifshowindexmark\@showidx{#1}\fi
9066 \protected@write\@auxout{\%
9067 {\string\@wrindexm\@{\@idxfile}{#1}{\@nameuse{the\@sptheid}}}\%
9068 \endgroup
9069 \esphack}\%
9070 \fi
9071 }
9072

```

\see These definitions are taken from the makeidx package.

```

\seename 9073 \newcommand*\see{[2]{\emph{\seename} #1}
\seealso 9074 \newcommand*\seealso{[2]{\emph{\seealso} #1}
\alsosome 9075 \newcommand*\seealso{[2]{\emph{\alsosome} #1}
9076 \newcommand*\alsosome{see also}
9077

```

\citeindexfile For the natbib package, and possibly other packages that do some special indexing.

```

9078 \newcommand{\citeindexfile}{\jobname}
9079 \AtBeginDocument{\@ifpackageloaded{natbib}{\%
9080 \def\NAT@index{\index[\citeindexfile]{\NAT@idxtxt}}}\%
9081

```

The next part of the code is essentially the showidx package. I tried putting index entries into marginpars but too many on a page led to the ‘too many floats’ problem.

```

\indexmarkstyle
\@indexbox 9082 \newtoks\indexmarkstyle
9083 \indexmarkstyle{\normalfont\footnotesize\ttfamily}
9084 \newinsert\@indexbox
9085 \dimen\@indexbox\maxdimen
9086
9087 \begingroup
9088 \catcode'\@active
9089 \expandafter\gdef\csname\string @sanitizeat\endcsname
9090 {\def @{\char'\@}}
9091 \endgroup
9092

```

```

\@showidx
9093 \newcommand{\@showidx}[1]{%
9094   \insert\@indexbox{%
9095     \@sanitizeat
9096     \the\indexmarkstyle
9097     \hsize\marginparwidth
9098     \hangindent\marginparsep \parindent\z@
9099     \everypar{}\let\par\@par \parfillskip\@flushglue
9100     \lineskip\normallineskip
9101     \baselineskip .8\normalbaselineskip\sloppy
9102     \raggedright \leavevmode
9103     \vrule \@height .7\normalbaselineskip \@width \z@\relax
9104     #1\relax
9105     \vrule \@height \z@ \@depth .3\normalbaselineskip \@width \z@\relax
9106   }%
9107   \ifhmode\penalty\@M \hskip\z@skip\fi}
9108

\@leftidx
\@rightidx 9109 \newcommand{\@leftidx}{\hskip-\marginparsep \hskip-\marginparwidth}
9110 \newcommand{\@rightidx}{\hskip\columnwidth \hskip\marginparsep}
9111

\@mkidx
9112 \newcommand{\@mkidx}{\vbox to \z@{%
9113   \rlap{%
9114     \if@twocolumn
9115       \if@firstcolumn \@leftidx \else \@rightidx \fi
9116     \else
9117       \if@twoside
9118         \ifodd\c@page \@rightidx \else \@leftidx \fi
9119       \else
9120         \@rightidx
9121       \fi
9122     \fi
9123     \box\@indexbox
9124   }%
9125   \vss}}
9126

```

25 Page bottom

\raggedbottom These kernel macros need changing because of the new \@indexbox marginal
\flushbottom insert.

```

\@texttop 9127 \renewcommand{\raggedbottom}{%
9128   \def\@textbottom{\vskip\z@ plus.0001fil}%
9129   \let\@texttop\@mkidx}
9130 \renewcommand{\flushbottom}{%

```

```

9131 \let\@textbottom\relax
9132 \let\@texttop\@mkidx}
9133 \let\@texttop\@mkidx
9134

```

25.1 Widows and sloppybottom

There was a discussion *widow handling?* on CTT in May 2006.

ivowel@gmail.com wrote

in experimenting with raggedbottom, widowpenalty, and clubpenalty, I think that I have not found a solution that strikes me as particularly desirable. I think what I would really like is that widows (i.e., left-over single lines that begin on the following page) are resolved not by pushing one extra line from the same paragraph also onto the next page, but by stretching the `\textheight` to allow this one extra at the bottom of the same page.

Donald Arseneau, as he so often does, came up with a solution he termed `\sloppybottom`. Here is a generalised version.

`\sloppybottom` `\sloppybottom` allows an extra line on a page to save a widow. You must increase the `\topskip` (by 60 percent is reasonable) and this will push the text lower on the page. Run `\checkandfixthelayout` after the change. For example:

```

\setlength{\topskip}{1.6\topskip}
\checkandfixthelayout
\sloppybottom
...

```

```

9135 \newcommand*{\sloppybottom}{%
9136 \def\@textbottom{\vskip \z@ \@plus.0001fil \@minus .95\topskip}%
9137 \topskip=1\topskip \@plus 0.625\topskip \@minus .95\topskip
9138 \def\@texttop{\vskip \z@ \@plus -0.625\topskip \@minus -0.95\topskip}}
9139

```

26 Glossaries

Standard LaTeX provides little and insufficient support for glossaries, just a `\makeglossary` and `\glossary` commands. This class does somewhat better. The code follows along the lines of that for indexes.

```

\ifonecolglossary TRUE for a one column glossary, otherwise its two column.
\onecolglossarytrue 9140 \newif\ifonecolglossary
\onecolglossaryfalse 9141 \onecolglossarytrue

```

```

\onecolglossary
\twocolglossary 9142 \newcommand*\onecolglossary{\onecolglossarytrue}
                  9143 \newcommand*\twocolglossary{\onecolglossaryfalse}
                  9144

theglossary      The environment theglossary is used for glossaries. It makes a glossary with
                  one or two columns, headed by a chapter-like title.

                  9145 \newenvironment{theglossary}{%
                  9146   \if@twocolumn
                  9147     \@restonecolfalse
                  9148   \else
                  9149     \@restonecoltrue
                  9150   \fi
                  9151   \ifonecolglossary
                  9152     \onecolumn
                  9153     \chapter*\glossaryname}
                  9154   \preglossaryhook
                  9155   \else
                  9156     \setlength{\columnseprule}{\glossaryrule}
                  9157     \setlength{\columnsep}{\glossarycolsep}
                  9158     \twocolumn[\@makeschapterhead{\glossaryname}
                  9159               \preglossaryhook]%
                  9160   \fi
                  9161   \glossarymark
                  9162   \ifnoglossaryintoc\else
                  9163     \phantomsection
                  9164     \addcontentsline{toc}{chapter}{\glossaryname}
                  9165   \fi
                  9166   \thispagestyle{chapter}\parindent\z@
                  9167   \parskip\z@ \@plus .3\p@\relax
                  9168   \begintheglossaryhook}%
                  9169   {\atendtheglossaryhook\if@restonecol\onecolumn\else\twocolumn\fi}
                  9170

\begintheglossaryhook  Two vacuous macros called as the last thing by \begin{theglossary} and the
\atendtheglossaryhook first thing by \end{theglossary} These could be used, for example, to insert
                      another kind of environment.

                      9171 \newcommand*\begintheglossaryhook{}
                      9172 \newcommand*\atendtheglossaryhook{}

\preglossaryhook      A vacuous macro called after the title is set and before the listing starts. The
                      user can modify it to, for example, add some explanatory text.

                      9173 \newcommand*\preglossaryhook{}
                      9174

\ifnoglossaryintoc    Flag and declarations to control whether or not the glossary title is added to the
\glossaryintoc        ToC. Default is to put the title in the ToC.
\noglossaryintoc 9175 \newif\ifnoglossaryintoc
                  9176 \newcommand*\glossaryintoc{\noglossaryintocfalse}

```

```

9177 \newcommand*\noglossaryintoc{\noglossaryintoctrue}
9178 \glossaryintoc
9179

```

\glossarycolsep When the glossary is two column these lengths control the column separation
\glossaryrule and the width of a rule between the columns.

```

9180 \newdimen\glossarycolsep \glossarycolsep=35\p@
9181 \newdimen\glossaryrule \glossaryrule=0\p@

```

\glossaryspace Vertical space between ‘letter blocks’ in the glossary. Note that this is a macro, not a length.

```

9182 \newcommand*\glossaryspace{%
9183     \par \vskip 1.0\onelineskip \@plus 5\p@ \@minus3\p@\relax}
9184

```

\makeglossary The preamble command to set up a glossary. Output is to the #1.glo file. We need the \providecommand in case \nofiles has been called before the class. \makeglossary sets up several file-specific macros.

```

9185 \providecommand*\makeglossary{}
9186 \renewcommand*\makeglossary[1][\jobname]{%
9187     \makememglossaryhook
9188     \@namedef{memglsact#1}{@}%          actual
9189     \@namedef{memglsnx#1}{}%           no ref
9190     \@namedef{memglsn#1}{\thepage}%    num by page
9191     \@namedef{memglsnf#1}{\memjustarg}% no special number format
9192     \if@filesw \expandafter\newwrite\csname #1memglofile\endcsname
9193         \expandafter\immediate\openout \csname #1memglofile\endcsname #1.glo\relax
9194     \typeout{Writing glossary file #1.glo}%
9195     \fi}
9196

```

\makememglossaryhook A vacuous macro called at the start of the \makeglossary code. It is redefinable to do something useful.

```

9197 \newcommand*\makememglossaryhook{}
9198

```

\glossary The user command for a raw glossary item. The full calling sequence looks like:
\glossary[*<file>*](*<key>*){*<term>*}{*<desc>*}
 but most is accomplished by lower level macros. It calls \@glossary to handle the first optional argument.

```

9199 \def\glossary{\@bsphack%
9200     \@ifnextchar [{\@glossary}{\@glossary[\jobname]}}%

```

\@glossary \@glossary[*<file>*] handles the first of the \glossary optional arguments, even though the calling sequence looks like:

```
\@glossary[<file>](<key>){<term>}{<desc>}.
```

It calls @@glossary to handle the second optional argument and the rest.

```

9201 \def\@glossary[#1]{%
9202     \@ifnextchar ({\@@glossary[#1]}{\@@glossary[#1]()}}

```

`\@glossary` The base macro for `\glossary`. Its real calling sequence is:
`\@glossary[⟨file⟩](⟨key⟩){⟨term⟩}{⟨desc⟩}`.
 Provided the glossary output file has been set up it calls `\@wrglom@m` to write the data, otherwise it throws away its arguments.

```

9203 \def\@glossary[#1](#2)#3#4{%
9204   \ifundefined{#1memglofile}{%
9205     \begingroup
9206     \@sanitize
9207     \endgroup
9208     \@esphack%
9209   }{%
9210     \def\memglofile{#1}%
9211     \begingroup
9212     \@sanitize

```

Use the correct key, `⟨key⟩` if given, otherwise `⟨term⟩`.

```

9213   \ifx\@empty#2\@empty
9214     \@wrglom@m{#3}{#3}{#4}%
9215   \else
9216     \@wrglom@m{#2}{#3}{#4}%
9217   \fi}}
9218

```

`\@wrglom@m` `\@wrglom@m{⟨key⟩}{⟨term⟩}{⟨desc⟩}` writes its three arguments, plus the `⟨ref⟩` and the `⟨num⟩` to the aux file as the arguments to the `\@wrglom@m` macro.

```

9219 \newcommand{\@wrglom@m}[3]{%
9220   \protected@write\@auxout{%
9221     {\string\@wrglom@m{\memglofile}{#1}{#2}{#3}{\@nameuse{memglsnx\memglofile}}{\@nam
9222   \endgroup
9223   \@esphack}
9224

```

`\@wrglom@m` The calling sequence looks like:
`\@wrglom@m{⟨file⟩}{⟨key⟩}{⟨term⟩}{⟨desc⟩}{⟨ref⟩}{⟨num⟩}`
 but is actually really only
`\@wrglom@m{⟨file⟩}`
 It saves the output file identifier in `\memglofile` and `⟨file⟩` as `\m@mgf`, then calls `\memwritetoglo` to handle the remaining ‘arguments’.

```

9225 \newcommand{\@wrglom@m}[1]{\begingroup
9226   \def\memglofile{\@nameuse{#1memglofile}}%
9227   \def\m@mgf{#1}%
9228   \@sanitize
9229   \memwritetoglo}
9230

```

`\memwritetoglo` `\memwritetoglo{⟨key⟩}{⟨name⟩}{⟨desc⟩}{⟨ref⟩}{⟨num⟩}` will, at an appropriate time write the raw glossary data to the appropriate `glo` file.
`\@ctualm@m``writetoglo` The aux file is read twice, once as one of the initial actions at the start of the document environment, and again at the end of the document environment. The

raw glossary data must only be written once from the `aux` to the `glo` file.

Initially `\memwritetoglo` is defined to do nothing, then, after the `aux` file has been read at the beginning is redefined to do its writing.

The Makeindex program expects its input to be like:

`\glosaryentry{key@data}{num}` Very cunningly the raw data is fed to the `glo` file looking like:

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

which as far as Makeindex is concerned is simply:

`\glosaryentry{key@{data}}{num}`

(note where all the braces are).

```

9231 \newcommand{\memwritetoglo}[5]{\endgroup}
9232 \newcommand{\actualm@m writetoglo}[5]{%
9233   \immediate\write \memglofile{\string\glossaryentry{#1\@nameuse{memglact}\m@gf}
9234     {\string\memgloterm{#2}}{\string\memglodesc{#3}}
9235     {\string\memgloref{#4}}\@nameuse{memgl snf}\m@gf}}{#5}}%
9236   \endgroup}
9237 \AtBeginDocument{%
9238   \let\memwritetoglo\actualm@m writetoglo}
9239
```

`\changeglossactual` There are internal macros that are file-dependant, their names being constructed from the file name plus a fixed name. The `\makeglossary` command initializes these but the user may well need to change them. In the macros below, `\changeglossref` `\changeglossnum` `\changeglossnumformat` `...actual` refers to the actual character in a Makeindex configuration file (default @), `...ref` to the `(ref)` glossary argument, `...num` to the metanum argument (default `\thepage`) and `...numformat` to the Makeindex encapsulating command. Example uses are:

```

\changeglossactual[file]{?}
\changeglossref[file]{\thepage}
\changeglossnum[file]{\thesection}
\changeglossnumref[file]{|textbf}
```

```

9240 \newcommand*{\changeglossactual}[2][\jobname]{%
9241   \@namedef{memglact#1}{#2}}
9242 \newcommand*{\changeglossref}[2][\jobname]{%
9243   \@namedef{memgl snx#1}{#2}}
9244 \newcommand*{\changeglossnum}[2][\jobname]{%
9245   \@namedef{memgl sn#1}{#2}}
9246 \newcommand*{\changeglossnumformat}[2][\jobname]{%
9247   \@namedef{memgl snf#1}{#2}}
9248
```

`\glossitem` The final sorted glossary data is in the form:

`\glossitem{<term>}{<desc>}{<ref>}{<num>}`

The definition here does little.

```

9249 \newcommand{\glossitem}[4]{#1 #2 #3 #4\par}
9250
```

`\memgloterm` These are wrappers round the $\langle term \rangle$, $\langle desc \rangle$, $\langle ref \rangle$ and $\langle num \rangle$ elements in the
`\memglodesc` final data. They are vacuous to start with.

```

\memgloref 9251 \newcommand*{\memgloterm}[1]{#1}
\memglonum 9252 \newcommand*{\memglodesc}[1]{#1}
          9253 \newcommand*{\memgloref}[1]{#1}
          9254 \newcommand*{\memglonum}[1]{#1}
          9255
\printglossary \printglossary[\langle file \rangle] inputs the glossary gst file.
          9256 \newcommand*{\printglossary}[1][\jobname]{\@input{#1.gls}}
          9257

```

27 Notes/Marginalia

27.1 A simple interface for specifying locations

The memoir class as of December 2009 supports `\marginpar`, `\sidepar`, and `\sidebar` for placing information in a margin. The means of specifying in which margin the information should be put is different for each of these. Here a consistent location interface is provided for all marginalia.

`\m@msetm@argin` A workhorse macro for setting a margin placement code. The argument may be
`\m@mm@argin` one of: left, right, outer, or inner, with `\m@mm@argin` defined as the corresponding
codes of 0, 1, 2, or 3. An unknown argument is coded as a negative number.

```

9258 \newcommand*{\m@msetm@argin}[1]{%
9259   \def\@tempa{#1}\def\@tempb{left}%
9260   \ifx\@tempa\@tempb
9261     \def\m@mm@argin{0}%
9262   \else
9263     \def\@tempb{right}%
9264     \ifx\@tempa\@tempb
9265       \def\m@mm@argin{1}%
9266     \else
9267       \def\@tempb{outer}%
9268       \ifx\@tempa\@tempb
9269         \def\m@mm@argin{2}%
9270       \else
9271         \def\@tempb{inner}%
9272         \ifx\@tempa\@tempb
9273           \def\m@mm@argin{3}%
9274         \else
9275           \def\m@mm@argin{-1}%
9276         \fi
9277       \fi
9278     \fi
9279   \fi}
9280

```



```

\ifmemtortm A flag set TRUE if the marginalia is to be moved to the right of the text and
\memtortmtrue FALSE if it is to be moved to the left of the text.
\memtortmfalse 9281 % MEM-T0-RighT-Margin
                9282 \newif\ifmemtortm
                9283 \memtortmtrue

\m@mwhich@margin \m@mwhich@margin{<code>} sets \ifmemtortm TRUE or FALSE depending on
the value of <code> (0 to 3), and other factors.

```

Two column document If the first column, to the left, otherwise to the right, irrespective the document being one- or two-side.

One sided document If code = 0 (left), to the left, otherwise (code = 1 (right), 2 (outer), 3 (inner)) to the right.

Two sided document depends on whether a recto or verso page:

Recto (odd) page right if code = 1 (right) or 2 (outer), otherwise left.

Verso (even) page left if code = 0 (left) or 2 (outer), otherwise right.

```

9284 \newcommand*{\m@mwhich@margin}[1]{%
9285   \memtortmtrue
9286   \if@twocolumn
9287     \if@firstcolumn% left
9288       \memtortmfalse
9289     \else%                right
9290       \memtortmtrue
9291   \fi
9292 \else
9293   \if@twoside
9294     \checkoddpaper
9295     \ifcase #1\relax% 0 left
9296       \memtortmfalse
9297     \or%                1 right
9298       \memtortmtrue
9299     \or%                2 outer
9300       \ifoddpage
9301         \memtortmtrue
9302       \else
9303         \memtortmfalse
9304       \fi
9305     \or%                3 inner
9306       \ifoddpage
9307         \memtortmfalse
9308       \else
9309         \memtortmtrue
9310       \fi
9311     \fi% end ifcase
9312   \else%                oneseide
9313     \ifnum #1=\z@% 0 left, all else right

```

```

9314         \mentortmfalse
9315     \else
9316         \mentortmtrue
9317     \fi
9318 \fi% end if@twoside
9319 \fi}
9320

```

27.2 Marginpars

A `\marginpar` is a kind of floating object — you can't control exactly where it will go. There is one problem with the kernel definition of `\marginpar` in that sometimes a `\marginpar` may end up on the wrong side of the page. The following is an attempt to fix that using the odd/even page check provided as part of the class.

`\@addmarginpar` The part of the code for `\marginpar` that deals with deciding which side of the page it should be printed on is `\@addmarginpar`, buried away in the kernel's `output` routine. A couple of minor changes are made to the kernel code. The first is at the beginning where I have added the `\checkoddpages` page checking code. In the new margin placement syntax, there is a new syntax: always left, in twoside mode. This cannot be implemented using the normal `\marginpar` placement controls, so we have to add our own.

```

9321 \def\@addmarginpar{%
9322     \checkoddpages

```

Continue with the kernel code. The twocolumn stuff in the new syntax scheme is the same as in the kernel.

```

9323     \@next\@marbox\@currlist{\@cons\@freelist\@marbox
9324     \@cons\@freelist\@currbox}\@latexbug\@tempcnta\@ne
9325     \if@twocolumn
9326         \if@firstcolumn \@tempcnta\m@ne \fi
9327     \else

```

Here we add the new stuff. We make use of the fact that we have hidden all the page checking in `\m@mwhich@margin`, and we only fire it off if we know `\marginparmargin` have been used (via `\ifm@msetmp`).

```

9328     \ifm@msetmp%
9329         % \@tempcnta > 0 => right side of page
9330         % \@tempcnta < 0 => left side of page
9331         \m@mwhich@margin{\m@mmpar@margin}% set left or right margin
9332         \ifmentortm%
9333             \@tempcnta\@ne\relax%
9334         \else%
9335             \@tempcnta\m@ne\relax%
9336         \fi%
9337     \else%
9338         \if@mparswitch

```

The next line, reading

```
\ifodd\c@page \else\@tempcnta\m@ne \fi
```

is where the odd/even page checking is done in the kernel code. I replace it with my code, and then continue with the kernel.

```
9339      \ifoddpage \else \@tempcnta\m@ne \fi%
9340      \fi%
9341      \if@reversemargin \@tempcnta -\@tempcnta \fi%
9342      \fi%
9343      \fi%
9344      \ifnum\@tempcnta <\z@ \global\setbox\@marbox\box\@currbox \fi
9345      \@tempdima\@mparbottom
9346      \advance\@tempdima -\@pageht
9347      \advance\@tempdima\ht\@marbox
9348      \ifdim\@tempdima >\z@
```

The next line in the kernel reads:

```
\@latex@warning@no@line {Marginpar on page \thepage\space moved}
```

I have changed the warning message to give the user an indication of the severity of the move. Then follow the kernel on to the end.

```
9349      \@latex@warning@no@line {Marginpar on page
9350                                \thepage\space moved by \the\@tempdima}%
9351      \else
9352      \@tempdima\z@
9353      \fi
9354      \global\@mparbottom\@pageht
9355      \global\advance\@mparbottom\@tempdima
9356      \global\advance\@mparbottom\dp\@marbox
9357      \global\advance\@mparbottom\marginparpush
9358      \advance\@tempdima -\ht\@marbox
9359      \global\setbox \@marbox
9360              \vbox {\vskip \@tempdima
9361                      \box \@marbox}%
9362      \global \ht\@marbox \z@
9363      \global \dp\@marbox \z@
9364      \kern -\@pagedp
9365      \nointerlineskip
9366      \hb@xt@\columnwidth
9367      {\ifnum \@tempcnta >\z@
9368        \hskip\columnwidth \hskip\marginparsep
9369      \else
9370        \hskip -\marginparsep \hskip -\marginparwidth
9371      \fi
9372      \box\@marbox \hss}%
9373      \nointerlineskip
9374      \hbox{\vrule \@height\z@ \@width\z@ \@depth\@pagedp}%
9375  }
9376
```

27.2.1 Marginpar – margin interface

The ‘standard’ method of specifying the desired margin for a `\marginpar` consists of the two macros `\normalmarginpar` and `\reversemarginpar`, whose effects depend on whether the document is one- or two-sided, whether the page is recto or verso, and if it is set in one- or two-columns.

```

\ifm@setmp TRUE if the macro \marginparmargin has been called.
m@setmptrue 9377 \newif\ifm@setmp
m@setmpfalse 9378 \m@setmpfalse

\marginparmargin The user command for specifying the desired margin for \marginpars. The code
\m@mm@margin is stored as \m@mm@margin, and \setmpbools is called to convert to the
regular \marginpar booleans.

9379 \newcommand*{\marginparmargin}[1]{%
9380   \m@setmptrue
9381   \m@setm@argin{#1}%
9382   \ifnum\m@mm@argin<\z@
9383     \@memwarn{Bad \string\marginparmargin\space argument ‘#1’\MessageBreak
9384       set to ‘outer’}%
9385     \gdef\m@mm@margin{2}% set as outer
9386   \else
9387     \global\let\m@mm@margin\m@mm@argin
9388   \fi
9389   \setmpbools}
9390

\setmpbools Given \m@mm@margin, \setmpbools sets the corresponding \marginpar
location booleans.

9391 \newcommand*{\setmpbools}{%
9392   \if@twoside
9393     \@mparswitchtrue
9394   \else
9395     \@mparswitchfalse
9396   \fi
9397   \ifcase\m@mm@margin\relax% 0 left
9398     \@reversemargintrue% \sideparswitchfalse \reversesidepartrue
9399   \or% 1 right
9400     \@reversemarginfalse% \sideparswitchfalse \reversesideparfalse
9401   \or% 2 outer
9402     \@reversemarginfalse% \sideparswitchtrue \reversesideparfalse
9403   \or% 3 inner
9404     \@reversemargintrue% \sideparswitchtrue \reversesidepartrue
9405   \fi}
9406

9407 \newcommand*{\m@setmpcodes}{%
9408   \if@mparswitch% 2 sided
9409     \if@reversemargin% inner

```

```

9410     \def\m@mmpar@margin{3}%
9411     \else% outer
9412     \def\m@mmpar@margin{2}%
9413     \fi
9414     \else% 1 sided
9415     \if@reversemargin% left
9416     \def\m@mmpar@margin{0}%
9417     \else% right
9418     \def\m@mmpar@margin{1}%
9419     \fi
9420     \fi}
9421

```

27.3 A fixed marginpar

Introduce a non-floating marginpar.

`\parnopar` From *The T_EXbook* Exercise 14.15. It creates an ‘invisible’ end/start paragraph, and may be used for getting T_EX to try a pagebreak. Roman Eisle (email 2008/09/05) found a problem with the original definition as its effects continued on:
`\newcommand{\parnopar}{\parfillskip=0pt\par\parskip=0pt\noindent}`
 and he supplied the fix used below.

```

9422 \newcommand{\parnopar}{\parfillskip=0pt\par\parskip=0pt\noindent
9423                               \parfillskip\@flushglue}
9424

```

`\ifreversemargin` Analogues of `\marginpar` controls.²²

```

\ifsideparswitch 9425 \newif\ifreversemargin
9426 % \reversemarginfalse
9427 \reversemargintrue
9428 \newif\ifsideparswitch
9429 \sideparswitchfalse
9430 \if@twoside \sideparswitchtrue \fi
9431

```

`\ifm@msetsp`

```

\m@msetsptrue 9432 %% true if \sideparmargin used
\m@msetspfalse 9433 \newif\ifm@msetsp
9434 \m@msetspfalse

```

`\sideparmargin` User command for specifying the margin for `\sidepar`. The numeric code is stored in `\m@msetspmargin`.

```

9435 \newcommand*{\sideparmargin}[1]{%
9436     \m@msetsptrue

```

²²Note that for some now forgotten reason the default placement of the `\sidepar` is reversed in comparison to say `\marginpar`. As not to change existing documents, we have left the default like this.

```

9437 \m@msetm@argin{#1}%
9438 \ifnum\m@mm@argin<\z@
9439 \@@memwarn{Bad \string\sideparmargin\space argument '#1'\MessageBreak
9440         set to 'outer'}}%
9441 \gdef\m@m@spar@margin{2}% set as outer
9442 \else
9443 \global\let\m@m@spar@margin\m@mm@argin
9444 \fi}
9445

```

`\m@msidepar@left` Move a `\sidepar` into the left/right margin.

```

\m@msidepar@right 9446 \newcommand*{\m@msidepar@left}{%
9447 \@tempdimc\marginparwidth
9448 \advance\@tempdimc\marginparsep
9449 \kern-\@tempdimc}
9450 \newcommand*{\m@msidepar@right}{%
9451 \@tempdimc\columnwidth
9452 \advance\@tempdimc\marginparsep
9453 \kern\@tempdimc}
9454

```

`\setspbools` Given the `\m@m@spar@margin` code, calculates the corresponding `\sidepar` location booleans.

```

9455 \newcommand*{\setspbools}{%
9456 \ifcase\m@m@spar@margin\relax% 0 left
9457 \sideparswitchfalse \reversesidepartrue
9458 \or% 1 right
9459 \sideparswitchfalse \reversesideparfalse
9460 \or% 2 outer
9461 \sideparswitchtrue \reversesideparfalse
9462 \or% 3 inner
9463 \sideparswitchtrue \reversesidepartrue
9464 \fi}
9465

```

`\setspcode` Given the `\sidepar` location booleans, sets up the corresponding `\m@m@spar@margin` numeric codes.

```

9466 \newcommand*{\setspcode}{%
9467 \ifsideparswitch
9468 \ifreversesidepar
9469 \def\m@m@spar@margin{3}% inner
9470 \else
9471 \def\m@m@spar@margin{2}% outer
9472 \fi
9473 \else
9474 \ifreversesidepar
9475 \def\m@m@spar@margin{0}% left
9476 \else
9477 \def\m@m@spar@margin{1}% right

```

```

9478 \fi
9479 \fi}
9480

```

`\sideparfont` The font to be used for `\sidepars` and the alignment for it. The `\sideparform` holds code that will set the text flush against the edge of the text.²³

```

9481 \newcommand*{\sideparfont}{\normalfont\normalsize}
9482 \newcommand*{\sideparform}{\ifmemtortm\raggedright\else\raggedleft\fi}

```

`\sidepar` `\sidepar[<left>]{<right>}` (per `\marginpar`). Sidepars had a nasty habit of moving up or down depending on whether characters in the sidepar and the main text line have ascenders and/or descenders. The length `\sideparvshift` was provided to enable adjustments. This is Dan Luecking's modified version of my original code (CTT *Re: sidepars drift up a point*, 2006/04/11) which does a much better job, and the default for `\sideparvshift` is now 0pt.

```

9483 \newcommand{\sidepar}{\@dblarg{\@sidepar}}
9484 \long\def\@sidepar[#1]#2{\leavevmode\@bsphack\strut\vadjust{%
9485   \checkoddpage
9486   \ifm@msetsp% \sideparmargin used
9487   %%% \setspbools
9488   \else% \sideparmargin not used, set the \m@mshpar@margin code
9489   \setspcode
9490   \fi
9491   \rlap{\kern-\parindent
9492     \m@mwhich@margin{\m@mshpar@margin}% set left or right margin
9493     \ifmemtortm
9494       \m@mshpar@right
9495     \else
9496       \m@mshpar@left
9497     \fi
9498   \setbox0=\vtop to 0pt{%
9499     \begin{minipage}[t]{\marginparwidth}%
9500     \sideparform\sideparfont%
9501     \ifmemtortm #2\else #1\fi
9502     \end{minipage}%
9503     \vss}%
9504   \vtop to 0pt{\kern\sideparvshift% default should be 0pt
9505     \kern-\dp\strutbox
9506     \kern-\ht0
9507     \box0 \vss}}}%
9508   \@esphack}
9509

```

Vertical shift for sidepar to align with text line

```

9510 \newlength{\sideparvshift}
9511 \setlength{\sideparvshift}{0pt}
9512 %%% \setlength{\sideparvshift}{-2.08ex}% seems to work for all font sizes
9513

```

²³One might consider redefining this using the `\RaggedRight` and `\RaggedLeft` macros from the `ragged2e` package.

27.4 Sidebars

On 2002/10/22 Donald Arseneau posted the following code to CTT for adding sidebars to plain TeX.

```
\newinsert\sideins
\skip\sideins=0pt
\count\sideins=0
\dimen\sideins=2in

\def\sidebarvsep{25pt}
\def\sidebarhsep{15pt}

\let\mainpagecontents\pagecontents

\def\sidecontents{%
  \ifvoid\sideins\else
    {\advance\hsize\sidebarhsep
     \moveright\hsize \vtop to0pt{%
       \vskip-\sidebarvsep \vskip\topskip % offset by difference
       \unvbox\sideins \vss}%
    } \fi}
\def\pagecontents{\sidecontents\mainpagecontents}
\long\def\sidebar#1{
  \insert\sideins{%
    \splittopskip\sidebarvsep\relax
    \hsize 1.5in \rightskip=0pt plus 20pt \it
    \noindent \vbox to \sidebarvsep{}\ignorespaces #1%
    \ifhmode \unskip\strut\fi \par
  }%
}
```

The following started off (2002/03/19) as my attempt to rewrite the above into a form suitable for LaTeX, and this class in particular.

Donald Arseneau came up with some improvements to my sidebar code. The impetus for this came from a CTT thread, *whitespace after my command*, 2006/11/30 and earlier. In email to me on 2006/11/30 he said:

For memoir `\sidebar`, it seems the rules are expanding to fill the space for the sidebar, so here is my suggested change to `\sidecontents`.

It occurs to me that a separate `\sidetsep` (or `\sidetopsep`) would be more valuable than having an explicit `1\onelineskip`.

I grouped common code into two macros, and altered `\sidebarform` too.

Since then we have been going back and forth, with the result that practically all the complex code is Donald's.

`\sideins` Create a new insert called `\sideins`, which also creates a corresponding skip, count, dimen, and box.
`\skip\sideins` is the extra vertical space to allow on the page for the insert,
`\count\sideins` is the magnification factor for page breaking,
`\dimen\sideins` is the maximum insertion size (height) per page,
and the inserted material will be in `\box\sideins` when a page is output.

```
9514 \newinsert\sideins
9515 \skip\sideins=0pt
9516 \count\sideins=0
9517
```

`\sidebartopsep` Separation at the top of a sidebar.

```
9518 \newlength{\sidebartopsep}
9519 \setlength{\sidebartopsep}{0pt}
```

`\setsidebarheight` The macro `\setsidebarheight{<length>}` sets the total height of sidebars on a page to `<length>`. This is an interface for controlling `\dimen\sideins` properly, taking account of `\topskip` and the sidebar font size (and therefore should be invoked *after* declaring `\sidebarfont`).

Note that `\sidebartopsep` is *not* included as part of the allocated height.

```
9520 \newcommand{\setsidebarheight}[1]{%
9521 \setlength{\dimen\sideins}{#1}%
9522 \advance\dimen\sideins-\topskip
9523 \advance\dimen\sideins\ht\strutbox}
9524
```

`\sidebarhsep` The length `\sidebarhsep` is the gap between the typeblock and the sidebar.

`\sidebarvsep` `\sidebarvsep` is the vertical gap between sidebars on a page. The maximum

`\sidebarwidth` number of sidebar lines on a page is $1 + (\dimen\sideins - \sidebarvsep)$, assuming `\dimen\sideins` is defined in terms of `\onelineskip`. The width of the sidebar is `\sidebarwidth`.

```
9525 \newlength{\sidebarhsep}
9526 \newlength{\sidebarvsep}
9527 \newlength{\sidebarwidth}
```

`\sidebarfont` The font for typesetting the contents of a sidebar.

```
9528 \newcommand{\sidebarfont}{\normalfont\normalsize}
```

`\setsidebars` `\setsidebars{<hsep>}{<width>}{<vsep>}{<topsep>}{}{<height>}` sets the main `\sidebar` parameters. A `*` argument means leave the setting as is.

```
9529 \newcommand*\setsidebars[6]{%
9530 \nametest{#1}{*}\ifsamename\else
9531 \setlength{\sidebarhsep}{#1}\@memznegtest{\sidebarhsep}%
9532 \fi
9533 \nametest{#2}{*}\ifsamename\else
9534 \setlength{\sidebarwidth}{#2}\@memznegtest{\sidebarwidth}%
9535 \fi
9536 \nametest{#3}{*}\ifsamename\else
```

```

9537 \setlength{\sidebarvsep}{#3}\@memnegtest{\sidebarvsep}%
9538 \fi
9539 \nametest{#4}{*}\ifsamename\else
9540 \setlength{\sidebartopsep}{#4}%
9541 \fi
9542 \nametest{#5}{*}\ifsamename\else
9543 \def\sidebarfont{#5}%
9544 \fi
9545 \nametest{#6}{*}\ifsamename\else
9546 \setsidebarheight{#6}%
9547 \ifdim\dimen\sideins>\z@\else
9548 %%% \memerror{\protect\sidebarheight\space is zero or negative}{\@ehd}%
9549 \memwarn{\protect\sidebarheight\space is zero or negative}%
9550 \fi
9551 \fi}
9552 \setsidebars{\marginparsep}% sidebarhsep
9553 \marginparwidth}% sidebarwidth
9554 \onelineskip}% sidebarvsep
9555 {0pt}% sidebartopsep
9556 {\normalsize\normalfont}% sidebarfont
9557 {\textheight}% sidebarheight
9558

```

`\sidebarform` `\sidebarform` sets the ‘raggedness’ in a sidebar. The current definition is less ragged than `\raggedright`. Examples:

```

for flushletright \renewcommand{\sidebarform}{}
for raggedleft \renewcommand{\sidebarform}{\raggedleft}

```

```

9559 \newcommand{\sidebarform}{%\rightskip=\z@ \@plus 2em}
9560 %\memRTLrightskip=\z@ \@plus 2em
9561 \ifmemtortm\raggedright\else\raggedleft\fi%
9562 % \ifmemtortm\memRTLrightskip=\z@ \@plus 2em\else\memRTLleftskip=\z@ \@plus 2em\fi
9563 }
9564

```

`\ifsidebaroneside` Flag for oneside.

```

9565 \newif\ifsidebaroneside
9566 \if@twoside\sidebaronesidefalse\else\sidebaronesidettrue\fi
9567

```

`\sidebarmargin` `\sidebarmargin{<side>}` sets the side where sidebars will be printed. Valid arguments are: left, right, outer, or inner.

```

9568 \newcommand*{\sidebarmargin}[1]{%
9569 \m@setm@argin{#1}%
9570 \ifnum\m@mm@argin<\z@
9571 \memwarn{Bad \string\sidebarmargin\space argument ‘#1’\MessageBreak
9572 set to ‘outer’}%
9573 \gdef\m@msidebar@margin{2}% set as outer

```

```

9574 \else
9575 \global\let\m@msidebar@margin\m@mm@argin
9576 \fi}
9577 %%%% default outer
9578 \gdef\m@msidebar@margin{2}

```

`\m@sideb@left` Code used later to put sidebars into the desired margin.

```

\m@sideb@right 9579 \newcommand*{\m@sideb@left}{%
9580 \@tempdimc \sidebarwidth
9581 \advance\@tempdimc\sidebarhsep
9582 \kern-\@tempdimc}
9583 \newcommand*{\m@sideb@right}{%
9584 \@tempdimc \columnwidth% or \hsize
9585 \advance\@tempdimc\sidebarhsep
9586 \kern\@tempdimc}
9587

```

`\sidecontents` This locates the horizontal position of a sidebar. On two column pages it is put in the margin next to the column where `\sidebar` is called, otherwise it is put in the outer (foreedge) margin, except for the `oneside` option when it is put into the right hand margin. This can be separately controlled via `\ifsidebaroneside`.

Emanuele Vicentini²⁴ suggested to code for the `oneside` option.

```

9588 \newcommand{\sidecontents}{\hbox to \z@{%
9589 \m@which@margin{\m@msidebar@margin}%
9590 \ifmementortm
9591 \m@sideb@right
9592 \else
9593 \m@sideb@left
9594 \fi

```

DA and I have gone to and fro on the next bit of code trying to get the top alignment correct.

```

\vtop to0pt{%
\vskip\onelineskip
\unvbox\sideins \vss}%

```

DA has the last word.

```

9595 \vtop to0pt{%
9596 \normalsize\normalfont\sidebarfont% select font so we know the strut size
9597 \vskip\topskip \vskip-\ht\strutbox
9598 \vskip\sidebartopsep% extra vertical shift
9599 \unvbox\sideins \vss}%
9600 \hss}}
9601

```

`\sidebar` `\sidebar{<text>}` puts `<text>` into a sidebar.

²⁴emanuelevicentini@yahoo.it, private email, 2004/04/05

Florence Henry (florence.henry@obspm.fr) in the thread *Of memoir, sidebar, and justification*, 2003/04/02, pointed out that `itemize` in a `\sidebar` could overflow. DA gave the solution hint.

DA says that having stacked insertions position properly is difficult. Here he uses the size of the strut to regulate both the height and the depth of each insertion (much as for footnotes). The following is his code and commentary.

```
9602 \newcommand{\sidebar}[1]{%
9603   \insert\sideins{%
```

Begin the insertion with settings. The height of the strut box (dependant on the font) will determine the top alignment both initially and after a page break. The setting for maximum depth does not control anything; see the code further on instead.

```
9604     \hsize\sidebarwidth
9605     \@parboxrestore
```

Even though the margin placement is handled inside `\sidecontents`, there are some timing issues such that `\ifmementorm` is executed too late for `\sidebars` internals to make use of it. Fixed by adding the margin check to `\sidebar` as well.

```
9606     \m@mwhich@margin{\m@msidebar@margin}%
9607     \sidebarform\sidebarfont
9608     \splittopskip=\ht\strutbox
9609     \splitmaxdepth=\dp\strutbox % doesn't do anything useful
```

Allow a free split at the top (so this whole insertion moves to the next page if it does not fit).

```
9610     \allowbreak
```

Control the (vertical) positioning of non-split entries. Footnotes (and previous versions of `\sidebar`) use a strut at the beginning of the text, but we will allow a `baseline-skip` to perform the same function. This method also introduces a `\parskip` when the paragraph begins, so we counteract that. (The problem with an initial strut is that it messes' up entries that begin with vertical space.)

```
9611     \prevdepth=\dp\strutbox    % supersedes a "top-strut"
9612     \vskip-\parskip
```

Now the user's sidebar entry:

```
9613     #1%
```

If the entry ended still in a paragraph, take the chance to insert a final strut; then end the paragraph.

```
9614     \ifvmode\else
9615         \unskip\@finalstrut\strutbox
9616     \fi\par
```

Explicitly force the effect of `\maxdepth` (`\splitmaxdepth`), but using the depth of the strut in the rôle of `\maxdepth`.

```
9617     \ifdim\prevdepth>\dp\strutbox \prevdepth=\dp\strutbox \fi
```

Now control how adjacent entries abut (whether or not a final strut got inserted) and how an entry fits at the bottom of the page. Use `\vskips` to move from the text bottom to where a strut would bottom out, but insert an allowed breakpoint at the baseline position (so that the entry baseline may match the page's baseline). Finally insert the extra separation between entries.

```

9618     \ifdim\prevdepth>99\p@
9619         \nobreak
9620         \vskip-\prevdepth
9621         \allowbreak
9622         \vskip\dp\strutbox
9623     \fi
9624     \vskip\sidebarvsep}}
9625

```

27.5 Footnotes

For my reference, in case I want to fiddle with it even more than I have already done so, here is the kernel code (from `ltxfloat.dtx`) for footnotes. The insert for footnotes.

```

\newinsert\footins
\skip\footins=\bigskipamount
\count\footins=1000
\dimen\footins=8in

```

`\footnoterule` Draws the rule separating footnotes from the main text. It is executed after a `\vspace` of `\skip\footins` and should take no vertical space.

```

\def\footnoterule{\kern-3\p@ \hrule \@width 2in \kern 2.6\p@}

\@definecounter{footnote}
\def\thefootnote{\@arabic\c@footnote}
\@definecounter{mpfootnote}
\def\thempfootnote{\itshape\@alph\c@mpfootnote}

```

`\@makefnmark` Generates the footnote marker from `\@thefnmark`.

```

\def\@makefnmark{\hbox{\@textsuperscript{\normalfont\@thefnmark}}}

\DeclareRobustCommand*\textsuperscript[1]{%
  \@textsuperscript{\selectfont#1}}
\def\@textsuperscript#1{%
  {\m@th\ensuremath{\sim{\mbox{\fontsize\sf@size\z@#1}}}}}

\newdimen\footnotesep

```

`\footnote` `\footnote[num]{text}` is what the user uses for footnoting. It defines

`\@thefnmark` and calls `\@footnotemark` and `\@footnotetext` to do the work.

```
\def\footnote{\@ifnextchar[\@xfootnote{\stepcounter\@mpfn
\protected@xdef\@thefnmark{\thempfn}%
\@footnotemark\@footnotetext}}
```

`\@xfootnote` Handles the optional $\langle num \rangle$ argument to `\footnote`. It defines `\@thefnmark` and calls `\@footnotemark` and `\@footnotetext` to do the work.

```
\def\@xfootnote[#1]{%
\begingroup
\csname c@\@mpfn\endcsname #1\relax
\unrestored@protected@xdef\@thefnmark{\thempfn}%
\endgroup
\@footnotemark\@footnotetext}
```

`\@footnotetext` `\@footnotetext{\text}` sets up for typesetting the footnote (at the bottom of the page). It calls `\@makefntext{\text}` to actually do the typesetting (`\@makefntext` has to be supplied by a class or package).

```
\long\def\@footnotetext#1{%
\insert\footins{%
\reset@font\footnotesize
\interlinepenalty\interfootnotelinepenalty
\splittopskip\footnotesep
\splitmaxdepth \dp\strutbox \floatingpenalty \@MM
\hsize\columnwidth \@parboxrestore
\protected@edef\@currentlabel{%
\csname p@footnote\endcsname\@thefnmark}%
\color@begingroup
\@makefntext{%
\hrule\z@\footnotesep\ignorespaces#1\@finalstrut\strutbox}%
\color@endgroup}}
```

`\footnotemark` `\footnotemark[\mathit{num}]` the user uses this to produce just a footnote mark in the text.

```
\def\footnotemark{%
\@ifnextchar[\@xfootnotemark
{\stepcounter{footnote}%
\protected@xdef\@thefnmark{\thefootnote}%
\@footnotemark}}
```

`\@xfootnotemark` Handles the optional $\langle num \rangle$ argument to `\footnotemark`.

```
\def\@xfootnotemark[#1]{%
\begingroup
\c@footnote #1\relax
```

```

\unrestored@protected@xdef\@thefnmark{\thefootnote}%
\endgroup
\@footnotemark}

```

`\@footnotemark` Typesets the mark in the main text, via `\@makefnmark`.

```

\def\@footnotemark{%
  \leavevmode
  \ifhmode\edef\x@sf{\the\spacefactor}\nobreak\fi
  \@makefnmark
  \ifhmode\spacefactor\x@sf\fi
  \relax}

```

`\footnotetext` `\footnotetext[num]{text}` is user view for creating a footnote without a marker in the main text.

```

\def\footnotetext{%
  \ifnextchar [\@xfootnotenext
    {\protected@xdef\@thefnmark{\thempfn}%
    \@footnotetext}}

```

`\@xfootnotetext` Handles the optional *num* argument to `\footnotetext`.

```

\def\@xfootnotenext[#1]{%
  \begingroup
  \csname c@\@mpfn\endcsname #1\relax
  \unrestored@protected@xdef\@thefnmark{\thempfn}%
  \endgroup
  \@footnotetext}

```

```

\@mpfn
\thempfn
\def\@mpfn{footnote}
\def\thempfn{\thefootnote}

```

And from `ltboxes.dtx` for the `minipage` environment.

`\@mpfn` When setting up the environment, it includes:

```

\thempfn
\footnotetext
\def\@mpfn{mpfootnote}\def\thempfn{\thempfootnote}\c@mpfootnote\z@
\let\@footnotetext\@mpfootnotetext

```

`\@mpfootnotetext` Later it defines `\@mpfootnotetext{text}` as the `minipage` version of `\@footnotetext`.

```

\long\def\@mpfootnotetext#1{%
  \global\setbox\@mpfootins\vbox{%
    \unvbox\@mpfootins

```

```

\reset@font\footnotesize
\hsize\columnwidth
\@parboxrestore
\protected@edef\@currentlabel{%
  \csname p@mpfootnote\endcsname\@thefnmark}%
\color@begingroup
  \@makefnmark{%
    \rule\z@\footnotesep\ignorespaces#1\@finalstrut\strutbox}%
\color@endgroup}}

```

That completes the kernel extracts.

\footnoterule Usually, footnotes are separated from the main body of the text by a small rule. This rule is drawn by the macro `\footnoterule`. We have to make sure that the rule takes no vertical space (see `plain.tex`) so we compensate for the natural height of the rule of 0.4pt by adding the right amount of vertical skip. To prevent the rule from colliding with the footnote we first add a little negative vertical skip, then we put the rule and make sure we end up at the same point where we begun this operation.

```

9626 \renewcommand{\footnoterule}{%
9627   \kern-3\p@
9628   \hrule width .4\columnwidth
9629   \kern 2.6\p@}

```

And just to make sure that we are using memoir's font-dependent skips:

```

9630 \skip\footins=\bigskipamount

```

\c@footnote Footnotes are numbered per chapter. The counter is predefined.

```

9631 \@addtoreset{footnote}{chapter}

```

The footnote mechanism of L^AT_EX calls the macro `\@makefnmark` to produce the actual footnote. The macro gets the text of the footnote as its argument and should use `\@thefnmark` as the mark of the footnote. The macro `\@makefnmark` is called when effectively inside a `\parbox` of width `\columnwidth` (i.e., with `\hsize = \columnwidth`).

An example of what can be achieved is given by the following piece of T_EX code.

```

\long\def\@makefnmark#1{%
  \@setpar{\@par
    \@tempdima = \hsize
    \advance\@tempdima-10pt
    \parshape \@ne 10pt \@tempdima}%
  \par
  \parindent 1em\noindent
  \hbox to \z@\{\hss\@makefnmark\}#1}

```

The effect of this definition is that all lines of the footnote are indented by 10pt, while the first line of a new paragraph is indented by 1em. To change these

dimensions, just substitute the desired value for ‘10pt’ (in both places) or ‘1em’. The mark is flushright against the footnote.

The standard classes use a simpler macro, in which the footnote text is set like an ordinary text paragraph, with no indentation except on the first line of a paragraph, and the first line of the footnote. Thus, all the macro must do is set `\parindent` to the appropriate value for succeeding paragraphs and put the proper indentation before the mark.

```
\newcommand{\@makefntext}[1]{%
  \parindent 1em%
  \noindent
  \hb@xt@1.8em{\hss\@makefnmark}\#1}
```

This class provides a configurable specification.

Normally, if two or more footnotes are sequentially applied:

...text\footnote{first}\footnote{second}...

the markers in the text run together (e.g., ...text¹²). The class provides a separator between the markers (e.g., ...text^{1,2}). The underlying ideas for this are from the `ledmac` package [Wil03], which in turn got them from the `footmisc` package [Fai03].

`\multfootsep` The separator between multiple footnote markers.

```
9632 \newcommand*{\multfootsep}{\textsuperscript{\normalfont,}}
```

`\multiplefootnotemarker` `\m@mmf@prepare` is a pair of self-cancelling kerns.

```
\m@mmf@prepare 9633 \newcommand*{\multiplefootnotemarker}{3sp}
9634 \newcommand*{\m@mmf@prepare}{%
9635   \kern-\multiplefootnotemarker
9636   \kern\multiplefootnotemarker\relax}

\m@mmf@check If \m@mmf@check recognises the last kern as \multiplefootnotemarker it
typesets \multfootsep.
9637 \newcommand*{\m@mmf@check}{%
9638   \ifdim\lastkern=\multiplefootnotemarker\relax
9639     \edef\@x@sf{\the\spacefactor}%
9640     \unkern
9641     \multfootsep
9642     \spacefactor\@x@sf\relax
9643   \fi}
9644
```

`\@footnotetext` We have to modify the kernel’s `\@footnotetext` and `\@footnotemark` to
`\m@mold@footnotetext` implement the separator.

```
\@footnotemark 9645 \renewcommand{\@footnotetext}[1]{\insert\footins{%
9646   \def\baselinestretch{\m@m@spacespace}% <- v1.61803 addition
9647   \reset@font% <- v1.6180 addition
9648   \foottextfont
9649   \@preamfntext}}
```

```

9650 \hsize\columnwidth
9651 \protected@edef\@currentlabel{%
9652   \csname p@footnote\endcsname\@thefnmark}%
9653 \color@begingroup
9654   \@makefnmark{%
9655     \rule\z@\footnotesep\ignorespaces{\foottextfont #1}%
9656     \@finalstrut\strutbox}%
9657 \color@endgroup}\m@mmf@prepare}
9658 \let\m@mold@footnotetext\@footnotetext
9659 \renewcommand*{\@footnotemark}{%
9660   \leavevmode
9661   \ifhmode
9662     \edef\x@sf{\the\spacefactor}%
9663     \m@mmf@check
9664     \nobreak
9665   \fi
9666   \@makefnmark
9667   \m@mmf@prepare
9668   \ifhmode\spacefactor\x@sf\fi
9669   \relax}
9670

```

`\footmarkwidth` The mark is typeset right justified in a box with width `\footmarkwidth`. Second
`\footmarksep` and later lines of the text are offset `\footmarksep` from the end of the box.
`\footparindent` Paragraphs in footnotes are indented by `\parindent`.

```

9671 \newlength{\footmarkwidth}
9672 \newlength{\footmarksep}
9673 \newlength{\footparindent}

```

`\footmarkstyle` The marker is typeset according to `\footscript{<marker>}`. This can be
`\footscript` specified by the user via `\footmarkstyle`.

```

9674 \newcommand*{\footmarkstyle}[1]{\def\footscript##1{#1}}

```

`\makefootmarkhook` A vacuous macro that the user can redefine to do something useful?

```

9675 \newcommand{\makefootmarkhook}{}
9676

```

`\footfootmark` This macro typesets the footnote marker.

```

9677 \newcommand{\footfootmark}{%
9678   \ifdim\footmarkwidth < \z@
9679     \llap{\hb@xt@ -\footmarkwidth{%
9680       \hss\normalfont\footscript{\@thefnmark}}}%
9681     \hspace*{-\footmarkwidth}}%
9682   \else
9683     \ifdim\footmarkwidth = \z@
9684       { \normalfont\footscript{\@thefnmark}}%
9685     \else

```

Positive width.

```

9686      \hb@xt@\footmarkwidth{\hss\normalfont\footscript{\@thefnmark}}%
9687      \fi
9688      \fi}
9689

```

`\makefootmark` The class version of `\@makefntext`.

```

\@makefntext 9690 \newcommand{\makefootmark}[1]{%
9691     \leavevmode
9692     \parindent \footparindent\noindent
9693     \leftskip\footmarksep\relax
9694     \advance\leftskip \footmarkwidth \null\nobreak\hskip -\leftskip\relax
9695     \makefootmarkhook\relax
9696     \footfootmark #1}

```

In the thread *memoir and footnote.sty trouble*, 2003/04/02, Patrik Nyman (patrik.nyman@orient.su.se) noted that in order to stop `footnote.sty` (from `mdwtools`) barfing my original

```

\newcommand{\@makefntext}[1]{\makefootmark{#1}}

```

had to be changed.

```

9697 \newcommand{\@makefntext}[1]{\makefootmark #1}

```

All that now remains for the footer part is to set the defaults.

```

9698 \footmarkstyle{\textsuperscript{#1}}
9699 \setlength{\footmarkwidth}{1.8em}
9700 \setlength{\footmarksep}{-1.8em}
9701 \setlength{\footparindent}{1em}
9702

```

`\@makefnmark` The footnote markers that are printed in the text to point to the footnotes should be produced by the macro `\@makefnmark`. We use the default definition for it.

```

9703 %\renewcommand\@makefnmark{\hbox{\@textsuperscript{%
9704 %                                \normalfont\@thefnmark}}}

```

`\footref` Sometimes it is desirable to reference a footnote more than once. If a footnote is labelled (e.g., `\footnote{text\label{fn}}`) then the macro `\footref{<fn>}` will print the footnote marker. The code is essentially a much simplified version of `\footnotemark` from `ltxfloat.dtx`.

```

9705 \newcommand{\footref}[1]{%
9706     \begingroup
9707     \unrestored@protected@xdef\@thefnmark{\ref{#1}}%
9708     \endgroup
9709     \@footnotemark}
9710

```

The following code for footnotes that can include verbatims is based on Jeremy Gibbons' *Footnotes with verbatim material* in the column *'Hey — it works'* (edited by Jeremy Gibbons), TeX and TUG NEWS, vol 2 no. 4, p 9, October 1993. I have updated the basic code so that it works for LaTeX2e.

`\verbfootnote` `\verbfootnote` is just like `\footnote` except that it can contain verbatim material.

```

9711 \def\verbfootnote{\@ifnextchar[\@xverbfootnote{\stepcounter\@mpfn
9712   \protected@xdef\@thefnmark{\thempfn}%
9713   \@footnotemark\@verbfootnotetext}}
9714
9715 \def\@xverbfootnote[#1]{%
9716   \begingroup
9717     \csname c@\@mpfn\endcsname #1\relax
9718     \unrestored@protected@xdef\@thefnmark{\thempfn}%
9719   \endgroup
9720   \@footnotemark\@verbfootnotetext}
9721

```

`\@verbfootnotetext` This is the secret ingredient. It is based on the `\footnote` macro in *The T_EXbook* page 363, which somehow manages to read an argument with verbatims. As Knuth says that it is subtle and involves trickery, don't expect me to even try to explain anything.

I'm not at all sure about the color bits, though!

```

9722 \long\def\@verbfootnotetext{%
9723   \insert\footins\bgroup
9724     \footnotesize
9725     \interlinepenalty\interfootnotelinepenalty
9726     \splittopskip\footnotesep
9727     \splitmaxdepth \dp\strutbox \floatingpenalty \@MM
9728     \hsize\columnwidth \@parboxrestore
9729     \edef\@currentlabel{\csname p@footnote\endcsname\@thefnmark}%
9730   %%% \color@begingroup
9731     \@makefntext{\rule{\z@}{\footnotesep}\ignorespaces}%
9732     \futurelet\next\fo@t
9733 }

```

`\fo@t` Guess what these do.

```

\fo@t 9734 \def\fo@t{\ifcat\bgroup\noexpand\next \let\next\fo@t
\fo@t 9735   \else \let\next\fo@t\fi \next}
\@foot 9736 \def\fo@t{\bgroup\aftergroup\@foot\let\next}
9737 \def\fo@t#1{%
9738   \color@begingroup
9739   #1\@foot
9740   \color@endgroup}
9741 \def\@foot{%
9742   \strut\egroup
9743   %%% \color@endgroup
9744   \m@mmf@prepare
9745 }
9746

```

`\@verbmpfootnotetext` Footnotes in minipages are a little different, so another version of `\...footnotetext`.

```

9747 \long\def\@verbmpfootnotetext{%
9748   \global\setbox\@mpfootins\vbox{%
9749     \reset@font\footnotesize
9750     \unvbox\@mpfootins
9751     \bgroup
9752     \hsize\columnwidth
9753     \@parboxrestore
9754     \edef\@currentlabel{\csname p\@mpfootnote\endcsname\@thefnmark}%
9755     \color@begingroup
9756     \@makefntext{\rule{\z@}{\footnotesep}\ignorespaces}%
9757     }
9758     \futurelet\next\fo@t
9759   }
9760

```

`\@minipagerestore` To get the `\verbfootnote` to work in a minipage we have to use `\@verbmpfootnotetext` instead of `\@verbfootnotetext`. There is a nice hook in the kernel minipage code for this. As of version 1.61803 the following code is placed earlier.

```

\def\@minipagerestore{\let\@verbfootnotetext\@verbmpfootnotetext}

```

27.6 Major extensions for footnotes

```

9761 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% major extension to footnoting
9762

```

This all requires modifications to the output routine's `\@makecol`, which will be done later.

```

\setfootnoterule Per Starbäck found (CTT feetbelowfloat in memoir, 2006/11/24) that
\feetbelowfloat did not affect anything when \raggedbottom is in effect.
\setfootnoterule[\vfill]{\langleuplift\rangle}{\langlewidth\rangle}{\langlethickness\rangle} defines the
\footnoterule. Memoir's default setting is
\setfootnoterule{3pt}{0.4\columnwidth}{\normalrulethickness}
To force footnotes to the bottom after a \raggedbottom use
\setfootnoterule[\vfill]{3pt}{0.4\columnwidth}{\normalrulethickness}

9763 \newcommand*\setfootnoterule[4][\%
9764   \def\footnoterule{\kern -#2\relax #1\relax
9765     \hrule width #3\relax
9766     \kern #2\kern-#4}}
9767 \setfootnoterule{3pt}{0.4\columnwidth}{\normalrulethickness}
9768

```

All sorts of extra kinds of footnotes.

```

\m@mdoextrafeet These macros are hooks into \@makecol.
\extrafeetins 9769 \newcommand{\m@mdoextrafeet}{\extrafeetins}
\extrafeetinshook 9770 \newcommand*\extrafeetins{%
9771   \setbox\@outputbox \vbox{%
9772     \boxmaxdepth \@maxdepth

```

```

9773 \unvbox\@outputbox
9774 \ifvoid\footinsv@r\else\@footstartv@r\@footgroupv@r\fi
9775 \extrafeetinshook}}
9776 \newcommand{\extrafeetinshook}{}
9777

```

`\m@dodoreinextrafeet` Hooks into `\@reinserts` for multiple footnote kinds.

```

\extrafeetreinshook 9778 \newcommand{\m@dodoreinextrafeet}{%
9779 \ifvoid\footinsv@r\else\insert\footinsv@r{\unvbox\footinsv@r}\fi
9780 \extrafeetreinshook}
9781 \newcommand{\extrafeetreinshook}{}
9782

```

`\foottextfont` General macros.

```

\footinsdim 9783 \newcommand{\foottextfont}{\footnotesize}
\@preamfntext 9784 \newlength{\footinsdim}
9785 \setlength{\footinsdim}{8in} % standard for \dimen\footins
9786 \newcommand{\@preamfntext}{%
9787 \interlinepenalty\interfootnotelinepenalty
9788 \floatingpenalty \@MM
9789 \splittopskip=\footnotesep
9790 \splitmaxdepth=\dp\strutbox
9791 \@parboxrestore}
9792

```

We are going to change some of the kernel footnote macros

```

\@mpfootnotetext
\m@mold@mmpfootnotetext 9793 \renewcommand{\@mpfootnotetext}[1]{%
9794 \global\setbox\@mpfootins\ vbox{%
9795 \unvbox\@mpfootins
9796 \def\baselinestretch{\m@m@singlespace}% <- v1.61803 addition
9797 \foottextfont \hsize\columnwidth \@parboxrestore
9798 \protected@edef\@currentlabel{%
9799 \csname p@mpfootnote\endcsname\@thefnmark}%
9800 \color@begingroup
9801 \reset@font% <- v1.6180 addition
9802 \@makefntext{%
9803 \rule{z@\footnotesep}\ignorespaces{\foottextfont #1}%
9804 \@finalstrut\strutbox}%
9805 \color@endgroup}}
9806

```

Save our revised version of `\@mpfootnotetext`

```

9807 \let\m@mold@mmpfootnotetext\@mpfootnotetext
9808

```

We also to patch the minipage environment. We can use `\@minipagerestore` for the `begin` part (done earlier), but have to modify `\endminipage`

```

\m@mdoextrafeetmini
\extrafeetminihook 9809 \newcommand{\m@mdoextrafeetmini}{%
\@minipagerestore 9810 \extrafeetminihook}
9811 \newcommand{\extrafeetminihook}{%
9812 %%%\renewcommand{\@minipagerestore}{\m@mdoextrafeetmini}
9813

\extrafeetendmini
\extrafeetendminihook 9814 \newcommand{\extrafeetendmini}{%
9815 \ifvoid\@mpfootinsv@r\else
9816 \vskip\skip\@mpfootins
9817 \normalcolor\footnoterule\mp@footgroupv@r
9818 \fi
9819 \extrafeetendminihook}
9820 \newcommand{\extrafeetendminihook}{%
9821

\m@mdoextrafeetendmini This is our patched version of \endminipage.
\endminipage 9822 \newcommand{\m@mdoextrafeetendmini}{\extrafeetendmini}
9823 \def\endminipage{%
9824 \par
9825 \unskip
9826 \ifvoid\@mpfootins\else
9827 \vskip\skip\@mpfootins
9828 \normalcolor
9829 \footnoterule
9830 \unvbox\@mpfootins
9831 \fi
9832 \m@mdoextrafeetendmini
9833 \@minipagefalse
9834 \color@endgroup
9835 \egroup
9836 \expandafter\@iiiparbox\@mpargs{\unvbox\@tempboxa}}
9837

\plainfootnotes The declaration for standard footnotes; easy, just use the saved versions of
\@footnotetext and \@mpfootnotetext.
9838 \newcommand{\plainfootnotes}{%
9839 \let\@footnotetext\m@mold@footnotetext
9840 \let\@mpfootnotetext\m@mold@mpfootnotetext}
9841

```

Now for lots of somewhat tedious code, interspersed with complex ‘Dirty Tricks’ type things.

Robert Schlicht ²⁵ [RS] has provided valuable help in tuning up the code.

²⁵Private email, 2004/03/12, (w.m.l@gmx.net)

`\newfootnoteseries` `\newfootnoteseries{<series>}` creates the set of macros required for footnote code (see the kernel code presented earlier). The created macros have `<series>` appended to their name.

```

9842 \newcommand{\newfootnoteseries}[1]{%
9843   \expandafter\newinsert\csname footins#1\endcsname% -> \footins#1
9844   \expandafter\skip\csname footins#1\endcsname \bigskipamount%
9845   %% - > \skip\footins#1 % [RS]
9846   \newcounter{footnote#1}% -> \c@footnote#1
9847   \@nameuse{c@footnote#1} \z@ -> \c@footnote#1=0
9848   \global\@namelet{p@footnote#1} \@empty% -> \p@footnote#1
9849   \@namedef{thefootnote#1}{\arabic{footnote#1}}% -> \thefootnote#1
9850   \@namedef{foottextfont#1}{\foottextfont}% -> \foottextfont#1
9851   \m@makefootnote{#1}% -> \footnote#1
9852   \m@make@xfootnote{#1}% -> \@xfootnote#1
9853   \m@make@footnotetext{#1}% -> \@footnotetext#1
9854   \m@makefootnotemark{#1}% -> \footnotemark#1
9855   \m@make@xfootnotemark{#1}% -> \@xfootnotemark#1
9856   \m@make@footnotemark{#1}% -> \@footnotemark#1
9857   \m@makefootnotetext{#1}% -> \footnotetext#1
9858   \m@make@xfootnotenext{#1}% -> \@xfootnotenext#1
9859   \m@make@mpfn{#1}% -> \@mpfn#1
9860   \m@makethempfn{#1}% -> \thempfn#1
9861   \m@make@makefnmark{#1}% -> \@makefnmark#1
9862   \m@makefootref{#1}% -> \footref#1
9863   \m@makefootfootmark{#1}% -> \footfootmark#1
9864   \m@makemakefootmark{#1}% -> \makefootmark#1
9865   \m@makefootmarkstyle{#1}% -> \footmarkstyle#1
9866   \@namedef{@makefntext#1}##1{\@nameuse{makefootmark#1} ##1}%
9867   \m@make@footstart{#1}% -> \@footstart#1
9868   \m@make@footgroup{#1}% -> \@footgroup#1
9869   \expandafter\newinsert\csname @mpfootins#1\endcsname% -> \@mpfootins#1
9870   \newcounter{mpfootnote#1}% -> \c@mpfootnote#1
9871   \global\@namelet{p@mpfootnote#1}\@empty
9872   \@namedef{thempfootnote#1}{\itshape\alph{mpfootnote#1}}%
9873   \m@make@mpfootnotetext{#1}% -> \@mpfootnotetext#1

```

Reset the counter per chapter, except for articles.

```

9874 \ifartopt\else% [RS]
9875   \expandafter\@cons\csname cl@chapter\endcsname {{footnote#1}}%
9876 \fi

```

Add the footnote to the (re)insert hooks.

```

9877 \g@addto@macro{\extrafeetinshook}{%
9878   \ifvoid\@nameuse{footins#1}\else
9879     \@nameuse{@footstart#1}\@nameuse{@footgroup#1}\fi}
9880 \g@addto@macro{\extrafeetreinshook}{%
9881   \ifvoid\@nameuse{footins#1}\else
9882     \insert\@nameuse{footins#1}{\unvbox\@nameuse{footins#1}}\fi}
9883 \g@addto@macro{\extrafeetendminihook}{%
9884   \ifvoid\@nameuse{@mpfootins#1}\else

```



```

9885     \vskip\skip\@mpfootins
9886     \normalcolor\footnoterule\@nameuse{mp@footgroup#1}\fi}
9887 \g@addto@macro{\extrafeetminihook}{%
9888   \@namedef{mpfn#1}{mpfootnote#1}
9889   \@namedef{thempfn#1}{\@nameuse{thempfootnote#1}}
9890   \csname c@mpfootnote#1\endcsname\z@
9891   \expandafter\let\expandafter\@t@mp \csname @mpfootnotetext#1\endcsname
9892   \expandafter\let \csname @footnotetext#1\endcsname \@t@mp}
9893 \g@addto@macro{\@mem@extranofeet}{% % [RS]
9894   \ifvoid\@nameuse{footins#1}\else\@mem@nofootfalse\fi}
9895 \plainfootstyle{#1}%
9896 }
9897

```

`\m@makefootnote` `\m@makefootnote{<series>}` creates `\footnote<series>`

```

9898 \newcommand{\m@makefootnote}[1]{
9899   \@namedef{footnote#1}{\@ifnextchar[
9900     {\@nameuse{@xfootnote#1}}{\advance \@nameuse{c@\@mpfn#1} by \@ne
9901       \stepcounter{\@mpfn#1}%
9902       \@name@p@xdef{@thefnmark#1}{\@nameuse{thempfn#1}}%
9903       \@nameuse{@footnotemark#1}\@nameuse{@footnotetext#1}}}}
9904

```

`\m@makexfootnote` `\m@makexfootnote{<series>}` creates `\xfootnote<series>`

```

9905 \newcommand{\m@makexfootnote}[1]{
9906   \@namedef{xfootnote#1}[##1]{%
9907     \begingroup
9908       \csname c@\@mpfn#1\endcsname ##1\relax
9909       \@name@unresp@xdef{@thefnmark#1}{\@nameuse{thempfn#1}}%
9910     \endgroup
9911     \@nameuse{@footnotemark#1}\@nameuse{@footnotetext#1}}
9912

```

`\m@make@footnotetext` `\m@make@footnotetext{<series>}` creates `\@footnotetext<series>`

```

9913 \newcommand{\m@make@footnotetext}[1]{%
9914   \@namelongdef{@footnotetext#1}##1{%
9915     \insert\@nameuse{footins#1}{%
9916       \def\baselinestretch{\m@m@single space}% <- v1.61803 addition
9917       \reset@font\@nameuse{foottextfont#1}%
9918       \@preamfntext
9919       \hsize\columnwidth
9920       \protected@edef\@currentlabel{%
9921         \csname p@footnote#1\endcsname\@nameuse{@thefnmark#1}}%
9922       \color@begingroup
9923         \@nameuse{@makefntext#1}{%
9924           \rule\z@\footnotesep\ignorespaces{\@nameuse{foottextfont#1}##1}% <- v1.6180339a
9925           \@finalstrut\strutbox}%
9926       \color@endgroup}%
9927   \m@mmf@prepare}}
9928

```

`\m@make@mpfootnotetext` `\m@make@mpfootnotetext{<series>}` creates `\@mpfootnotetext<series>`

```

9929 \newcommand{\m@make@mpfootnotetext}[1]{%
9930   \@namelongdef{\mpfootnotetext#1}##1{%
9931     \global\setbox\@nameuse{\mpfootins#1}\vbox{%
9932       \unvbox\@nameuse{\mpfootins#1}%
9933       \def\baselinestretch{\m@msinglespace}% <- v1.61803 addition
9934       \reset@font\@nameuse{\foottextfont#1}%
9935       \hsize\columnwidth \@parboxrestore
9936       \protected@edef\@currentlabel{%
9937         \csname p@mpfootnote#1\endcsname\@nameuse{\@thefnmark#1}}%
9938       \color@begingroup
9939         \@nameuse{\makefnmark#1}{%
9940           \rule{\z@}{\footnotesep}\ignorespaces{\@nameuse{\foottextfont#1}##1}% <- v1.6180339
9941           \@finalstrut\strutbox}%
9942       \color@endgroup}%
9943   \m@mmf@prepare}}
9944
```

`\m@makefootnotemark` `\m@makefootnotemark{<series>}` creates `\footnotemark<series>`

```

9945 \newcommand{\m@makefootnotemark}[1]{
9946   \@namedef{\footnotemark#1}{%
9947     \@ifnextchar[ {\@nameuse{\xfootnotemark#1}}
9948     {%\advance\@nameuse{c@footnote#1} by \@ne%
9949       \stepcounter{footnote#1}%
9950       \@name@p@xdef{\@thefnmark#1}{\@nameuse{\thefootnote#1}}%
9951       \@nameuse{\footnotemark#1}}}%
9952
```

`\m@make@xfootnotemark` `\m@make@xfootnotemark{<series>}` creates `\@xfootnotemark<series>`

```

9953 \newcommand{\m@make@xfootnotemark}[1]{%
9954   \@namedef{\xfootnotemark#1}[##1]{%
9955     \begingroup
9956       \@nameuse{c@footnote#1} ##1\relax
9957       \@name@unresp@xdef{\@thefnmark#1}{\@nameuse{\thefootnote#1}}%
9958     \endgroup
9959     \@nameuse{\footnotemark#1}}%
9960
```

`\m@make@footnotemark` `\m@make@footnotemark{<series>}` creates `\@footnotemark<series>`

```

9961 \newcommand{\m@make@footnotemark}[1]{%
9962   \@namedef{\footnotemark#1}{%
9963     \leavevmode
9964     \ifhmode
9965       \edef\x@sf{\the\spacefactor}%
9966       \m@mmf@check
9967       \nobreak
9968     \fi
9969     \@nameuse{\makefnmark#1}%
9970     \m@mmf@prepare

```

```

9971 \ifhmode\spacefactor\@x@sf\fi
9972 \relax}}
9973

```

`\m@makefootmarkstyle` `\m@makefootmarkstyle{<series>}` creates `\footmarkstyle<series>`

```

9974 \newcommand{\m@makefootmarkstyle}[1]{%
9975 \@namedef{footmarkstyle#1}##1{%
9976 \@namedef{footscript#1}####1{##1}}}
9977

```

`\m@makefootnotetext` `\m@makefootnotetext{<series>}` creates `\footnotetext<series>`

```

9978 \newcommand{\m@makefootnotetext}[1]{%
9979 \@namedef{footnotetext#1}{%
9980 \ifnextchar[ {\@nameuse{@xfootnotenext#1}}%
9981 {\@name@p@xdef{@thefnmark#1}{\@nameuse{thempfn#1}}}%
9982 \@nameuse{@footnotetext#1}}}}
9983

```

`\m@make@xfootnotenext` `\m@make@xfootnotenext{<series>}` creates `\@xfootnotenext<series>`

```

9984 \newcommand{\m@make@xfootnotenext}[1]{
9985 \@namedef{@xfootnotenext#1}[##1]{%
9986 \begingroup
9987 \csname c@{\@mpfn#1}\endcsname ##1\relax
9988 \@name@unresp@xdef{@thefnmark#1}{\@nameuse{thempfn#1}}%
9989 \endgroup
9990 \@nameuse{@footnotetext#1}}}
9991

```

`\m@make@mpfn` `\m@make@mpfn{<series>}` creates `\@mpfn<series>`

```

9992 \newcommand{\m@make@mpfn}[1]{%
9993 \@namedef{@mpfn#1}{\@nameuse{footnote#1}}}
9994

```

`\m@makethempfn` `\m@makethempfn{<series>}` creates `\thempfn<series>`

```

9995 \newcommand{\m@makethempfn}[1]{%
9996 \@namedef{thempfn#1}{\@nameuse{thefootnote#1}}}
9997

```

`\m@make@makefnmark` `\m@make@makefnmark{<series>}` creates `\@makefnmark<series>`

```

9998 \newcommand{\m@make@makefnmark}[1]{%
9999 \@namedef{@makefnmark#1}{%
10000 \hbox{\@textsuperscript{\normalfont\@nameuse{@thefnmark#1}}}}
10001

```

`\m@makefootref` `\m@makefootref{<series>}` creates `\footref<series>`

```

10002 \newcommand{\m@makefootref}[1]{%
10003 \@namedef{footref#1}##1{%
10004 \begingroup
10005 \@name@unresp@xdef{@thefnmark#1}{\ref{##1}}}%

```

```

10006 \endgroup
10007 \@nameuse{@footnotemark#1}}
10008

```

`\m@makefootfootmark` `\m@makefootfootmark{<series>}` creates `\footfootmark<series>`

```

10009 \newcommand{\m@makefootfootmark}[1]{%
10010   \@namedef{footfootmark#1}{%
10011     \ifdim\footmarkwidth < \z@
10012       \llap{\hb@xt@-\footmarkwidth{%
10013         \hss\normalfont\@nameuse{footscript#1}%
10014         {\@nameuse{@thefnmark#1}}}%
10015       \hspace*{-\footmarkwidth}}%
10016     \else
10017       \ifdim\footmarkwidth = \z@
10018         {\normalfont\@nameuse{footscript#1}{\@nameuse{@thefnmark#1}}}%
10019       \else
10020         \hb@xt@\footmarkwidth{%
10021           \hss\normalfont\@nameuse{footscript#1}%
10022           {\@nameuse{@thefnmark#1}}}%
10023       \fi
10024     \fi}}
10025

```

`\m@makemakefootmark` `\m@makemakefootmark{<series>}` creates `\makefootmark<series>`

```

10026 \newcommand{\m@makemakefootmark}[1]{%
10027   \@namedef{makefootmark#1}##1{%
10028     \leavevmode
10029     \parindent \footparindent\noindent
10030     \leftskip\footmarksep\relax
10031     \advance\leftskip \footmarkwidth
10032     \null\nobreak\hskip -\leftskip\relax
10033     \makefootmarkhook\relax
10034     \@nameuse{footfootmark#1}##1}}
10035

```

`\m@make@footgroup` `\m@make@footgroup{<series>}` creates `\@footgroup<series>`

```

10036 \newcommand{\m@make@footgroup}[1]{%
10037   \@namedef{@footgroup#1}{\unvbox\@nameuse{footins#1}}}
10038

```

`\m@makemp@footgroup` `\m@makemp@footgroup{<series>}` creates `\mp@footgroup<series>`

```

10039 \newcommand{\m@makemp@footgroup}[1]{%
10040   \@namedef{mp@footgroup#1}{\unvbox\@nameuse{mpfootins#1}}}
10041

```

`\m@make@footstart` `\m@make@footstart{<series>}` creates `\@footstart<series>`

```

10042 \newcommand{\m@make@footstart}[1]{%
10043   \@namedef{@footstart#1}{%
10044     \vskip\bigskipamount

```

```

10045 \leftskip=\z@
10046 \rightskip=\z@
10047 \footnoterule}}
10048

```

`\plainfootstyle` `\plainfootstyle{<series>}` specifies a plain (normal) footnote style for `<series>`.

```

10049 \newcommand{\plainfootstyle}[1]{%
10050 \m@make@footnotetext{#1}%
10051 \m@make@footgroup{#1}%
10052 \m@make@footstart{#1}%
10053 \m@make@mpfootnotetext{#1}%
10054 \m@makemp@footgroup{#1}%
10055 \@nameuse{footmarkstyle#1}{\textsuperscript{##1}}
10056 \expandafter\dimen\csname footins#1\endcsname=\footinsdim
10057 \expandafter\count\csname footins#1\endcsname=1000\relax}
10058

```

Now the basic code for footnote declarations (we will be using `footnotev@r` instead of the regular `footnote`).

`\footinsv@r` Define the new `\footinsv@r` insert.

```

10059 \newinsert\footinsv@r
10060 \skip\footinsv@r\bigskipamount
10061 \count\footinsv@r=1000 % no magnification
10062 \dimen\footinsv@r=\footinsdim

```

Create a new `v@r` footnote series.

```

10063 \m@make@footstart{v@r}

```

`\@footgroupv@r` Initialise the `footgroup` for the series to do nothing.

```

10064 \newcommand{\@footgroupv@r}{}
10065

```

And also for minipages:

```

\@mpfootinsv@r
\mp@footgroupv@r10066 \newinsert\@mpfootinsv@r
10067 \newcommand{\mp@footgroupv@r}{}
10068

```

ledmac package style footnotes (see `ledmac.dtx` for more detailed explanations).

The two- and three-column notes use a macro `\m@mrigidbodybalance` to split text into a number of columns. This is based on *The T_EXbook*, page 397.

```

\m@m@k
\m@m@h10069 \newcount\m@m@k \newdimen\m@m@h

```

```

\m@mrigidbalance \m@mrigidbalance{<box>}{<num>}{<length>} splits a <box> (of text) into <num>
\m@mdosplits columns with <length> space between the top baseline and the top of the \vbox.
\m@msplitoff 10070 \newcommand*{\m@mrigidbalance}[3]{\setbox0=\box#1 \m@m@k=#2 \m@m@h=#3
10071 \@@line{\splittopskip=\m@m@h \vbadness=\@M \hfilneg
10072 \valign{##\vfill\cr\m@mdosplits}}}}
10073 \newcommand*{\m@mdosplits}{\ifnum\m@m@k>0 \noalign{\hfil}\m@msplitoff
10074 \global\advance\m@m@k-1\cr\m@mdosplits\fi}
10075 \newcommand*{\m@msplitoff}{\dimen0=\ht0
10076 \divide\dimen0 by\m@m@k \advance\dimen0 by\m@m@h
10077 \setbox2 \vsplit0 to \dimen0
10078 \unvbox2 }
10079

```

27.6.1 Two column footnotes

`\twocolumnfootnotes` Declaration for two column footnotes. This causes the standard `\footnote` and friends to internally use the `v@r` series, which here produces two column footnotes.

```

10080 \newcommand{\twocolumnfootnotes}{%
10081 \namedef{foottextfontv@r}{\foottextfont}% % [RS]
10082 \let\@footnotetext\@twocolfootnotetext
10083 \dimen\footinsv@r=2\footinsdim
10084 \count\footinsv@r=500\relax
10085 \make@twocol@footgroup{v@r}%
10086 \let\@footgroupv@r\@twocol@footgroupv@r
10087 \let\@mpfootnotetext\@mptwocolfootnotetext
10088 \make@mptwocol@footgroup{v@r}%
10089 \let\mp@footgroupv@r\@mptwocol@footgroupv@r}
10090

```

`\@twocolfootnotetext` `\@twocolfootnotetext{<text>}` is the two column version of `\@footnotetext` for the `v@r` series.

```

10091 \newcommand{\@twocolfootnotetext}[1]{\insert\footinsv@r{%
10092 \def\baselinestretch{\m@m@singlespace}% <- v1.61803 addition
10093 \reset@font\foottextfont
10094 \@preamfntext
10095 \protected@edef\@currentlabel{%
10096 \csname p@footnote\endcsname\@thefnmark}%
10097 \color@begingroup
10098 \@twocolfootfmt{#1}%
10099 \color@endgroup}%
10100 \m@m@f@prepare}
10101

```

`\@preamtwofmt` Give each column 0.45 of the textwidth.

```

10102 \newcommand{\@preamtwofmt}{%
10103 \hsize .45\hsize
10104 \parindent=\z@
10105 \tolerance=5000\relax

```

```

10106 \raggedright
10107 \leavevmode}
10108

```

```

\@twocolfootfmt \@twocolfootfmt

```

```

10109 \newcommand{\@twocolfootfmt}[1]{%
10110 \preamtwofmt
10111 {\footfootmark\strut {\foottextfont #1}\strut\par}\allowbreak}
10112

```

`\@mptwocolfootnotetext` `\@mptwocolfootnotetext{<text>}` is the two column version of `\@mpfootnotetext` for the `v@r` series for minipages.

```

10113 \newcommand{\@mptwocolfootnotetext}[1]{%
10114 \global\setbox\@mpfootinsv@r\vbox{%
10115 \unvbox\@mpfootinsv@r
10116 \def\baselinestretch{\m@msinglespace}% <- v1.61803 addition
10117 \reset@font\foottextfont
10118 \hsize\columnwidth \@parboxrestore
10119 \protected@edef\@currentlabel{%
10120 \csname p@mpfootnote\endcsname\@thefnmark}%
10121 \color@begingroup
10122 \@twocolfootfmt{#1}%
10123 \color@endgroup}%
10124 \m@mmf@prepare}
10125

```

`\twocolumnfootstyle` `\twocolumnfootstyle{<series>}` specifies a two column footnote style for `<series>`.

```

10126 \newcommand{\twocolumnfootstyle}[1]{%
10127 \m@make@twocolfootnotetext{#1}%
10128 \m@make@mptwocolfootnotetext{#1}%
10129 \m@make@twocolfootfmt{#1}%
10130 \m@make@twocol@footgroup{#1}%
10131 \m@make@mptwocol@footgroup{#1}%
10132 \m@make@footstart{#1}%
10133 \@namelongdef{\@footnotetext#1}##1{%
10134 \@nameuse{\@twocolfootnotetext#1}{##1}}%
10135 \@namelongdef{\@mpfootnotetext#1}##1{%
10136 \@nameuse{\@mptwocolfootnotetext#1}{##1}}%
10137 \@namedef{\@footgroup#1}{\@nameuse{\@twocol@footgroup#1}}%
10138 \@namedef{\mp@footgroup#1}{\@nameuse{\@mptwocol@footgroup#1}}%
10139 \expandafter\dimen\csname footins#1\endcsname=2\footinsdim
10140 \expandafter\count\csname footins#1\endcsname=500\relax}
10141

```

`\m@make@twocolfootnotetext` `\m@make@twocolfootnotetext{<series>}` creates `\@twocolfootnotetext{<series>}`

```

10142 \newcommand{\m@make@twocolfootnotetext}[1]{%
10143 \@namelongdef{\@twocolfootnotetext#1}##1{%
10144 \insert\@nameuse{footins#1}{%

```

```

10145 \def\baselinestretch{\m@msinglespace}% <- v1.61803 addition
10146 \reset@font\@nameuse{foottextfont#1}%
10147 \@preamfntext
10148 \protected@edef\@currentlabel{%
10149 \csname p@footnote#1\endcsname \@nameuse{@thefnmark#1}}%
10150 \color@begingroup
10151 \@nameuse{@twocolfootfmt#1}{##1}%
10152 \color@endgroup}%
10153 \m@mmf@prepare}}
10154

\m@make@mptwocolfootnotetext \m@make@mptwocolfootnotetext{<series>} creates
\@mptwocolfootnotetext{<series>}
10155 \newcommand{\m@make@mptwocolfootnotetext}[1]{%
10156 \@namelongdef{\@mptwocolfootnotetext#1}##1{%
10157 \global\setbox\@nameuse{\@mpfootins#1}\vbox{%
10158 \unvbox\@nameuse{\@mpfootins#1}
10159 \def\baselinestretch{\m@msinglespace}% <- v1.61803 addition
10160 \reset@font\@nameuse{foottextfont#1}%
10161 \hsize\columnwidth \@parboxrestore
10162 \protected@edef\@currentlabel{%
10163 \csname p@mpfootnote#1\endcsname\@nameuse{@thefnmark#1}}%
10164 \color@begingroup
10165 \@nameuse{@twocolfootfmt#1}{##1}%
10166 \color@endgroup}\m@mmf@prepare}}
10167

\m@make@twocolfootfmt \m@make@twocolfootfmt{<series>} creates \@twocolfootfmt{<series>}
10168 \newcommand{\m@make@twocolfootfmt}[1]{%
10169 \@namedef{\@twocolfootfmt#1}##1{%
10170 \@preamtwofmt
10171 {\@nameuse{footfootmark#1}\strut
10172 {\@nameuse{foottextfont#1}##1}\strut\par}\allowbreak}}
10173

\m@make@twocol@footgroup \m@make@twocol@footgroup{<series>} creates \@twocol@footgroup{<series>}
10174 \newcommand{\m@make@twocol@footgroup}[1]{%
10175 \@namedef{\@twocol@footgroup#1}{%
10176 \@nameuse{foottextfont#1} \splittopskip=\ht\strutbox
10177 \m@mrigidbalance{\@nameuse{footins#1}}{\tw@}{\splittopskip}}}%
10178

\m@make@mptwocol@footgroup \m@make@mptwocol@footgroup{<series>} creates \@mptwocol@footgroup{<series>}
10179 \newcommand{\m@make@mptwocol@footgroup}[1]{%
10180 \@namedef{\@mptwocol@footgroup#1}{%
10181 \@nameuse{foottextfont#1} \splittopskip=\ht\strutbox
10182 \m@mrigidbalance{\@nameuse{\@mpfootins#1}}{\tw@}{\splittopskip}}}%
10183

```


27.6.2 Three column footnotes

`\threecolumnfootnotes` Declaration for three column footnotes.

```

10184 \newcommand{\threecolumnfootnotes}{%
10185   \@namedef{foottextfontv@r}{\foottextfont}% % [RS]
10186   \let\@footnotetext\@threecolfootnotetext
10187   \dimen\footinsv@r=3\footinsdim
10188   \count\footinsv@r=333\relax
10189   \m@make@threecol@footgroup{v@r}%
10190   \let\@footgroupv@r\@threecol@footgroupv@r
10191   \let\@mpfootnotetext\@mpthreecolfootnotetext
10192   \m@make@mpthreecol@footgroup{v@r}%
10193   \let\mp@footgroupv@r\@mpthreecol@footgroupv@r}
10194
```

`\@threecolfootnotetext` `\@threecolfootnotetext{<text>}` is the three column version of `\@footnotetext`

```

10195 \newcommand{\@threecolfootnotetext}[1]{\insert\footinsv@r{%
10196   \def\baselinestretch{\m@m@spacesinglespace}% <- v1.61803 addition
10197   \reset@font\foottextfont
10198   \@preamfntext
10199   \protected@edef\@currentlabel{%
10200     \csname p@footnote\endcsname\@thefnmark}%
10201   \color@begingroup
10202     \@threecolfootfmt{#1}%
10203   \color@endgroup}\m@mmf@prepare}
10204
```

`\@preamthreefmt` Give each column 0.3 of the text width.

```

10205 \newcommand{\@preamthreefmt}{%
10206   \hsize .3\hsize
10207   \parindent=\z@
10208   \tolerance=5000\relax
10209   \raggedright
10210   \leavevmode}
10211
```

`\@threecolfootfmt`

```

10212 \newcommand{\@threecolfootfmt}[1]{%
10213   \@preamthreefmt
10214   {\footfootmark\strut {\foottextfont #1}\strut\par}\allowbreak}
10215
```

`\@mpthreecolfootnotetext` `\@mpthreecolfootnotetext{<text>}` is the three column version of `\@mpfootnotetext`

```

10216 \newcommand{\@mpthreecolfootnotetext}[1]{%
10217   \global\setbox\@mpfootinsv@r\vbox{%
10218     \unvbox\@mpfootinsv@r
10219     \def\baselinestretch{\m@m@spacesinglespace}% <- v1.61803 addition
```

```

10220 \reset@font\foottextfont
10221 \hsize\columnwidth \@parboxrestore
10222 \protected@edef\@currentlabel{%
10223 \csname p@mpfootnote\endcsname\@thefnmark}%
10224 \color@begingroup
10225 \@threecolfootfmt{#1}%
10226 \color@endgroup}\m@mmf@prepare}
10227

```

`\threecolumnfootstyle` `\threecolumnfootstyle{<series>}` specifies three column footnote style for `<series>`

```

10228 \newcommand{\threecolumnfootstyle}[1]{%
10229 \m@make@threecolfootnotetext{#1}%
10230 \m@make@mpthreecolfootnotetext{#1}%
10231 \m@make@threecolfootfmt{#1}%
10232 \m@make@threecol@footgroup{#1}%
10233 \m@make@mpthreecol@footgroup{#1}%
10234 \m@make@footstart{#1}%
10235 \@namelongdef{\@footnotetext#1}##1{%
10236 \@nameuse{\@threecolfootnotetext#1}{##1}}%
10237 \@namelongdef{\@mpfootnotetext#1}##1{%
10238 \@nameuse{\@mpthreecolfootnotetext#1}{##1}}%
10239 \@namedef{\@footgroup#1}{\@nameuse{\@threecol@footgroup#1}}%
10240 \@namedef{\@mp@footgroup#1}{\@nameuse{\@mpthreecol@footgroup#1}}%
10241 \expandafter\dimen\csname footins#1\endcsname=3\footinsdim
10242 \expandafter\count\csname footins#1\endcsname=333\relax}
10243

```

`\m@make@threecolfootnotetext` `\m@make@threecolfootnotetext{<series>}` creates `\@threecolfootnotetext{<series>}`

```

10244 \newcommand{\m@make@threecolfootnotetext}[1]{%
10245 \@namelongdef{\@threecolfootnotetext#1}##1{%
10246 \insert\@nameuse{footins#1}{%
10247 \def\baselinestretch{\m@m@spacespace}% <- v1.61803 addition
10248 \reset@font\@nameuse{foottextfont#1}%
10249 \@preamfntext
10250 \protected@edef\@currentlabel{%
10251 \csname p@footnote#1\endcsname \@nameuse{\@thefnmark#1}}%
10252 \color@begingroup
10253 \@nameuse{\@threecolfootfmt#1}{##1}%
10254 \color@endgroup}\m@mmf@prepare}}
10255

```

`\m@make@mpthreecolfootnotetext` `\m@make@mpthreecolfootnotetext{<series>}` creates `\@mpthreecolfootnotetext{<series>}`

```

10256 \newcommand{\m@make@mpthreecolfootnotetext}[1]{%
10257 \@namelongdef{\@mpthreecolfootnotetext#1}##1{%
10258 \global\setbox\@nameuse{\@mpfootins#1}\vbox{%
10259 \unvbox\@nameuse{\@mpfootins#1}

```

```

10260 \def\baselinestretch{\m@msinglespace}% <- v1.61803 addition
10261 \reset@font\@nameuse{foottextfont#1}%
10262 \hsize\columnwidth \@parboxrestore
10263 \protected@edef\@currentlabel{%
10264 \csname p@mpfootnote#1\endcsname\@nameuse{@thefnmark#1}}%
10265 \color@begingroup
10266 \@nameuse{@threecolfootfmt#1}{##1}%
10267 \color@endgroup}\m@mmf@prepare}}
10268

```

`\m@make@threecolfootfmt` `\m@make@threecolfootfmt{<series>}` creates `\@threecolfootfmt{<series>}`

```

10269 \newcommand{\m@make@threecolfootfmt}[1]{%
10270 \@namelongdef{@threecolfootfmt#1}{##1}%
10271 \@preamthreefmt
10272 {\@nameuse{footfootmark#1}\strut
10273 {\@nameuse{foottextfont#1}{##1}\strut\par}\allowbreak}}
10274

```

`\m@make@threecol@footgroup` `\m@make@threecol@footgroup{<series>}` creates `\@threecol@footgroup{<series>}`

```

10275 \newcommand{\m@make@threecol@footgroup}[1]{%
10276 \@namedef{@threecol@footgroup#1}{%
10277 \@nameuse{foottextfont#1} \splittopskip=\ht\strutbox
10278 \m@mrigrbalance{\@nameuse{footins#1}}{\thr@@}{\splittopskip}}}}
10279

```

`\make@mpthreecol@footgroup` `\m@make@mpthreecol@footgroup{<series>}` creates

```

\@mpthreecol@footgroup{<series>
10280 \newcommand{\m@make@mpthreecol@footgroup}[1]{%
10281 \@namedef{@mpthreecol@footgroup#1}{%
10282 \@nameuse{foottextfont#1} \splittopskip=\ht\strutbox
10283 \m@mrigrbalance{\@nameuse{mpfootins#1}}{\thr@@}{\splittopskip}}}}
10284

```

27.6.3 Paraphrased footnotes

Paraphrased footnotes are based on *The T_EXbook*, page 398ff, in the Dirty Tricks appendix. It does a lot of box manipulations.

`\m@munvxh` `\m@munvxh{<vbox>}`: unvbox, extract the last line, and unhbox it.

```

10285 \newcommand{\m@munvxh}[1]{%
10286 \setbox0=\vbox{\unvbox#1%
10287 \global\setbox1=\lastbox}%
10288 \unhbox1
10289 \unskip
10290 \unskip
10291 \unpenalty
10292 \hskip\m@mipn@skip}
10293

```

`\m@mungebox`

```
10294 \newcommand{\m@mungebox}{%
10295   \setbox0=\hbox{\m@munvxh0}%
10296   \dp0=\z@
10297   \ht0=\footfudgefactor\wd0
10298   \box0
10299   \penalty0}
10300
```

`\m@mipn@skip` These are ‘inter-para-note-skip’ and ‘inter-parafootnote’ glue, for paragraphed
`\m@minterparanoteglue` footnotes.

```
10301 \newskip\m@mipn@skip
10302 \newcommand*{\m@minterparanoteglue}[1]{%
10303   {\foottextfont\global\m@mipn@skip=#1\relax}}
10304 \m@minterparanoteglue{1em plus.4em minus.4em}
10305
```

`\m@mmakehboxofhboxes` Make an hbox of hboxes.

```
10306 \newcommand*{\m@mmakehboxofhboxes}{\setbox0=\hbox{%
10307   \loop
10308     \unpenalty
10309     \setbox2=\lastbox
10310     \ifhbox2
10311       \setbox0=\hbox{\box2\unhbox0}
10312     \repeat}
10313
```

`\m@mremovehboxes`

```
10314 \newcommand*{\m@mremovehboxes}{\setbox0=\lastbox
10315   \ifhbox0{\m@mremovehboxes}\unhbox0 \fi}
10316
```

`\footfudgefiddle` TeX uses `\footfudgefactor` to estimate the space required for paragraphed footnotes. If it underestimates then the notes approach, or cover, the footer.
`\footfudgefiddle` can be changed (upward) from its default to improve matters.

```
10317 \newcommand*{\footfudgefiddle}{64}
10318
```

`\paragraphfootnotes` Declaration for paragraphed footnotes.

```
10319 \newcommand{\paragraphfootnotes}{%
10320   \@namedef{foottextfontv@r}{\foottextfont}% % [RS]
10321   \let\@footnotetext\@parafootnotetext
10322   \dimen\footinsv@r=\footinsdim
10323   \count\footinsv@r=1000\relax
10324   \m@make@para@footgroup{v@r}%
10325   \let\@footgroupv@r\@para@footgroupv@r
10326   \let\@mpfootnotetext\@mpparafootnotetext
10327   \m@make@mppara@footgroup{v@r}%
10328   \let\mp@footgroupv@r\@mppara@footgroupv@r
```

```

10329 {\foottextfont
10330 \dimen0=\baselineskip
10331 \multiply\dimen0 by 1024
10332 \divide\dimen0 by \hsize \multiply\dimen0 by \footfudgefiddle
10333 \xdef\footfudgefactor{\expandafter\strip@pt\dimen0 }}
10334

```

`\@parafootnotetext` `\@parafootnotetext{<text>}` is a paragraph version of `\@footnotetext`

```

10335 \newcommand{\@parafootnotetext}[1]{\insert\footinsv@r{
10336 \def\baselinestretch{\m@m@spacesinglespace}% <- v1.61803 addition
10337 \reset@font\foottextfont
10338 \@preamfntext
10339 \protected@edef\@currentlabel{%
10340 \csname p@footnote\endcsname\@thefnmark}%
10341 \setbox0=\vbox{\hsize=\maxdimen
10342 \color@begingroup
10343 \noindent \@parafootfmt{#1}%
10344 \color@endgroup}%
10345 \m@mungebox}\m@mmf@prepare}
10346

```

`\@parafootfmt` `\@parafootfmt{<text>}` is a paragraph version of `\@footfmt`

```

10347 \newcommand{\@parafootfmt}[1]{%
10348 \parindent=\z@
10349 \parfillskip=0pt \@plus 1fil
10350 {\footfootmark\strut {\foottextfont #1}\penalty-10}}
10351

```

`\@mpparafootnotetext` `\@mpparafootnotetext{<text>}` is a paragraph version of `\@mpfootnotetext`

```

10352 \newcommand{\@mpparafootnotetext}[1]{%
10353 \global\setbox\@mpfootinsv@r\vbox{%
10354 \unvbox\@mpfootinsv@r
10355 \def\baselinestretch{\m@m@spacesinglespace}% <- v1.61803 addition
10356 \reset@font\foottextfont
10357 \hsize\columnwidth \@parboxrestore
10358 \protected@edef\@currentlabel{%
10359 \csname p@mpfootnote\endcsname\@thefnmark}%
10360 \setbox0=\vbox{\hsize=\maxdimen
10361 \color@begingroup
10362 \noindent \@parafootfmt{#1}%
10363 \color@endgroup}%
10364 \m@mungebox}\m@mmf@prepare}
10365

```

`\paragraphfootstyle` `\paragraphfootstyle{<series>}` specifies paragraphed footnote style for `<series>`.

```

10366 \newcommand{\paragraphfootstyle}[1]{%
10367 \m@make@parafootnotetext{#1}%
10368 \m@make@mpparafootnotetext{#1}%

```

```

10369 \m@make@parafootfmt{#1}%
10370 \m@make@para@footgroup{#1}%
10371 \m@make@mppara@footgroup{#1}%
10372 \m@make@para@footstart{#1}%
10373 \@namelongdef{footnotetext#1}##1{%
10374   \@nameuse{parafootnotetext#1}{##1}}%
10375 \@namelongdef{mpfootnotetext#1}##1{%
10376   \@nameuse{mpparafootnotetext#1}{##1}}%
10377 \@namedef{footgroup#1}{\@nameuse{para@footgroup#1}}%
10378 \@namedef{mp@footgroup#1}{\@nameuse{mppara@footgroup#1}}%
10379 \@namedef{footstart#1}{\@nameuse{para@footstart#1}}%
10380 \expandafter\dimen\csname footins#1\endcsname=\footinsdim
10381 \expandafter\count\csname footins#1\endcsname=1000\relax
10382 {\@nameuse{foottextfont#1}}%
10383 \dimen0=\baselineskip
10384 \multiply\dimen0 by 1024
10385 \divide\dimen0 by \hsize \multiply\dimen0 by 64
10386 \xdef\footfudgefactor{\expandafter\strip@pt\dimen0 }}
10387

```

\m@make@parafootnotetext \m@make@parafootnotetext{*series*} creates \@parafootnotetext(*series*)

```

10388 \newcommand{\m@make@parafootnotetext}[1]{%
10389 \@namelongdef{parafootnotetext#1}##1{%
10390   \insert\@nameuse{footins#1}{
10391     \def\baselinestretch{\m@m@spacespace}%    <- v1.61803 addition
10392     \reset@font\@nameuse{foottextfont#1}}%
10393   \@preamfntext
10394   \protected@edef\@currentlabel{%
10395     \csname p@footnote#1\endcsname \@nameuse{@thefnmark#1}}%
10396   \setbox0=\vbox{\hsize=\maxdimen
10397     \color@begingroup
10398       \noindent \@nameuse{parafootfmt#1}{##1}%
10399     \color@endgroup}%
10400   \m@mungebox}\m@mmf@prepare}}
10401

```

\m@make@mpparafootnotetext \m@make@mpparafootnotetext{*series*} creates \@mpparafootnotetext(*series*)

```

10402 \newcommand{\m@make@mpparafootnotetext}[1]{%
10403 \@namelongdef{mpparafootnotetext#1}##1{%
10404   \global\setbox\@nameuse{mpfootins#1}\vbox{%
10405     \unvbox\@nameuse{mpfootins#1}
10406     \def\baselinestretch{\m@m@spacespace}%    <- v1.61803 addition
10407     \reset@font\@nameuse{foottextfont#1}}%
10408     \hsize\columnwidth \@parboxrestore
10409     \protected@edef\@currentlabel{%
10410       \csname p@mpfootnote#1\endcsname\@nameuse{@thefnmark#1}}%
10411     \setbox0=\vbox{\hsize=\maxdimen
10412       \color@begingroup
10413         \noindent \@nameuse{parafootfmt#1}{##1}%
10414       \color@endgroup}%

```

```

10415 \m@ungebox}\m@mmf@prepare}}
10416

```

```

\m@make@parafootfmt \m@make@parafootfmt{\series} creates \@parafootfmt\series
10417 \newcommand{\m@make@parafootfmt}[1]{%
10418 \@namelongdef{\parafootfmt#1}##1{%
10419 \parindent=\z@
10420 \parfillskip=0pt \@plus 1fil
10421 {\@nameuse{footfootmark#1}\strut
10422 {\@nameuse{foottextfont#1}##1}\penalty-10}}
10423

```

```

\m@make@para@footgroup {\series} creates \@para@footgroup\series
10424 \newcommand{\m@make@para@footgroup}[1]{%
10425 \@namedef{\para@footgroup#1}{%
10426 \unvbox\@nameuse{footins#1}
10427 \m@mmakehboxofhboxes
10428 \setbox0=\hbox{\unhbox0 \m@mremovehboxes}%
10429 \@nameuse{foottextfont#1}%
10430 \noindent\unhbox0\par}}
10431

```

```

\m@make@mppara@footgroup \m@make@mppara@footgroup{\series} creates \@mppara@footgroup\series
10432 \newcommand{\m@make@mppara@footgroup}[1]{%
10433 \@namedef{\mppara@footgroup#1}{%
10434 \unvbox\@nameuse{\mpfootins#1}
10435 \m@mmakehboxofhboxes
10436 \setbox0=\hbox{\unhbox0 \m@mremovehboxes}%
10437 \@nameuse{foottextfont#1}%
10438 \noindent\unhbox0\par}}
10439

```

```

\m@make@para@footstart \m@make@para@footstart{\series} creates \@para@footstart\series
10440 \newcommand{\m@make@para@footstart}[1]{%
10441 \@namedef{\para@footstart#1}{%
10442 \vskip\bigskipamount
10443 \leftskip=\z@
10444 \rightskip=\z@
10445 \parindent=\z@
10446 \vskip\skip\@nameuse{footins#1}%
10447 \footnoterule}}
10448

```

27.7 Nasty insert bits

\sidebar is implemented as a new kind of \insert and the extended footnotes are also implemented via new kinds of \inserts. New \inserts mean that the kernel's \makecol must be revised to cater for them. As this is part of the output process it is nasty to do.

Before v1.61803 \@makecol got (re)defined at several points in the code. Here is what I hope is the final version of \@makecol and associated code after having resolved and combined all the changes.

```

\if@mem@nofoot These are from [RS] as \@docclearpage should check for more than just
\@mem@testifnofoot \footins being void. Note that this covers all current class defined inserts,
\@mem@extranofoot including sidebars.

10449 \newif\if@mem@nofoot
10450 \newcommand*{\@mem@testifnofoot}{%
10451   \@mem@nofoottrue
10452   \ifvoid\footins\else\@mem@nofootfalse\fi
10453   \ifvoid\footinsv\else\@mem@nofootfalse\fi
10454   \ifvoid\sideins\else\@mem@nofootfalse\fi
10455   \@mem@extranofoot}
10456 \newcommand*{\@mem@extranofoot}{%
10457

\memold@docclearpage I thought that I could get away with using the kernel's \@docclearpage but [RS]
\mem@docclearpage discovered that I couldn't.
\@docclearpage10458 \let\memold@docclearpage\@docclearpage
10459 \newcommand{\mem@docclearpage}{%
10460   \@mem@testifnofoot
10461   \if@mem@nofoot
10462     \setbox\@tempboxa\vsplit\@cclv to\z@ \unvbox\@tempboxa
10463     \setbox\@tempboxa\box\@cclv
10464     \xdef\@deferlist{\@toplist\@botlist\@deferlist}%
10465     \global\let\@toplist\@empty
10466     \global\let\@botlist\@empty
10467     \global\@colroom\@colht
10468     \ifx \@currlist\@empty
10469       \else
10470         \@latexerr{Float(s) lost}\@ehb
10471         \global\let\@currlist\@empty
10472       \fi
10473     \@makefcolumn\@deferlist
10474     \@whiles\if@fcolmade \fi{\@opcol\@makefcolumn\@deferlist}%
10475     \if@twocolumn
10476       \if@firstcolumn
10477         \xdef\@dbldeferlist{\@dbltoplist\@dbldeferlist}%
10478         \global\let\@dbltoplist\@empty
10479         \global\@colht\textheight
10480         \begingroup
10481           \@dblfloatplacement
10482           \@makefcolumn\@dbldeferlist
10483           \@whiles\if@fcolmade \fi{\@outputpage
10484             \@makefcolumn\@dbldeferlist}%
10485         \endgroup
10486       \else
10487         \vbox{}\clearpage

```



```

10488     \fi
10489     \fi
10490   \else
10491     \setbox\@cclv\vbox{\box\@cclv\vfil}%
10492     \@makecol\@opcol
10493     \clearpage
10494   \fi}

```

Replace the kernel's `\@doclearpage`.

```

10495 \gdef\@doclearpage{\mem@doclearpage}
10496

```

`\m@m@makecolfloats` These two macros contain code common to versions of `\@makecol`

```

\m@m@makecoltext 10497 \newcommand*{\m@m@makecolfloats}{%
10498   \xdef\@freelist{\@freelist\@midlist}%
10499   \global\let\@midlist\@empty
10500   \@combinefloats}
10501 \newcommand*{\m@m@makecoltext}{%
10502   \ifvbox\@kludgeins
10503     \@makespecialcolbox
10504   \else
10505     \setbox\@outputbox \vbox to\@colht{%
10506       \@texttop
10507       \dimen@ \dp\@outputbox
10508       \unvbox \@outputbox
10509       \vskip -\dimen@
10510       \@textbottom}%
10511   \fi}
10512

```

`\m@m@makecolintro` A hook into the revised `\@makecol`

```

10513 \newcommand*{\m@m@makecolintro}{%
10514

```

`\m@mopfootnote` (footnote) code for possible use in `\@makecol`.

```

10515 \newcommand*{\m@mopfootnote}{\setbox\@outputbox \vbox{%
10516   \boxmaxdepth\@maxdepth
10517   \@tempdima\dp\@cclv
10518   \unvbox\@cclv
10519   \vskip-\@tempdima
10520   \vskip \skip\footins
10521   \color@begingroup
10522     \normalcolor
10523     \footnoterule
10524     \unvbox \footins
10525   \color@endgroup}}
10526

```

`\m@mopfootnotebf` (footnote) code in support of footnotes below floats. Problem with original code noted by Jørgen Larsen (jl@ruc.dk) on 2008/05/24.

```

10527 \newcommand*{\m@mopfootnotebf}{%
10528   \setbox\@outputbox \vbox{%
10529     \boxmaxdepth\@maxdepth
10530     \unvbox\@outputbox
10531     \vskip\skip\footins
10532     \color@begingroup
10533       \normalcolor
10534       \footnoterule
10535       \unvbox \footins
10536     \color@endgroup}}
10537

```

`\m@mopsidebar` (sidebar) code for possible use in `\@makecol`. From DA's latest sidebar fixes.

```

10538 \newcommand*{\m@mopsidebar}{%
10539   \ifvoid\sideins\else
10540     \setbox\@outputbox \vbox{%
10541       \sidecontents
10542       \unvbox\@outputbox}
10543   \fi}
10544

```

`\mem@makecol` DA's latest version of `\@makecol` (giving the standard footnote order (bottom floats after footnotes)) putting the sidebar insert after the others (from mempatch v4.9).

```

10545 \gdef\mem@makecol{%
10546   \m@m@makecolintro
10547   \ifvoid\footins
10548     \setbox\@outputbox \box\@cclv
10549   \else
10550     \m@mopfootnote
10551   \fi
10552   \m@mdoextrafeet
10553   \m@m@makecolfloats
10554   \m@mopsidebar
10555   \m@m@makecoltext
10556   \global \maxdepth \@maxdepth}
10557

```

`\mem@makecolbf` A version of `\@makecol` which puts footnotes at the bottom of the page (after any bottom floats).

```

10558 \gdef\mem@makecolbf{%
10559   \m@m@makecolintro
10560   \setbox\@outputbox \box\@cclv
10561   \m@m@makecolfloats
10562   \ifvoid\footins\else
10563     \m@mopfootnotebf
10564   \fi
10565   \m@mdoextrafeet
10566   \m@mopsidebar

```

```

10567 \m@m@makecoltext
10568 \global\maxdepth \@maxdepth}
10569

```

`\mem@makecoldbl` A version of `\@makecol` which is a placeholder to fix the doublefloat problem.

```

10570 \gdef\mem@makecoldbl{%
10571 \m@m@makecolintro
10572 \setbox\@outputbox \box\@cclv
10573 \m@m@makecolfloats
10574 \m@mopsidebar% <- added
10575 \ifvoid\footins
10576 \else
10577 \m@mopfootnote
10578 \fi
10579 \m@m@doextrafeet
10580 \m@m@makecoltext
10581 \global \maxdepth \@maxdepth}
10582

```

`\feetabovelfloat` declarations to put footnotes above bottom floats (standard L^AT_EX) or at the

`\feetbelowfloat` bottom of the page

```

\@makecol{
10583 \newcommand{\feetabovelfloat}{\gdef\@makecol{\mem@makecol}}
10584 \newcommand{\feetbelowfloat}{\gdef\@makecol{\mem@makecolbf}}
10585 \feetabovelfloat
10586

```

`\@reinserts` DA's final version from last patch.

```

10587 \gdef\@reinserts{%
10588 \ifvoid\footins\else\insert\footins{\unvbox\footins}\fi
10589 \m@m@do@reinsert@extrafeet
10590 \ifvbox\@kludgeins\insert\@kludgeins{\unvbox\@kludgeins}\fi
10591 \ifvoid\sideins\else\insert\sideins{\unvbox\sideins}\fi}
10592

```

27.8 Side footnotes

27.8.1 Extension to the regular footnote

The code here enables the regular `\footnote` to be set in the margin. It is based on Robin Fairbairns' `footmisc` package.

`\ifm@mfnmargin` Declarations for regular footnotes to be set as normal at the foot of the page, or
`\footnotesatfoot` in the margin.

```

\footnotesinmargin
10593 \newif\ifm@mfnmargin
10594 \newcommand*{\footnotesatfoot}{\m@mfnmarginfalse}
10595 \newcommand*{\footnotesinmargin}{\m@mfnmargintrue}
10596 \footnotesatfoot
10597

```

`\@footnotetext` Change `\@footnotetext` to enable footnotes to be put into the margin by using `\marginpar`.

```
10598 \renewcommand{\@footnotetext}[1]{%
10599   \ifm@mfnmargin%   use marginpar
      Use \marginpar for placing the footnote.
10600   \marginpar{%
10601     \def\baselinestretch{\m@m@singlespace}%
10602     \reset@font
10603     \foottextfont
10604     \protected@edef\@currentlabel{%
10605       \csname p@footnote\endcsname\@thefnmark}%
10606     \color@begingroup
10607       \@makefntext{\ignorespaces#1}%
10608     \color@endgroup}%
10609   \else% regular feet
```

This is the original code for `\@footnotetext`.

```
10610   \insert\footins{%
10611     \def\baselinestretch{\m@m@singlespace}%
10612     \reset@font
10613     \foottextfont
10614     \@preamfntext
10615     \hsize\columnwidth
10616     \protected@edef\@currentlabel{%
10617       \csname p@footnote\endcsname\@thefnmark}%
10618     \color@begingroup
10619       \@makefntext{%
10620         \rule{\z@\footnotesep}{\ignorespaces{\foottextfont #1}%
10621         \@finalstrut\strutbox}%
10622     \color@endgroup}%
10623   \fi%
10624   \m@m@mmf@prepare}
10625
```

27.9 Bottom aligned side footnotes

The code for the bottom aligned marginal footnotes is based partly on the `\sidebar` code and code from David Salomon, 'Output Routines: Examples and Techniques. Part III: Insertions', *TUGboat*, 11, 4, pp 588–605, Nov 1990.

`\sidefootmargin` `\sidefootmargin` is the user command for setting the side for side footnotes
`\m@m@sidefoot@margin` (stored as `\m@m@sidefoot@margin`). The default is outer.

```
10626 \newcommand*{\sidefootmargin}[1]{%
10627   \m@msetm@argin{#1}%
10628   \ifnum\m@m@mm@argin<\z@
10629     \@memwarn{Bad \string\sidefootmargin\space argument '#1'\MessageBreak
10630       set to 'outer'}%
10631     \gdef\m@m@sidefoot@margin{2}%   set as outer
10632   \else
```

```

10633 \global\let\m@msidefoot@margin\m@mm@argin
10634 \fi}
10635 \sidefootmargin{outer}
10636

```

`\sidefootins` As for any non-main text matter we need an insert.

```

10637 \newinsert\sidefootins
10638 \skip\sidefootins=0pt
10639 \count\sidefootins=0\relax
10640

```

`\sidefootadjust` The length `\sidefootadjust` can be used for fine control over the vertical position of the base of the column of side footnotes. The length `\sidefootheight` is the height of the column and is best set by the `\setsidefootheight` macro, which takes the desired height as its argument and sets the height of the `\sidefootins \insert`. The height is initially set to the `\textheight`.

```

10641 \newlength{\sidefootadjust}
10642 \setlength{\sidefootadjust}{0pt}
10643 \newlength{\sidefootheight}
10644 \newcommand*{\setsidefootheight}[1]{%
10645   \setlength{\dimen\sidefootins}{#1}%
10646   \advance\dimen\sidefootins -\topskip
10647   \advance\dimen\sidefootins \ht\strutbox
10648   \setlength{\sidefootheight}{\dimen\sidefootins}}
10649 \setsidefootheight{\textheight}
10650

```

`\sidefoothsep` The horizontal space between the text block and side footnotes
`\sidefootwidth` (`\sidefoothsep`), the width of the column of notes (`\sidefootwidth`), and the
`\sidefootvsep` vertical space between consecutive side footnotes (`\sidefootvsep`).

```

10651 \newlength{\sidefoothsep}
10652 \newlength{\sidefootvsep}
10653 \newlength{\sidefootwidth}
10654

```

`\setsidefeet` The macro `\setsidefeet{<hsep>}{<width>}{<vsep>}{<adj>}{}{<height>}` sets the specifications for the side footnotes. An ‘*’ means ‘use the current value’.

```

10655 \newcommand*{\setsidefeet}[6]{%
10656   \nametest{#1}{*}\ifsamename\else
10657     \setlength{\sidefoothsep}{#1}\@memznegtest{\sidefoothsep}%
10658   \fi
10659   \nametest{#2}{*}\ifsamename\else
10660     \setlength{\sidefootwidth}{#2}\@memznegtest{\sidefootwidth}%
10661   \fi
10662   \nametest{#3}{*}\ifsamename\else
10663     \setlength{\sidefootvsep}{#3}\@memznegtest{\sidefootvsep}%
10664   \fi

```

```

10665 \nametest{#4}{*}\ifsamename\else
10666 \setlength{\sidefootadjust}{#4}%
10667 \fi
10668 \nametest{#5}{*}\ifsamename\else
10669 \def\sidefoottextfont{#5}%
10670 \fi
10671 \nametest{#6}{*}\ifsamename\else
10672 \setsidefootheight{#6}%
10673 \ifdim\dimen\sidefootins>\z@ \else
10674 \@@memerror{\protect\sidefootheight\space is zero or negative}{\@ehd}%
10675 \fi
10676 \fi}
10677 \setsidefeet{\marginparsep}{\marginparwidth}%
10678 {\onelineskip}{0pt}%
10679 {\normalfont\footnotesize}{\textheight}%
10680

```

`\sidefootform` Set the sidefootnotes raggedy right

```

10681 \newcommand*{\sidefootform}{\rightskip=\z@ \@plus 2em}
10682

```

`\m@sideft@left` Macros placing the sidefootnotes at the left and the right respectively.

```

\m@sideft@right 10683 \newcommand*{\m@sideft@left}{%
10684 \@tempdimc \sidefootwidth
10685 \advance\@tempdimc\sidefoothsep
10686 \kern-\@tempdimc}
10687 \newcommand*{\m@sideft@right}{%
10688 \@tempdimc \columnwidth% or \hsize
10689 \advance\@tempdimc\sidefoothsep
10690 \kern\@tempdimc}
10691

```

`\m@mdownsf` A length used in the vertical positioning of sidefootnotes. (Perhaps one of the `\@tempdim` lengths could be used instead?)

```

10692 \newlength{\m@mdownsf}
10693

```

`\sidefootcontents` The essence of the sidefootnote task. This first positions the column to the left or the right and then tries to make the bottom of the column align with the bottom of the textblock.

```

10694 \newcommand*{\sidefootcontents}{\hbox to \z@{%
10695 \m@mwhich@margin{\m@msidefoot@margin}%
10696 \ifmemtortm
10697 \m@sideft@right
10698 \else
10699 \m@sideft@left
10700 \fi

```

Now the fun part. The general idea is to measure the height of the insert's contents, subtract this from the specified height of the insert, and move the

contents down by that amount. The code below seems to work except when sidefootnotes spill over to the following page; also, like `\sidebar`, they are out of vertical alignment when on a `\chapter` page. I haven't found a way to automatically adjust for these, which is why the `\sidefootadjust` length is there to enable manual adjustment.

```

10701 \vtop to 0pt{%      original
10702   \normalsize\normalfont\sidefoottextfont
10703   \vskip\topskip \vskip-\ht\strutbox
10704   \vskip\sidefootadjust%      use this for minor vertical adjustment
10705   \m@mdownsf=\dimen\sidefootins
10706   \advance\m@mdownsf-\ht\sidefootins
10707   \advance\m@mdownsf-\dp\sidefootins

```

Here's where the fiddling occurs, arrived at by a mixture of theory and experiment.

```

10708   \ifdim\m@mdownsf>\sidefootvsep
10709     \advance\m@mdownsf\sidefootvsep
10710     \advance\m@mdownsf 0.5\ht\strutbox
10711   \fi
10712   \vskip\m@mdownsf%      --- basically works
10713   \unvbox\sidefootins%
10714   \vss}%
10715 \hss}}
10716

```

`\m@mopsidefoot` If there are any sidefeet then add them to the output.

```

10717 \newcommand*{\m@mopsidefoot}{%
10718   \ifvoid\sidefootins\else
10719     \setbox\@outputbox \vbox{%
10720       \sidefootcontents
10721       \unvbox\@outputbox}
10722   \fi}
10723

```

`\mem@makecol` Revise these to cater for the new sidefoot insert.

```

\mem@makecolbf10724 \gdef\mem@makecol{
  \reinserts10725 \m@m@makecolintro
\@mem@extranofeet10726 \ifvoid\footins
\mem@testifnofoot10727   \setbox\@outputbox \box\@cclv
10728   \else
10729     \m@mopfootnote
10730   \fi
10731   \m@m@doextrafeet
10732   \m@m@makecolfloats
10733   \m@mopsidebar
10734   \m@mopsidefoot
10735   \m@m@makecoltext
10736   \global \maxdepth \@maxdepth}
10737

```

```

10738 \gdef\mem@makecolbf{
10739   \m@m@makecolintro
10740   \setbox\@outputbox \box\@cclv
10741   \m@m@makecolfloats
10742   \ifvoid\footins
10743   \else
10744     \m@mopfootnotebf
10745   \fi
10746   \m@m@doextrafeet
10747   \m@m@opsidebar
10748   \m@m@opsidefoot
10749   \m@m@makecoltext
10750   \global \maxdepth \@maxdepth}
10751
10752 \gdef\@reinserts{%
10753   \ifvoid\footins\else\insert\footins{\unvbox\footins}\fi
10754   \m@m@do@re@in@extrafeet
10755   \ifvbox\@kludgeins\else\insert\@kludgeins{\unvbox\@kludgeins}\fi
10756   \ifvoid\sideins\else\insert\sideins{\unvbox\sideins}\fi
10757   \ifvoid\sidefootins\else\insert\sidefootins{\unvbox\sidefootins}\fi}
10758

```

The easy way of extending the `\clearpage` code is by:

```

\renewcommand*{\@mem@extranofeet}{%
  \ifvoid\sidefootins\else\@mem@nofootfalse\fi}

```

but as so many other changes have to be made it's probably better to leave that alone and change `\@mem@testifnofoot` directly.

```

10759 \renewcommand*{\@mem@testifnofoot}{%
10760   \@mem@nofoottrue
10761   \ifvoid\footins\else\@mem@nofootfalse\fi
10762   \ifvoid\footinsv@r\else\@mem@nofootfalse\fi
10763   \ifvoid\sideins\else\@mem@nofootfalse\fi
10764   \ifvoid\sidefootins\else\@mem@nofootfalse\fi
10765   \@mem@extranofeet}
10766

```

27.9.1 The sidefootnote macros

The macros here are essentially a copy of the ones for `\footnote`. I have not done anything about sidefootnotes in minipages, as I can't imagine a use for them while simultaneously making the final appearance looking good.

The usual counter.

```

10767 \newcounter{sidefootnote}
10768 \renewcommand{\thesidefootnote}{\@arabic\c@sidefootnote}
10769 \@addtoreset{sidefootnote}{chapter}
10770

```


By using:

```
\letcountercounter{sidefootnote}{footnote}
the same counter will be used for both footnotes and sidefootnotes. This can be
reverted to separate counters by;
\unletcounter{sidefootnote}
```

`\sidefoottextfont` The font for the sidefootnotes.

```
10771 %\newcommand*{\sidefoottextfont}{\normalfont\footnotesize}
10772
```

The remainder of the code is a copied from that for `\footnote` but with the appropriate variables for `\sidefootnotes`

```
\@sidempfn
\thesidempfn 10773 \newcommand*{\@sidempfn}{sidefootnote}
10774 \newcommand*{\thesidempfn}{\thesidefootnote}
10775
```

```
\@makesidefnmark
10776 \newcommand*{\@makesidefnmark}{\hbox{\@textsuperscript{%
10777   \normalfont\@thesidefnmark}}}
10778
```

```
\@preamsidefntext
10779 \newcommand{\@preamsidefntext}{%
10780   \interlinepenalty\interfootnotelinepenalty
10781   \floatingpenalty \@MM
10782   \splittopskip=\footnotesep
10783   \splitmaxdepth=\dp\strutbox
10784   \@parboxrestore}
10785
```

```
\@sidefootnotetext
10786 \newcommand{\@sidefootnotetext}[1]{\insert\sidefootins{%
10787   \hsize\sidefootwidth
10788   \@parboxrestore
10789   \def\baselinestretch{\m@m@single space}%
10790   \sidefootform \normalsize\normalfont\sidefoottextfont
10791   \splittopskip=\ht\strutbox
10792   \splitmaxdepth=\dp\strutbox
10793   \allowbreak
10794   \prevdepth=\dp\strutbox
10795   \vskip-\parskip
10796   \protected@edef\@currentlabel{%
10797     \csname p@sidefootnote\endcsname\@thesidefnmark}%
10798   \color@begingroup
10799   \@makesidefntext{\sidefoottextfont #1}}%
10800 \color@endgroup
10801 \ifvmode\else
```

```

10802     \unskip\@finalstrut\strutbox
10803     \fi
10804     \par
10805     \ifdim\prevdepth>\dp\strutbox \prevdepth=\dp\strutbox\fi
10806     \ifdim\prevdepth>99\p@
10807         \nobreak
10808         \vskip-\prevdepth
10809         \allowbreak
10810         \vskip\dp\strutbox
10811     \fi
10812     \vskip\sidefootvsep}%
10813     \m@mmf@prepare}
10814
\@sidefootnotemark
10815 \newcommand*{\@sidefootnotemark}{%
10816     \leavevmode
10817     \ifhmode
10818         \edef\x@sf{\the\spacefactor}%
10819         \m@mmf@check
10820         \nobreak
10821     \fi
10822     \@makesidefnmark
10823     \m@mmf@prepare
10824     \ifhmode\spacefactor\x@sf\fi
10825     \relax}
10826
\sidefootnote
10827 \newcommand*{\sidefootnote}{\@ifnextchar[
10828     \@xsidefootnote{\stepcounter\@sidempfn
10829     \protected@xdef\@thesidefnmark{\thesidempfn}%
10830     \@sidefootnotemark\@sidefootnotetext}}

\@xsidefootnote
10831 \def\@xsidefootnote[#1]{%
10832     \begingroup
10833     \csname c@\@sidempfn\endcsname #1\relax
10834     \unrestored@protected@xdef\@thesidefnmark{\thesidempfn}%
10835     \endgroup
10836     \@sidefootnotemark\@sidefootnotetext}
10837
\sidefootnotemark
10838 \newcommand{\sidefootnotemark}{%
10839     \@ifnextchar[
10840         \@xsidefootnotemark
10841         {\stepcounter{sidefootnote}%
10842         \protected@xdef\@thesidefnmark{\thesidefootnote}%
10843         \@sidefootnotemark}}

```

```

\@xsidefootnotemark
10844 \def\@xsidefootnotemark[#1]{%
10845   \begingroup
10846   \c@sidefootnote #1\relax
10847   \unrestored@protected@xdef\@thesidefnmark{\thesidefootnote}%
10848   \endgroup
10849   \@sidefootnotemark}
10850

\sidefootnotetext
10851 \newcommand*\sidefootnotetext{%
10852   \@ifnextchar[
10853     \@xsidefootnotetext
10854     {\protected@xdef\@thesidefnmark{\thesidempfn}%
10855     \@sidefootnotetext}}

\@xsidefootnotetext
10856 \def\@xsidefootnotetext[#1]{%
10857   \begingroup
10858   \csname c@\@sidempfn\endcsname #1\relax
10859   \unrestored@protected@xdef\@thesidefnmark{\thesidempfn}%
10860   \endgroup
10861   \@sidefootnotetext}
10862

\sidefootmarkstyle
\sidefootscript10863 \newcommand*\sidefootmarkstyle[1]{\def\sidefootscript##1{#1}}
\makesidefootmarkhook10864 \newcommand*\makesidefootmarkhook{}
\sidefootfootmark10865 \newcommand*\sidefootfootmark{%
10866   \ifdim\sidefootmarkwidth < \z@
10867     \llap{\hb@xt@-\sidefootmarkwidth{%
10868       \hss\normalfont\sidefootscript{\@thesidefnmark}}}%
10869     \hspace*{-\sidefootmarkwidth}}%
10870   \else
10871     \ifdim\sidefootmarkwidth = \z@
10872       {\normalfont\sidefootscript{\@thesidefnmark}}%
10873     \else
10874       \hb@xt@\sidefootmarkwidth{\hss\normalfont\sidefootscript{\@thesidefnmark}}%
10875     \fi
10876   \fi}
10877

\makesidefootmark
10878 \newcommand*\makesidefootmark[1]{%
10879   \leavevmode
10880   \parindent \sidefootparindent\noindent
10881   \leftskip\sidefootmarksep\relax
10882   \advance\leftskip \sidefootmarkwidth \null\nobreak\hskip -\leftskip\relax
10883   \makesidefootmarkhook\relax

```

```

10884 \sidefootfootmark #1}
10885
\@makesidefntext
10886 \newcommand{\@makesidefntext}[1]{\makesidefootmark #1}
10887
Final layout.

\sidefootmarkwidth
\sidefootmarksep10888 \newlength{\sidefootmarkwidth}
\sidefootparindent10889 \setlength{\sidefootmarkwidth}{0em}
10890 \newlength{\sidefootmarksep}
10891 \setlength{\sidefootmarksep}{0em}
10892 \newlength{\sidefootparindent}
10893 \setlength{\sidefootparindent}{1em}
10894 \sidefootmarkstyle{\textsuperscript{#1}}
10895

```

27.10 End notes

This is from the pagenote package, with minor amendments.

```

\ifm@mpnpageopt We need two flags for the options. \m@mpnpageoptfalse means that page
\ifm@mpncontopt numbers are not available and \m@mpncontopt means that note numbers are
reset at each \chapter.
10896 \newif\ifm@mpnpageopt
10897 \m@mpnpageoptfalse
10898 \newif\ifm@mpncontopt
10899 \m@mpncontoptfalse

\c@pagenote We need a counter for the notes. This is the default definition.
\thepagenote10900 \newcounter{pagenote}[chapter]
10901 \renewcommand{\thepagenote}{\arabic{pagenote}}
10902 \setcounter{pagenote}{0}
10903

\notepageref Let the user change the default options. \notepageref makes page numbers
\continuousnotenums available (the package page option) and \continuousnotenums means that notes
will be numbered continuously throughout the document(the package continuous
option).
10904 \newcommand*{\notepageref}{\m@mpnpageopttrue}
10905 \@onlypreamble\notepageref
10906 \newcommand*{\continuousnotenums}{%
10907 \counterwithout{pagenote}{chapter}
10908 \renewcommand{\thepagenote}{\arabic{pagenote}}}
10909 \@onlypreamble\continuousnotenums
10910

```

`\ifmempagenotes` Need to check if notes are required.

```
10911 \newif\ifmempagenotes
10912 \mempagenotesfalse
10913
```

`\makepagenote` This sets up the note file. At the end it emasculates itself so it can only be used once.

```
10914 \newcommand*\makepagenote{%
10915   \newwrite\@notefile
10916   \immediate\openout\@notefile=\jobname.ent
10917   \mempagenotestru
```

`\pagenote` Make sure that this has a useful definition.

```
10918 \def\pagenote{\@bsphack\beginingroup
10919   \@sanitize
10920   \m@m@wrpnote}%

10921 \typeout{Writing note file \jobname.ent}%
10922 \let\makepagenote\@empty}
10923
```

`\immediate@protected@write` We might have to do some immediate writes. This is an immediate version of the kernel `\protected@write`.

```
10924 \newcommand*\immediate@protected@write}[3]{%
10925   \beginingroup
10926   #2%
10927   \let\protect\@unexpandable@protect
10928   \edef\reserved@a{\immediate\write#1{#3}}%
10929   \reserved@a
10930   \endgroup
10931   \if@nobreak\ifvmode\nobreak\fi\fi}
10932
```

`\m@m@pnwrite` If the `page` option is used we cannot use an immediate write because the page number is only known in the output routine.

```
10933 \let\m@m@pnwrite\immediate@protected@write
10934 \AtBeginDocument{%
10935   \if@mpnpagopt
10936     \let\m@m@pnwrite\protected@write
10937   \fi}
10938
```

`\pnchap` Redefine these for different subheadings in the notes list.

```
\pnschap10939 \newcommand*\pnchap{\f@rtoc}
10940 \newcommand*\pnschap{\f@rbdy}
10941
```

`\m@m@wrpnote` This writes the note information to the note file. If the optional argument is empty it increments the note counter and calls `\notenumintext` to handle its appearance in the body text.

```

10942 \newcommand{\m@m@wrpnote}[2][{}]{%
10943   \@ifmtarg{#1}{\refstepcounter{pagenote}%
10944     \notenumintext{\thepagenote}}{}}%
    Check if this is the first note in a division, and if so indicate this in the file.
10945   \ifm@mpn@new@chap
10946     \global\m@mpn@new@chapfalse
10947     \addtonotes{\string\pagenotesubhead{\@chapapp}{\thechapter}{\pnchap}}%
10948   \fi
10949   \ifm@mpn@new@schap
10950     \global\m@mpn@new@schapfalse
10951     \addtonotes{\string\pagenotesubhead{\@chapapp}{\pnschap}}%
10952   \fi
    Finally, write the entry.
10953   \m@m@pnwrite\@notefile{
10954     {\string\noteentry{\thepagenote}{#1}{#2}{\thepage}}%
10955   \endgroup
10956   \@esphack}
10957

```

`\pagenote` The user command to generate a note. It is given substance by `\makepagenote`.

```

10958 \def\pagenote{\@bsphack\begingroup \@sanitize\m@m@pagenote}

```

`\m@m@pagenote`

```

10959 \newcommand{\m@m@pagenote}[2][{}]{\endgroup\@esphack}
10960

```

`\pagetofootnote` Let the user change pagenotes to footnotes, or vice-versa. In either case the `\foottopagenote` optional argument is ignored.

```

\memsavefootnote 10961 \newcommand*\pagetofootnote{%
\memsavepagenote 10962   \let\memsavepagenote\pagenote
10963   \renewcommand{\pagenote}[2][{}]{\footnote{##2}}
10964 \newcommand*\foottopagenote{%
10965   \let\memsavefootnote\footnote
10966   \renewcommand*\footnote[2][{}]{\pagenote{##2}}
10967

```

`\addtonotes` `\addtonotes{<text>}` puts `<text>` into the notes file.

```

10968 \newcommand{\addtonotes}[1]{\ifmempagenotes
10969   \IfFileExists{\jobname.ent}{\m@m@pnwrite\@notefile{#1}{\mempnofilewarn}}%
10970 \fi}
10971

```

`\notenumintext` `\notenumintext{<notenum>}` typesets `<notenum>` (in the body text).

```

\notenuminnotes 10972 \newcommand{\notenumintext}[1]{%
10973   \textsuperscript{#1}}

```

`\notenuminnotes{<notenum>}` typesets `<notenum>` (as part of the note).

```
10974 \newcommand{\notenuminnotes}[1]{%
10975   {\normalfont #1.}\space}
```

`\noteentry` `\noteentry{<notenum>}{<id>}{<pagenum>}{<text>}` typesets a note.

```
10976 \newcommand{\noteentry}[4]{%
10977   \prenoteinnotes
10978   \noteidinnotes{#1}{#2}\pageinnotes{#4}\noteinnotes{#3}%
10979   \postnoteinnotes}
10980
```

`\idtextinnotes` `\idtextinnotes{<id text>}` typesets the note's `<id text>`.

```
10981 \newcommand{\idtextinnotes}[1]{%
10982   [#1]\space}
```

`\noteidinnotes` `\noteidinnotes{<notenum>}{<id>}` is used to typeset the note identification (in the note listing). It is set so that it typesets the `<id>` if it is not empty, otherwise it sets the `<notenum>`.

```
10983 \newcommand{\noteidinnotes}[2]{%
10984   \@ifmtarg{#2}{%
10985     \notenuminnotes{#1}}{\idtextinnotes{#2}}}
```

`\pageinnotes` `\pageinnotes{<pagenum>}` calls `\printpageinnotes{<pagenum>}` to typeset the originating page number (in the note), but only if the `\notepageref` declaration has been used (the page number is not trustworthy unless the `\notepageref` has been used).

```
10986 \newcommand{\pageinnotes}[1]{%
10987   \ifm@mpnpageopt \printpageinnotes{#1}\fi}
10988 \newcommand*{\printpageinnotes}[1]{%
10989   (\pagerefname\ #1)\space}
```

`\noteinnotes` `\noteinnotes{<text>}` is used to typeset the note's text (in the note list).

```
10990 \newcommand{\noteinnotes}[1]{#1}
10991
```

`\prenoteinnotes` These are called immediately before and after the note information is typeset.

```
\postnoteinnotes 10992 \newcommand{\prenoteinnotes}{\par\noindent}
10993 \newcommand{\postnoteinnotes}{\par}
10994
```

`\notesname` Heading for note list.

```
\notedivision 10995 \providecommand*{\notesname}{Notes}
10996 \newcommand*{\notedivision}{\chapter{\notesname}}
10997
```

`\printnotes` User commands to print the note file.

```
\printpagenotes* 10998 \newcommand*{\printpagenotes}{\@ifstar{\@sprintpagenotes}{\@printpagenotes}}
```

```

\mempnofilewarn Warning when the notes file does not exist.
10999 \newcommand*\mempnofilewarn}{%
11000 \ClassWarning{memoir}{There is no .ent file}}
11001

\@sprintpagenotes Macro implementing \printpagenotes*.
11002 \newcommand*\@sprintpagenotes}{%
11003 \ifmempagenotes
11004 \notedivision
11005 \IfFileExists{\jobname.ent}{%
11006 \immediate\closeout\@notefile
11007 \input{\jobname.ent}%
11008 \immediate\openout\@notefile=\jobname.ent%
11009 }{%
11010 \mempnofilewarn
11011 }%
11012 \fi}
11013

\@printpagenotes Macro implementing \printpagenotes.
11014 \newcommand*\@printpagenotes}{%
11015 \ifmempagenotes
11016 \notedivision
11017 \IfFileExists{\jobname.ent}{%
11018 \immediate\closeout\@notefile
11019 \input{\jobname.ent}%
11020 }{%
11021 \mempnofilewarn
11022 }
11023 \fi}
11024

\pagenotesubhead The section heading before each set of notes.
\pagenotesubhead{\chaptername}{\number}{\title}
11025 \newcommand*\pagenotesubhead[3]{%
11026 \section*{#1 #2 #3}}
11027

```

28 Change marks

When preparing a manuscript it normally goes through several iterations. The commands provided may be used to identify changes made to a document during its life cycle.

The code for this part of the class is based on the version controls in the iso class [Wil00b].

28.1 Print control

Members of the development group often need to see the changes between document versions, while the general public does not.

`\ifchangemarks` This controls the appearance of the version controls defined below.

```
11028 \newif\ifchangemarks\changemarksfalse
```

The marks only work properly when the `draft` option is in effect. Also, the command `\changemarkstrue` must be put in the document preamble.

`\changemarks` More user friendly version of `\changemarks(true/false)`.

```
\nochangemarks 11029 \newcommand*\changemarks{\changemarkstrue}
```

```
11030 \newcommand*\nochangemarks{\changemarksfalse}
```

```
11031
```

`\v@rid` This acts as an alias for `\marginpar` when both `\ifchangemarks` is true and the `draft` option is in effect, otherwise it throws away its two arguments.

```
11032 \newcommand{\v@rid}[2]{%
```

```
11033 \@bsphack
```

```
11034 \ifchangemarks
```

```
11035 \ifdraftdoc
```

```
11036 \marginpar[#1]{#2}%
```

```
11037 \fi\fi
```

```
11038 \@esphack}
```

```
11039
```

28.2 Change marking

The following commands flag changes in the typeset document. Each of the commands takes one parameter which is intended to be a ‘change number’ or comment for tracking purposes. A symbol and the *change-id* is put into the margin near where the command is given. The marking commands should be attached to some word or punctuation mark in the text otherwise extraneous spaces may creep into the final document.

`\added` `\added{<change-id>}` Flags, with the symbol \oplus , that something has been added to the manuscript.

```
11040 \newcommand{\added}[1]{%
```

```
11041 \@bsphack
```

```
11042 \ifchangemarks
```

```
11043 \v@rid{\small$\oplus$ #1}{\small$\oplus$ #1}%
```

```
11044 \fi
```

```
11045 \@esphack}
```

`\deleted` `\deleted{<change-id>}` Flags, with the symbol \neq , that something has been deleted from the manuscript.

```
11046 \newcommand{\deleted}[1]{%
```

```
11047 \@bsphack
```

```

11048 \ifchangemarks
11049   \v@rid{\small$\neq$ #1}{\small$\neq$ #1}%
11050 \fi
11051 \@esphack}

```

`\changed` `\changed{⟨change-id⟩}` Flags, with the symbol \Leftrightarrow , that something has been changed in the text.

```

11052 \newcommand{\changed}[1]{%
11053   \@bsphack
11054   \ifchangemarks
11055     \v@rid{\small$\Leftrightarrow$ #1}{\small$\Leftrightarrow$ #1}%
11056   \fi
11057   \@esphack}
11058

```

29 Trimming marks

The `showtrims` options prints trimming marks at the corners of the logical page. The code for this comes from ideas gleaned from Martin Schröder’s `everyshi` package [Sch98] and Melchior Franz’s `crop` package [Fra00]. The implementation and any errors are mine.

The implementation up to October 2002 was limited to putting a cross at the page corners. The manual directed users to the `crop` package if they wanted anything more. In October 2002 the implementation was substantially extended. The background to this is below.

Before the release of version 1.7 of `crop` in May 2002 its author asked me to make some changes to `memoir` that would be helpful to him, and I did so. Later he told me that the changes were unnecessary and I reverted to the original `memoir` code. On 2002/10/06 Peter Heslin (peter.heslin@ucd.ie) started a thread on `comp.text.tex` titled ‘Incompatibility of `memoir.cls` and `crop.sty`’ in which he said that `memoir` and `crop` did not seem to work together. The following are some snippets from that thread, identified by the various proponents.

crop Ah, yes. It cannot work with all those `\settrims` etc. `crop.sty` does only respect `\stock{width,height}`, but none of the other `memoir` specific trim marks stuff ...

crop ... I’m not a `memoir` expert. `crop.sty` respects all the usual LaTeX paper dimensions and simply puts the marks around the page. It works with all standard classes and the KOMA classes. `memoir` seems to do things completely differently ...

Sorry, but I’m afraid I have to pass the problem to the `memoir` author. ...

memoir I will try and take a look at the problem but I’m not a `crop` expert — I’ve never used it. However it seems odd to me that `crop` doesn’t work with `memoir`. `Memoir` provides a different interface for specifying the page layout but then translates everything to the standard length variables. ...

crop You've just taken code from it ...:->

crop doesn't have its own idea at all. It just uses that of DEK and LaTeX, which says, that the reference point of the output box in the output routine is assumed 1 inch to the left [later corrected to right] and 1 inch down from the upper left (virtual) paper corner.

crop With other words: crop changes `\hoffset` and `\voffset`, in order to center the logical (virtual) page on the physical sheet of paper, in the middle of the crop marks. The length macros `\evensidemargin` and `\oddsidemargin` are unchanged and refer still to the logical page (modulo 1 inch).

The memoir class, in contrast (mis)uses the LaTeX margins `\even(odd)sidemargin` for that purpose.

Thus, the meaning of `\even(odd)sidemargin` is different in both packages.

Given all the above I concluded that crop would remain as it was, and that in order to satisfy Peter Heslin I would have to extend the trimming marks. The following was originally limited to extensions asked for by Peter Heslin.

`\showtrimsoff` Switch trimming marksoff and on. Requested by James Hunt on CTT *Are crop marks needed on every page?*, February 2006.

```
11059 \newcommand*{\showtrimsoff}{\showtrimsfalse}
11060 \newcommand*{\showtrimson}{\showtrimstrue}
11061
```

`\trimmark` This is a cross in a zero sized picture for marking the corner of a logical page. Up to version 1.0 the macro used `\setlength` for adjusting the `\unitlength` but Henrik Holm²⁶ discovered that if the calc package is used then LaTeX complains about a missing number.

```
11062 \newcommand*{\trimmark}{%
11063   \begin{picture}(0,0)
11064     \unitlength 1cm
11065     \thinlines
11066     \put(-2,0){\line(1,0){4}}
11067     \put(0,-2){\line(0,1){4}}
11068   \end{picture}}
11069
```

`\Ltrimpictl` 'L' shaped trim marks for four corners.

```
\Ltrimpictl1070 \newcommand*{\Ltrimpictl}{%
\Ltrimpicbl1071   \begin{picture}(0,0)
\Ltrimpicbn1072     \unitlength 1mm
11073     \thinlines
11074     \put(-2,0){\line(-1,0){18}}
11075     \put(0,2){\line(0,1){18}}
11076   \end{picture}}
```

²⁶Message to CTT on 2002/01/04 (h.holm@spray.no)

```

11077 \newcommand*\Ltrimpictr{%
11078   \begin{picture}(0,0)
11079     \unitlength 1mm
11080     \thinlines
11081     \put(2,0){\line(1,0){18}}
11082     \put(0,2){\line(0,1){18}}
11083   \end{picture}}
11084 \newcommand*\Ltrimpicbl{%
11085   \begin{picture}(0,0)
11086     \unitlength 1mm
11087     \thinlines
11088     \put(-2,0){\line(-1,0){18}}
11089     \put(0,-2){\line(0,-1){18}}
11090   \end{picture}}
11091 \newcommand*\Ltrimpicbr{%
11092   \begin{picture}(0,0)
11093     \unitlength 1mm
11094     \thinlines
11095     \put(2,0){\line(1,0){18}}
11096     \put(0,-2){\line(0,-1){18}}
11097   \end{picture}}
11098

```

\Ftrimpicbl Frame the page.

```

11099 \newcommand*\Ftrimpicbl{%
11100   \begin{picture}(0,0)
11101     \unitlength 1pt
11102     \thinlines
11103     \put(0,0){\framebox(\strip@pt\paperwidth,\strip@pt\paperheight){}}
11104   \end{picture}}
11105

```

\tmarktl The trimming marks used for corner display.

```

\tmarktr 11106 \newcommand*\tmarktl{\trimmark}
\tmarkbl 11107 \newcommand*\tmarktr{\trimmark}
\tmarkbr 11108 \newcommand*\tmarkbl{\trimmark}
11109 \newcommand*\tmarkbr{\trimmark}
11110

```

\tmarktm The trimming marks used for mid-side display.

```

\tmarkml 11111 \newcommand*\tmarktm{%
\tmarkmr 11112   \begin{picture}(0,0)%
\tmarkbm 11113     \unitlength 1mm
11114     \thinlines
11115     \put(0,2){\line(0,1){10}}
11116   \end{picture}}
11117 \newcommand*\tmarkml{\tmarkml}%
11118   \begin{picture}(0,0)%
11119     \unitlength 1mm

```

```

11120 \thinlines
11121 \put(-2,0){\line(-1,0){10}}
11122 \end{picture}}
11123 \newcommand*{\tmarkmr}{%
11124 \begin{picture}(0,0)%
11125 \unitlength 1mm
11126 \thinlines
11127 \put(2,0){\line(1,0){10}}
11128 \end{picture}}
11129 \newcommand*{\tmarkbm}{%
11130 \begin{picture}(0,0)%
11131 \unitlength 1mm
11132 \thinlines
11133 \put(0,-12){\line(0,1){10}}
11134 \end{picture}}
11135

```

`\trimXmarks` These are declarations for the different kinds of trimming marks.

```

\trimLmarks 11136 \newcommand*{\trimXmarks}{%
\trimFrame 11137 \let\tmarktl\trimmark
\trimNone 11138 \let\tmarktr\trimmark
11139 \let\tmarkbl\trimmark
11140 \let\tmarkbr\trimmark}
11141 \newcommand*{\trimLmarks}{%
11142 \let\tmarktl\Ltrimpictr
11143 \let\tmarktr\Ltrimpictr
11144 \let\tmarkbl\Ltrimpicbl
11145 \let\tmarkbr\Ltrimpicbr}
11146 \newcommand*{\trimFrame}{%
11147 \let\tmarktl\null
11148 \let\tmarktr\null
11149 \let\tmarkbl\Ftrimpicbl
11150 \let\tmarkbr\null}
11151 \newcommand*{\trimNone}{%
11152 \let\tmarktl\relax
11153 \let\tmarktr\relax
11154 \let\tmarkbl\relax
11155 \let\tmarkbr\relax
11156 \let\tmarktm\relax
11157 \let\tmarkml\relax
11158 \let\tmarkmr\relax
11159 \let\tmarkbm\relax}
11160

```

`\trimmarks` This positions four marks (`\trimmark`) at the corners of a logical page. It is basically a `\vbox` with zero height and width.

Bastiaan Niels Veelo reported (2003/07/11) odd things (e.g., some macros ignored) when changing trim marks. Turns out to be related to the use of `\protect`.

```

11161 \newcommand*{\trimmarks}{%
11162   \vbox to \z@{\vskip-1in \vskip\trimtop % top of logical page
11163     \hb@xt@\z@{\hskip-1in
11164       \ifodd\c@page
11165         \hskip\stockwidth \hskip-\trimedge \hskip-\paperwidth
11166       \else
11167         \if@twoside
11168           \hskip\trimedge % left of logical page
11169         \else
11170           \hskip\stockwidth \hskip-\trimedge \hskip-\paperwidth
11171         \fi
11172       \fi
11173     \vbox to \paperheight{%
11174       \let\protect\relax % <- v1.4 addition
11175       \hb@xt@\paperwidth{\tmarktl\hfil\tmarktm\hfil\tmarktr}%
11176       \vfil
11177       \hb@xt@\paperwidth{\tmarkml\hfil\tmarkmr}%
11178       \vfil
11179       \hb@xt@\paperwidth{\tmarkbl\hfil\tmarkbm\hfil\tmarkbr}}}%
11180   \hss}%
11181 \vss}}
11182

```

29.1 Quark marks

William Adams (2006/08/28) supplied the following code to use trim marks along the style of Quark Xpress.

```

\registrationColour Trim marks on the style of Quark Express.
\quarkmarks
11183 \newcommand*{\registrationColour}[1]{#1}
11184 \newcommand*{\quarkmarks}{%
11185   \renewcommand*{\tmarktl}{\registrationColour{
11186     \begin{picture}(0,0)
11187       \setlength{\unitlength}{1bp}\thicklines
11188       \put(-36,0){\line(1,0){24}}
11189       \put(0,12){\line(0,1){24}}
11190       \put(3,27){\ttfamily\fontsize{8bp}{10bp}\selectfont\jobname\
11191         \today\ \ \printtime\ \ Page \thepage}
11192     \end{picture}}}
11193   \renewcommand*{\tmarktm}{\registrationColour{
11194     \begin{picture}(0,0)
11195       \setlength{\unitlength}{1bp}\thicklines
11196       \put(-24,24){\line(1,0){48}}
11197       \put(0,12){\line(0,1){24}}
11198       \put(0,24){\oval(12,12)}
11199     \end{picture}}}
11200   \renewcommand*{\tmarktr}{\registrationColour{
11201     \begin{picture}(0,0)
11202       \setlength{\unitlength}{1bp}\thicklines

```

```

11203 \put(12,0){\line(1,0){24}}
11204 \put(0,12){\line(0,1){24}}
11205 \end{picture}}
11206 \renewcommand*{\tmarkmr}{\registrationColour{%
11207 \begin{picture}(0,0)
11208 \setlength{\unitlength}{1bp}\thicklines
11209 \put(12,0){\line(1,0){24}}
11210 \put(24,-24){\line(0,1){48}}
11211 \put(24,0){\oval(12,12)}
11212 \end{picture}}}
11213 \renewcommand*{\tmarkbr}{\registrationColour{%
11214 \begin{picture}(0,0)
11215 \setlength{\unitlength}{1bp}\thicklines
11216 \put(12,0){\line(1,0){24}}
11217 \put(0,-36){\line(0,1){24}}
11218 \end{picture}}}
11219 \renewcommand*{\tmarkbm}{\registrationColour{%
11220 \begin{picture}(0,0)
11221 \setlength{\unitlength}{1bp}\thicklines
11222 \put(-24,-24){\line(1,0){48}}
11223 \put(0,-36){\line(0,1){24}}
11224 \put(0,-24){\oval(12,12)}
11225 \end{picture}}}
11226 \renewcommand*{\tmarkbl}{\registrationColour{%
11227 \begin{picture}(0,0)
11228 \setlength{\unitlength}{1bp}\thicklines
11229 \put(-36,0){\line(1,0){24}}
11230 \put(0,-36){\line(0,1){24}}
11231 \end{picture}}}
11232 \renewcommand*{\tmarkml}{\registrationColour{%
11233 \begin{picture}(0,0)
11234 \setlength{\unitlength}{1bp}\thicklines
11235 \put(-36,0){\line(1,0){24}}
11236 \put(-24,-24){\line(0,1){48}}
11237 \put(-24,0){\oval(12,12)}
11238 \end{picture}}}
11239 \renewcommand*{\trimmarks}{%
11240 %% \special{papersize=\the\stockwidth,\the\stockheight}
11241 {%
11242 \vbox to \z@{\vskip-1in \vskip\trimtop % top of logical page
11243 \hb@xt@\z@{\hskip-1in
11244 \ifodd\c@page
11245 \hskip\stockwidth \hskip-\trimedge \hskip-\paperwidth
11246 \else
11247 \if@twoside
11248 \hskip\trimedge % left of logical page
11249 \else
11250 \hskip\stockwidth \hskip-\trimedge \hskip-\paperwidth
11251 \fi
11252 \fi

```

```

11253      \vbox to \paperheight{%
11254          \let\protect\relax %      <- v1.4 addition
11255          \hb@xt@\paperwidth{\tmarktl\hfil\tmarktm\hfil\tmarktr}%
11256          \vfil
11257          \hb@xt@\paperwidth{\tmarkml\hfil\tmarkmr}%
11258          \vfil
11259          \hb@xt@\paperwidth{\tmarkbl\hfil\tmarkbm\hfil\tmarkbr}}%
11260      \hss}%
11261      \vss}}%
11262  }}
11263

```

Any marks are put onto the pages by adding to the `\shipout` routine.

`\mem@oldshipout` Keep a copy of the current version of `\shipout` in `\mem@oldshipout`.

```

11264 \let\mem@oldshipout\shipout

```

`\mem@shipi` Effectively these will add the `\trimmarks` to the box holding the contents of the
`\mem@shipii` page. Note that any `\makeindex` must come *before* `pagesel` (and `selectp`) package.

```

11265 \newcommand*{\mem@shipi}{%
11266     \ifvoid\@cclv\expandafter\aftergroup\fi\mem@shipii}
11267 \newcommand*\mem@shipii{%
11268     \ifvoid\@cclv
11269         \mem@oldshipout\box\@cclv
11270     \else
11271         \ifshowtrims

```

Heiko Oberdiek responded to a problem reported by Rolf Niepraschk. The earlier implementation might cause the output box to be shifted. Heikos explanation (in response to the line

```

\mem@oldshipout\vbox{\trimmarks\ifvbox\@cclv\unvbox\else\box\fi\@cclv})
:

```

In vertical mode (`\mem@oldshipout\vbox{...}`) TeX puts interline skip between two boxes (`\trimmarks` and `\@cclv`), here `\lineskip` (1pt) is put inbetween causing the shift downwards.

Heiko also provided the fix used below. Actually Heikos fix also fixes a problem that might cause glue settings to disappear.

```

11272      \mem@oldshipout\vbox{%
11273          \trimmarks
11274          \nointerlineskip
11275          \box\@cclv
11276      }%
11277      \else
11278          \mem@oldshipout\box\@cclv
11279      \fi
11280      \fi}

```


`\shipout` Our new version of `\shipout`, which is only needed for the `showtrims` option. This adds `\mem@shipi` to the page box which then calls the original version of `\shipout`.

```
11281 \ifshowtrims
11282   \renewcommand*{\shipout}{\afterassignment\mem@shipi\setbox\@cclv=}
11283 \fi
11284
```

30 Verbatims, boxes, and files

All the code in this section was added for version 1.2 of the class.

30.1 Modified version of the verbatim package

Much of this is from the `verbatim` package code [SRR01]. Unless indicated otherwise, the code and commentary is from that package.

30.1.1 Preliminaries

`\every@verbatim` The hook (i.e., token register) `\every@verbatim` is initialized to *empty*.
`\afterevery@verbatim` PW added the `\afterevery@verbatim` hook.

```
11285 \newtoks\every@verbatim
11286   \every@verbatim={}
11287 \newtoks\afterevery@verbatim
11288   \afterevery@verbatim={}
11289
```

`\@makeother` `\@makeother` takes as argument a character and changes its category code to 12 (other).

```
11290 \def\@makeother#1{\catcode'#112\relax}
```

`\@vobeyspaces` The macro `\@vobeyspaces` causes spaces in the input to be printed as spaces in the output.

```
11291 \begingroup
11292   \catcode'\ =\active%
11293   \def\x{\def\@vobeyspaces{\catcode'\ \active\let \@xobeysp}}
11294   \expandafter\endgroup\x
```

`\@xobeysp` The macro `\@xobeysp` produces exactly one space in the output, protected against breaking just before it. (`\@M` is an abbreviation for the number 10000.)

```
11295 \def\@xobeysp{\leavevmode\penalty\@M\ }
```

`\verbatim@line` We use a newly defined token register called `\verbatim@line` that will be used as the character buffer.

```
11296 \newtoks\verbatim@line
```

PW. I have extended the original verbatim package code to handle TABs within verbatims. Normally TeX replaces a TAB by either a single space or ignores it altogether. For this purpose I have bits of code from the moreverb package [Fai98] for handling TABs.

Code and commentary from moreverb.

We define a few auxiliary macros and counters for expanding tabs.

```
11297 \newcount\tab@position
```

`\@xobeytab` `\@xobeytab` puts enough spaces in to get to the next nominal tab stop

```
11298 \def\@xobeytab{%
11299   \loop
11300     \toks@\expandafter{\the\toks@\@xobeysp}%
11301     \advance\tab@position-1
11302   \ifnum\tab@position>0 \repeat
11303 }
```

`\@vobeytabs` `\@vobeytabs` initialises use of `\@xobeytab`. Needs to be executed within a group, as mustn't be allowed to leak out into the wide world.

```
11304 \begingroup
11305   \catcode'\^^I=\active
11306   \gdef\@vobeytabs{\catcode'\^^I\active\let^^I\@xobeytab}%
11307 \endgroup
```

`\verbatim@tabexpand` `\verbatim@tabexpand{<body of line>}` `\@nil` processes every character of a line by tail recursion, counting the characters and juggling things when a tab is encountered.

```
11308 \def\verbatim@tabexpand#1{%
11309   \ifx#1\@nil
11310     \the\toks@
11311   \expandafter\par
11312   \else
11313     \ifx#1\@xobeytab
11314       \@xobeytab
11315     \else
```

We can safely put `\@xobeysp` into the token register, since it does precisely what we need

```
11316     \toks@\expandafter{\the\toks@#1}%
11317     \advance\tab@position\m@ne
11318   \fi
11319   \ifnum\tab@position=0 \tab@position\tab@size \fi
11320   \expandafter\verbatim@tabexpand
11321 \fi
11322 }
11323
```

End of code and commentary from moreverb.

PW. Some macros for turning tabbing on and off.

`\tabson` `\tabson` turns tabbing on, and `\tabsoff` turns it off. Default is no tabbing.

```

\tabsoff{11324 \newif\ift@bs
\maybeobeytabs11325 \newcommand{\tabson}[1][4]{%
11326   \ifnum\@ne > #1\relax
11327     \tabsoff
11328   \else
11329     \t@bstrue
11330     \def\tab@size{#1\relax}%
11331     \def\maybeobeytabs{\@vobeytabs}%
11332   \fi
11333 }
11334 \newcommand{\tabsoff}{%
11335   \t@bsfalse
11336   \def\tab@size{z}%
11337   \def\maybeobeytabs{}%
11338 }
11339 \tabsoff
11340

```

`\tabverbatim@processline` Process a line with TABs (extracted from `moreverb`).

```

11341 \def\tabverbatim@processline{\tab@position\tab@size
11342   \toks@{}}%
11343 \expandafter\verbatim@tabexpand\the\verbatim@line\@nil}

```

`\notabverbatim@processline` Processes a line ignoring TABs (this is the original `verbatim` package definition of `\verbatim@processline`).

```

11344 \def\notabverbatim@processline{\the\verbatim@line\par}
11345

```

We are now back to the verbatim code.

The following four macros are defined globally in a way suitable for the `verbatim` and `verbatim*` environments.

```

\verbatim@startline \verbatim@startline initializes processing of a line by emptying the character
\verbatim@addtoline buffer (\verbatim@line).
\verbatim@processline11346 \def\verbatim@startline{\verbatim@line{}}

```

`\verbatim@addtoline` adds the tokens in its argument to our buffer register `\verbatim@line` without expanding them.

```

11347 \def\verbatim@addtoline#1{%
11348   \verbatim@line\expandafter{\the\verbatim@line#1}}

```

Processing a line inside a `verbatim` or `verbatim*` environment means printing it. Ending the line means that we have to begin a new paragraph. We use `\par` for this purpose. Note that `\par` is redefined in `\@verbatim` to force `TEX` into horizontal mode and to insert an empty box so that empty lines in the input do appear in the output. (PW changed next line from

```

\def\verbatim@processline{\the\verbatim@line\par}
11349 \def\verbatim@processline{\notabverbatim@processline}

```

`\verbatim@finish` As a default, `\verbatim@finish` processes the remaining characters. When this macro is called we are facing the following problem: when the `\end{verbatim}` command is encountered `\verbatim@processline` is called to process the characters preceding the command on the same line. If there are none, an empty line would be output if we did not check for this case.

If the line is empty `\the\verbatim@line` expands to nothing. To test this we use a trick similar to that on p. 376 of the *TeXbook*, but with `$. . . |$` instead of the `!` tokens. These `$` tokens can never have the same category code as a `$` token that might possibly appear in the token register `\verbatim@line`, as such a token will always have been read with category code 12 (other). Note that `\ifcat` expands the following tokens so that `\the\verbatim@line` is replaced by the accumulated characters

```
11350 \def\verbatim@finish{\ifcat$\the\verbatim@line$\else
11351   \verbatim@processline\fi}
```

30.1.2 The verbatim and verbatim* environments

`\verbatim@font` We start by defining the macro `\verbatim@font` that is to select the font and to set font-dependent parameters. Then we go through `\verbatim@nolig@list` to avoid certain ligatures. `\verbatim@nolig@list` is a macro defined in the $\text{\LaTeX} 2_{\epsilon}$ kernel to expand to

```
\do\‘\do\<\do\>\do\,\do\’\do\-
```

All the characters in this list can be part of a ligature in some font or other. PW. This is the original version which I’m going to replace.

```
\def\verbatim@font{\normalfont\ttfamily
\hyphenchar\font\m@ne
\let\do\do@noligs
\verbatim@nolig@list}
```

Actually the kernel defines the macro `\@noligs` which just runs the last two lines of the `\verbatim@font` above. As other package may add stuff to `\@noligs`, we will use that instead.

`\setverbatimfont` User level handle for changing the font used for verbatim text.

```
\m@mverbfont 11352 \newcommand{\setverbatimfont}[1]{\def\m@mverbfont{#1}}
\verbatim@font 11353 \setverbatimfont{\normalfont\ttfamily}
11354
11355 \def\verbatim@font{\m@mverbfont
11356   \hyphenchar\font\m@ne
11357   \@noligs}
11358
```

`\@verbatim` The macro `\@verbatim` sets up things properly. First of all, the tokens of the `\every@verbatim` hook are inserted. Then a `trivlist` environment is started

and its first `\item` command inserted. Each line of the `verbatim` or `verbatim*` environment will be treated as a separate paragraph.

```
11359 \def\@verbatim{\the\every@verbatim
11360 \trivlist \item \relax
```

The following extra vertical space is for compatibility with the \LaTeX kernel: otherwise, using the `verbatim` package changes the vertical spacing of a `verbatim` environment nested within a `quote` environment.

```
11361 \if@minipage\else\vskip\parskip\fi
```

The paragraph parameters are set appropriately: the penalty at the beginning of the environment, left and right margins, paragraph indentation, the glue to fill the last line, and the vertical space between paragraphs. The latter space has to be zero since we do not want to add extra space between lines.

```
11362 \@beginparpenalty \predisplaypenalty
11363 %% \leftskip\@totalleftmargin\rightskip\z@
11364 \memRTLleftskip\@totalleftmargin\memRTLrightskip\z@
11365 \parindent\z@\parfillskip\@flushglue\parskip\z@
```

There's one point to make here: the `list` environment uses \TeX 's `\parshape` primitive to get a special indentation for the first line of the list. If the list begins with a `verbatim` environment this `\parshape` is still in effect. Therefore we have to reset this internal parameter explicitly. We could do this by assigning 0 to `\parshape`. However, there is a simpler way to achieve this: we simply tell \TeX to start a new paragraph. As is explained on p. 103 of the \TeX book, this resets `\parshape` to zero.

```
11366 \@@par
```

We now ensure that `\par` has the correct definition, namely to force \TeX into horizontal mode and to include an empty box. This is to ensure that empty lines do appear in the output. Afterwards, we insert the `\interlinepenalty` since \TeX does not add a penalty between paragraphs (here: lines) by its own initiative. Otherwise a `verbatim` environment could be broken across pages even if a `\samepage` declaration were present.

However, in a top-aligned minipage, this will result in an extra empty line added at the top. Therefore, a slightly more complicated construct is necessary. One of the important things here is the inclusion of `\leavevmode` as the first macro in the first line, for example, a blank `verbatim` line is the first thing in a list item.

```
11367 \def\par{%
11368   \if@tempswa
11369     \leavevmode\null\@@par\penalty\interlinepenalty
11370   \else
11371     \@tempswatrue
11372     \ifhmode\@@par\penalty\interlinepenalty\fi
11373   \fi}%
```

But to avoid an error message when the environment doesn't contain any text, we redefine `\@noitemerr` which will in this case be called by `\endtrivlist`.

```
11374 \def\@noitemerr{\@warning{No verbatim text}}%
```

Now we call `\obeylines` to make the end of line character active,

```
11375 \obeylines
```

change the category code of all special characters, to 12 (other).

```
11376 \let\do\@makeother \dospecials
```

and switch to the font to be used.

```
11377 \verbatim@font
```

To avoid a breakpoint after the labels box, we remove the penalty put there by the list macros: another use of `\unpenalty`!

```
11378 \everypar \expandafter{\the\everypar \unpenalty}%
```

PW added next code at end of `\@verbatim`.

```
11379 \wrapright\the\afterevery@verbatim}
```

`\verbatim` Now we define the toplevel macros. `\verbatim` is slightly changed: after setting up things properly it calls `\verbatim@start`. This is done inside a group, so that `\verbatim` can be used directly, without `\begin`.

PW. The following is the original code, but I want a version of `verbatim` that recognises TABs.

```
\begin{macrocode}
\def\verbatim{%
  \begingroup\@verbatim \frenchspacing\@vobeyspaces
  \verbatim@start}
\end{macrocode}
\cs{verbatim*} is defined accordingly.
\begin{macrocode}
\@namedef{verbatim*}{\begingroup\@verbatim\verbatim@start}
\def\endverbatim{\endtrivlist\endgroup}
\expandafter\let\csname endverbatim*\endcsname=\endverbatim
\end{macrocode}
```

PW. My code for these is a modified version of the original `verbatim` code.

```
11380 \def\verbatim{\begingroup
11381   \ift@bs
11382   \def\verbatim@processline{\tabverbatim@processline}%
11383   \fi
11384   \@verbatim \frenchspacing\@vobeyspaces\@maybeobeytabs\verbatim@start}
11385 \@namedef{verbatim*}{\begingroup
11386   \ift@bs
11387   \def\verbatim@processline{\tabverbatim@processline}%
11388   \fi
11389   \@verbatim\@maybeobeytabs\verbatim@start}
```

`\endverbatim` To end the `verbatim` and `verbatim*` environments it is only necessary to finish the `trivlist` environment started in `\@verbatim` and close the corresponding group, and handle²⁷ following (non-) paragraph, by using `\@doendpe`.

²⁷Noted by Zarko Cucej (zarko.cucej@uni-mb.si).

```

11390 \def\endverbatim{\endtrivlist\endgroup\@doendpe}
11391 \@namelet{endverbatim*}\endverbatim
11392

```

For abnormal `\parskip` the NTG class included the following, but I'm not sure if it is relevant here, but if it is just how it should be included.

```

From NTG, where it is a \cs{def}
\providecommand*{\verbatim}{%
  \topsep=-0.5\parskip
  \@verbatim
  \frenchspacing\@vobeyspaces \@xverbatim}

```

30.1.3 The comment environment

The `\comment` macro is similar to `\verbatim*`. However, we do not need to switch fonts or set special formatting parameters such as `\parindent` or `\parskip`. We need only set the category code of all special characters to 12 (other) and that of `~` (the end of line character) to 13 (active). The latter is needed for macro parameter delimiter matching in the internal macros defined below. In contrast to the default definitions used by the `\verbatim` and `\verbatim*` macros, we define `\verbatim@addtoline` to throw away its argument and `\verbatim@processline`, `\verbatim@startline`, and `\verbatim@finish` to act as no-ops. Then we call `\verbatim@`. But the first thing we do is to call `\@bsphack` so that this environment has no influence whatsoever upon the spacing.

PW: This is the original code for the `comment` environment, which I'm going to change.

```

\def\comment{\@bsphack
  \let\do\@makeother\dospecials\catcode'\~M\active
  \let\verbatim@startline\relax
  \let\verbatim@addtoline\@gobble
  \let\verbatim@processline\relax
  \let\verbatim@finish\relax
  \verbatim@}

```

`\endcomment` is very simple: it only calls `\@esphack` to take care of the spacing. The `\end` macro closes the group and therefore takes care of restoring everything we changed.

```

\let\endcomment=\@esphack

```

PW: The remainder of this section is my code.

```

\setupcomment \setupcomment does all the \lets in the original \comment code.
11393 \newcommand{\setupcomment}{%

```

```

11394 \let\do\@makeother\dospecials\catcode'\^^M\active
11395 \let\verbatim@startline\relax
11396 \let\verbatim@addtoline\@gobble
11397 \let\verbatim@processline\relax
11398 \let\verbatim@finish\relax}

```

The macros below do no checking to see if something has (not) been defined previously. It's 'user beware' time.

`\newcomment` `\newcomment{<name>}` creates a new comment environment called `<name>`. This is a generalisation of the original comment code.

```

11399 \newcommand{\newcomment}[1]{%
11400   \expandafter\def\csname #1\endcsname{\@bsphack\setupcomment\verbatim@}%
11401   \expandafter\let\csname end#1\endcsname=\@esphack}

```

`\commentsoff` `\commentsoff{<name>}` switches off the `<name>` comment environment by defining the relevant macros to do nothing.

```

11402 \newcommand{\commentsoff}[1]{%
11403   \expandafter\def\csname #1\endcsname{}%
11404   \expandafter\def\csname end#1\endcsname{}}

```

`\commentson` `\commentson{<name>}` switches on the `<name>` comment environment. It has to do the same things as `\newcomment` does, so let `\newcomment` do the work.

```

11405 \newcommand{\commentson}[1]{\newcomment{#1}}
11406

```

We had better supply the `comment` environment, as promised.

```

11407 \newcommand{comment}
11408

```

PW: That is the end of my changes and extensions to the original `comment` environment code.

30.1.4 The main loop

Here comes the tricky part: During the definition of the macros we need to use the special characters `\`, `{`, and `}` not only with their normal category codes, but also with category code 12 (other). We achieve this by the following trick: first we tell `TEX` that `\`, `{`, and `}` are the lowercase versions of `!`, `[`, and `]`. Then we replace every occurrence of `\`, `{`, and `}` that should be read with category code 12 by `!`, `[`, and `]`, respectively, and give the whole list of tokens to `\lowercase`, knowing that category codes are not altered by this primitive!

But first we have ensure that `!`, `[`, and `]` themselves have the correct category code! To allow special settings of these codes we hide their setting in the macro `\vrb@catcodes`. If it is already defined our new definition is skipped.

```

11409 \@ifundefined{vrb@catcodes}%
11410   {\def\vrb@catcodes{%
11411     \catcode'\!12\catcode'\[12\catcode'\]12}}{}

```


This trick allows us to use this code for applications where other category codes are in effect.

We start a group to keep the category code changes local.

```
11412 \begingroup
11413 \vrb@catcodes
11414 \lccode'\!='\ \lccode'\[='\{ \lccode'\]='\}
```

We also need the end-of-line character `^^M`, as an active character. If we were to simply write `\catcode'\^^M=\active` then we would get an unwanted active end of line character at the end of every line of the following macro definitions. Therefore we use the same trick as above: we write a tilde `~` instead of `^^M` and pretend that the latter is the lowercase variant of the former. Thus we have to ensure now that the tilde character has category code 13 (active).

```
11415 \catcode'\~= \active \lccode'\~= '\^^M
```

The use of the `\lowercase` primitive leads to one problem: the uppercase character `'C'` needs to be used in the code below and its case must be preserved. So we add the command:

```
11416 \lccode'\C= '\C
```

Now we start the token list passed to `\lowercase`. We use the following little trick (proposed by Bernd Raichle): The very first token in the token list we give to `\lowercase` is the `\endgroup` primitive. This means that it is processed by `TeX` immediately after `\lowercase` has finished its operation, thus ending the group started by `\begingroup` above. This avoids the global definition of all macros.

```
11417 \lowercase{\endgroup
```

`\verbatim@start` The purpose of `\verbatim@start` is to check whether there are any characters on the same line as the `\begin{verbatim}` and to pretend that they were on a line by themselves. On the other hand, if there are no characters remaining on the current line we shall just find an end of line character. `\verbatim@start` performs its task by first grabbing the following character (its argument). This argument is then compared to an active `^^M`, the end of line character.

```
11418 \def\verbatim@start#1{%
11419 \verbatim@startline
11420 \if\noexpand#1\noexpand~%
```

If this is true we transfer control to `\verbatim@` to process the next line. We use `\next` as the macro which will continue the work.

```
11421 \let\next\verbatim@
```

Otherwise, we define `\next` to expand to a call to `\verbatim@` followed by the character just read so that it is reinserted into the text. This means that those characters remaining on this line are handled as if they formed a line by themselves.

```
11422 \else \def\next{\verbatim@#1}\fi
```

Finally we call `\next`.

```
11423 \next}%
```

`\verbatim@` The three macros `\verbatim@`, `\verbatim@@`, and `\verbatim@@@` form the “main loop” of the `verbatim` environment. The purpose of `\verbatim@` is to read exactly one line of input. `\verbatim@@` and `\verbatim@@@` work together to find out whether the four characters `\end` (all with category code 12 (other)) occur in that line. If so, `\verbatim@@@` will call `\verbatim@test` to check whether this `\end` is part of `\end{verbatim}` and will terminate the environment if this is the case. Otherwise we continue as if nothing had happened. So let’s have a look at the definition of `\verbatim@`:

```
11424 \def\verbatim@#1~{\verbatim@@#1!end\@nil}%
```

Note that the `!` character will have been replaced by a `\` with category code 12 (other) by the `\lowercase` primitive governing this code before the definition of this macro actually takes place. That means that it takes the line, puts `\end` (four character tokens) and `\@nil` (one control sequence token) as a delimiter behind it, and then calls `\verbatim@@`.

`\verbatim@@` `\verbatim@@` takes everything up to the next occurrence of the four characters `\end` as its argument.

```
11425 \def\verbatim@@#1!end{%
```

That means: if they do not occur in the original line, then argument `#1` is the whole input line, and `\@nil` is the next token to be processed. However, if the four characters `\end` are part of the original line, then `#1` consists of the characters in front of `\end`, and the next token is the following character (always remember that the line was lengthened by five tokens). Whatever `#1` may be, it is verbatim text, so `#1` is added to the line currently built.

```
11426 \verbatim@addtoline{#1}%
```

The next token in the input stream is of special interest to us. Therefore `\futurelet` defines `\next` to be equal to it before calling `\verbatim@@@`.

```
11427 \futurelet\next\verbatim@@@}%
```

`\verbatim@@@` `\verbatim@@@` will now read the rest of the tokens on the current line, up to the final `\@nil` token.

```
11428 \def\verbatim@@@#1\@nil{%
```

If the first of the above two cases occurred, i.e. no `\end` characters were on that line, `#1` is empty and `\next` is equal to `\@nil`. This is easily checked.

```
11429 \ifx\next\@nil
```

If so, this was a simple line. We finish it by processing the line we accumulated so far. Then we prepare to read the next line.

```
11430 \verbatim@processline
```

```
11431 \verbatim@startline
```

```
11432 \let\next\verbatim@
```

Otherwise we have to check what follows these `\end` tokens.

```
11433 \else
```

Before we continue, it's a good idea to stop for a moment and remember where we are: We have just read the four character tokens `\end` and must now check whether the name of the environment (surrounded by braces) follows. To this end we define a macro called `\@tempa` that reads exactly one character and decides what to do next. This macro should do the following: skip spaces until it encounters either a left brace or the end of the line. But it is important to remember which characters are skipped. The `\end{optional spaces}` characters may be part of the verbatim text, i.e. these characters must be printed. Assume for example that the current line contains

```
uuuuuu\end_u{AVeryLongEnvironmentName}
```

As we shall soon see, the scanning mechanism implemented here will not find out that this is text to be printed until it has read the right brace. Therefore we need a way to accumulate the characters read so that we can reinsert them if necessary. The token register `\@temptokena` is used for this purpose.

Before we do this we have to get rid of the superfluous `\end` tokens at the end of the line. To this end we define a temporary macro whose argument is delimited by `\end\@nil` (four character tokens and one control sequence token) to be used below on the rest of the line, after appending a `\@nil` token to it. (Note that this token can never appear in #1.) We use the following definition of `\@tempa` to get the rest of the line (after the first `\end`).

```
11434 \def\@tempa##1!end\@nil{##1}%
```

We mentioned already that we use token register `\@temptokena` to remember the characters we skip, in case we need them again. We initialize this with the `\end` we have thrown away in the call to `\@tempa`.

```
11435 \@temptokena{!end}%
```

We shall now call `\verbatim@test` to process the characters remaining on the current line. But wait a moment: we cannot simply call this macro since we have already read the whole line. Therefore we have to first expand the macro `\@tempa` to insert them again after the `\verbatim@test` token. A `^^M` character is appended to denote the end of the line. (Remember that this character comes disguised as a tilde.)

```
11436 \def\next{\expandafter\verbatim@test\@tempa#1\@nil~}%
```

That's almost all, but we still have to now call `\next` to do the work.

```
11437 \fi \next}%
```

`\verbatim@test` We define `\verbatim@test` to investigate every token in turn.

```
11438 \def\verbatim@test#1{%
```

First of all we set `\next` equal to `\verbatim@test` in case this macro must call itself recursively in order to skip spaces.

```
11439 \let\next\verbatim@test
```

We have to distinguish four cases:

1. The next token is a \sim M, i.e. we reached the end of the line. That means that nothing special was found. Note that we use `\if` for the following comparisons so that the category code of the characters is irrelevant.

```
11440 \if\noexpand#1\noexpand~%
```

We add the characters accumulated in token register `\@temptokena` to the current line. Since `\verbatim@addtoline` does not expand its argument, we have to do the expansion at this point. Then we `\let \next` equal to `\verbatim@` to prepare to read the next line.

```
11441 \expandafter\verbatim@addtoline
11442 \expandafter{\the\@temptokena}%
11443 \verbatim@processline
11444 \verbatim@startline
11445 \let\next\verbatim@
```

2. A space character follows. This is allowed, so we add it to `\@temptokena` and continue.

```
11446 \else \if\noexpand#1
11447 \@temptokena\expandafter{\the\@temptokena#1}%
```

3. An open brace follows. This is the most interesting case. We must now collect characters until we read the closing brace and check whether they form the environment name. This will be done by `\verbatim@testend`, so here we let `\next` equal this macro. Again we will process the rest of the line, character by character. The characters forming the name of the environment will be accumulated in `\@tempc`. We initialize this macro to expand to nothing.

```
11448 \else \if\noexpand#1\noexpand[%
11449 \let\@tempc\@empty
11450 \let\next\verbatim@testend
```

Note that the `[` character will be a `{` when this macro is defined.

4. Any other character means that the `\end` was part of the verbatim text. Add the characters to the current line and prepare to call `\verbatim@` to process the rest of the line.

```
11451 \else
11452 \expandafter\verbatim@addtoline
11453 \expandafter{\the\@temptokena}%
11454 \def\next{\verbatim@#1}%
11455 \fi\fi\fi
```

The last thing this macro does is to call `\next` to continue processing.

```
11456 \next}%
```

`\verbatim@testend` `\verbatim@testend` is called when `\end{optional spaces}` was seen. Its task is to scan everything up to the next `}` and to call `\verbatim@@testend`. If no `}` is found it must reinsert the characters it read and return to `\verbatim@`. The following definition is similar to that of `\verbatim@test`: it takes the next character and decides what to do.

```
11457 \def\verbatim@testend#1{%
```

Again, we have four cases:

1. `^~M`: As no `}` is found in the current line, add the characters to the buffer. To avoid a complicated construction for expanding `\@temptokena` and `\@tempc` we do it in two steps. Then we continue with `\verbatim@` to process the next line.

```
11458 \if\noexpand#1\noexpand~%
11459 \expandafter\verbatim@addtoline
11460 \expandafter{\the\@temptokena[]}%
11461 \expandafter\verbatim@addtoline
11462 \expandafter{\@tempc}%
11463 \verbatim@processline
11464 \verbatim@startline
11465 \let\next\verbatim@
```

2. `}`: Call `\verbatim@@testend` to check if this is the right environment name.

```
11466 \else\if\noexpand#1\noexpand]%
11467 \let\next\verbatim@@testend
```

3. `\`: This character must not occur in the name of an environment. Thus we stop collecting characters. In principle, the same argument would apply to other characters as well, e.g., `{`. However, `\` is a special case, since it may be the first character of `\end`. This means that we have to look again for `\end{environment name}`. Note that we prefixed the `!` by a `\noexpand` primitive, to protect ourselves against it being an active character.

```
11468 \else\if\noexpand#1\noexpand!%
11469 \expandafter\verbatim@addtoline
11470 \expandafter{\the\@temptokena[]}%
11471 \expandafter\verbatim@addtoline
11472 \expandafter{\@tempc}%
11473 \def\next{\verbatim@!}%
```

4. Any other character: collect it and continue. We cannot use `\edef` to define `\@tempc` since its replacement text might contain active character tokens.

```
11474 \else \expandafter\def\expandafter\@tempc\expandafter
11475 {\@tempc#1}\fi\fi\fi
```

As before, the macro ends by calling itself, to process the next character if appropriate.

```
11476 \next}%
```

`\verbatim@@testend` Unlike the previous macros `\verbatim@@testend` is simple: it has only to check if the `\end{...}` matches the corresponding `\begin{...}`

```
11477 \def\verbatim@@testend%
```

We use `\next` again to define the things that are to be done. Remember that the name of the current environment is held in `\@currenvir`, the characters accumulated by `\verbatim@testend` are in `\@tempc`. So we simply compare these and prepare to execute `\end{current environment}` macro if they match. Before we do this we call `\verbatim@finish` to process the last line. We define `\next` via `\edef` so that `\@currenvir` is replaced by its expansion. Therefore we need `\noexpand` to inhibit the expansion of `\end` at this point.

```
11478 \ifx\@tempc\@currenvir
```

```
11479 \verbatim@finish
```

```
11480 \edef\next{\noexpand\end{\@currenvir}}%
```

Without this trick the `\end` command would not be able to correctly check whether its argument matches the name of the current environment and you'd get an interesting L^AT_EX error message such as:

```
! \begin{verbatim*} ended by \end{verbatim*}.
```

But what do we do with the rest of the characters, those that remain on that line? We call `\verbatim@rescan` to take care of that. Its first argument is the name of the environment just ended, in case we need it again.

`\verbatim@rescan` takes the list of characters to be reprocessed as its second argument. (This token list was inserted after the current macro by `\verbatim@@@`.) Since we are still in an `\edef` we protect it by means of `\noexpand`.

```
11481 \noexpand\verbatim@rescan{\@currenvir}}%
```

If the names do not match, we reinsert everything read up to now and prepare to call `\verbatim@` to process the rest of the line.

```
11482 \else
```

```
11483 \expandafter\verbatim@addtoline
```

```
11484 \expandafter{\the\@temptokena}%
```

```
11485 \expandafter\verbatim@addtoline
```

```
11486 \expandafter{\@tempc}}%
```

```
11487 \let\next\verbatim@
```

```
11488 \fi
```

Finally we call `\next`.

```
11489 \next}%
```

`\verbatim@rescan` In principle `\verbatim@rescan` could be used to analyse the characters remaining after the `\end{...}` command and pretend that these were read “properly”, assuming “standard” category codes are in force.²⁸ But this is not always possible (when there are unmatched curly braces in the rest of the line). Besides, we think that this is not worth the effort: After a `verbatim` or

²⁸Remember that they were all read with category codes 11 (letter) and 12 (other) so that control sequences are not recognized as such.

`verbatim*` environment a new line in the output is begun anyway, and an `\end{comment}` can easily be put on a line by itself. So there is no reason why there should be any text here. For the benefit of the user who did put something there (a comment, perhaps) we simply issue a warning and drop them. The method of testing is explained in Appendix D, p. 376 of the *TEXbook*. We use `^^M` instead of the `!` character used there since this is a character that cannot appear in `#1`. The two `\noexpand` primitives are necessary to avoid expansion of active characters and macros.

One extra subtlety should be noted here: remember that the token list we are currently building will first be processed by the `\lowercase` primitive before *TEX* carries out the definitions. This means that the `'C'` character in the argument to the `\@warning` macro must be protected against being changed to `'c'`. That's the reason why we added the `\lccode'\C='\'C` assignment above. We can now finish the argument to `\lowercase` as well as the group in which the category codes were changed.

```
11490 \def\verbatim@rescan#1#2~{\if\noexpand~\noexpand#2~\else
11491 \warning{Characters dropped after '\string\end{#1}'}\fi}}
```

30.1.5 The `\verbatiminput` command

`\verbatimin@stream` We begin by allocating an input stream (out of the 16 available input streams).

```
11492 \newread\verbatimin@stream
```

`\verbatim@readfile` The macro `\verbatim@readfile` encloses the main loop by calls to the macros `\verbatim@startline` and `\verbatim@finish`, respectively. This makes sure that the user can initialize and finish the command when the file is empty or doesn't exist. The `verbatim` environment has a similar behaviour when called with an empty text.

```
11493 \def\verbatim@readfile#1{%
11494 \verbatim@startline
```

When the file is not found we issue a warning.

```
11495 \openin\verbatimin@stream #1\relax
11496 \ifeof\verbatimin@stream
11497 \typeout{No file #1.}%
11498 \else
```

At this point we pass the name of the file to `\@addtofilelist` so that its appears appears in the output of a `\listfiles` command. In addition, we use `\ProvidesFile` to make a log entry in the transcript file and to distinguish files read in via `\verbatiminput` from others.

```
11499 \@addtofilelist{#1}%
11500 \ProvidesFile{#1}[\verbatimim]%
```

While reading from the file it is useful to switch off the recognition of the end-of-line character. This saves us stripping off spaces from the contents of the line.

```
11501 \expandafter\endlinechar\expandafter\m@ne
```

```

11502 \expandafter\verbatim@read@file
11503 \expandafter\endlinechar\the\endlinechar\relax
11504 \closein\verbatim@in@stream
11505 \fi
11506 \verbatim@finish
11507 }

```

`\verbatim@read@file` All the work is done in `\verbatim@read@file`. It reads the input file line by line and recursively calls itself until the end of the file.

```

11508 \def\verbatim@read@file{%
11509 \read\verbatim@in@stream to\next
11510 \ifeof\verbatim@in@stream
11511 \else

```

For each line we call `\verbatim@addtoline` with the contents of the line.
`\verbatim@processline` is called next.

```

11512 \expandafter\verbatim@addtoline\expandafter{\next}%
11513 \verbatim@processline

```

After processing the line we call `\verbatim@startline` to initialize all before we read the next line.

```

11514 \verbatim@startline

```

Without `\expandafter` each call of `\verbatim@read@file` uses space in T_EX's input stack.²⁹

```

11515 \expandafter\verbatim@read@file
11516 \fi
11517 }

```

`\verbatiminput` `\verbatiminput` first starts a group to keep font and category changes local. Then it calls the macro `\verbatim@input` with additional arguments, depending on whether an asterisk follows.
 PW. I added the TAB checking code.

```

11518 \def\verbatiminput{\begingroup
11519 \ift@bs
11520 \def\verbatim@processline{\tabverbatim@processline}%
11521 \fi
11522 \@ifstar{\verbatim@input{\@maybeobeytabs}}%
11523 {\verbatim@input{\frenchspacing\@vobeyspaces\@maybeobeytabs}}}

```

`\verbatim@input` `\verbatim@input` first checks whether the file exists, using the standard macro `csIfFileExists` which leaves the name of the file found in `\@filef@und`. Then everything is set up as in the `\verbatim` macro.

```

11524 \def\verbatim@input#1#2{%
11525 \IfFileExists {#2}{\@verbatim #1\relax

```

²⁹A standard T_EX would report an overflow error if you try to read a file with more than ca. 200 lines. The same error occurs if the first line of code in §390 of “*T_EX: The Program*” is missing.

Then it reads in the file, finishes off the `trivlist` environment started by `\@verbatim` and closes the group. This restores everything to its normal settings.

```
11526 \verbatim@readfile{\@file@und}\endtrivlist\endgroup\@doendpe}%
```

If the file is not found a more or less helpful message is printed. The final `\endgroup` is needed to close the group started in `\verbatiminput` above.

```
11527 {\typeout {No file #2.}\endgroup}}
```

That completes my borrowings from verbatim.

The next bunch of code implements wrapping verbatim lines so they, hopefully, stay within the typeblock.

`\verbatimindent` The length `\verbatimindent` is the distance continuation lines are indented from the left margin. `\verbatimbreakchar` is the character to indicate a wrapped line.

```
11528 \newlength{\verbatimindent}
11529 \setlength{\verbatimindent}{3em}
11530 \newcommand*\verbatimbreakchar{\char'\%}
11531 \newcommand*\setverbatimbreak{%
11532   \vspace*{-\baselineskip}%
11533   \def\xobeysp{~\discretionary{\verbatimbreakchar}%
11534     {\kern\verbatimindent}{}}%
11535 }
11536
```

`\raggedwrap` `\wrappingon` and `\wrappingoff` enable/prohibit wrapping. The default is

`\wrappingon` `\wrappingoff`.

`\wrapright` The macro `\wrapright` is used to set paragraph skips; without `raggedright` the lines may break at the first space *outside* the text area. However, Paul (paulaugust2003@yahoo.com) found that wrapped verbatims in a list (e.g., `itemize`) were not indented although regular verbatims were indented. `\raggedwrap` is a variation on `\raggedright` (`\leftskip` is set to `\@totalleftmargin` instead of 0pt), and seems to have fixed the problem with the original code which used `\raggedright`.

```
11537 \newcommand*\raggedwrap{%
11538   \@rightskip\@flushglue
11539   %% \rightskip\@rightskip
11540   \memRTLrightskip\@rightskip
11541   %% \leftskip\@totalleftmargin
11542   \memRTLleftskip\@totalleftmargin
11543   \parindent\ragrparindent}
11544 \newcommand*\wrappingon{%
11545   \def\xobeysp{~\discretionary{\verbatimbreakchar}%
11546     {\kern\verbatimindent}{}}%
11547   \def\wrapright{\raggedwrap}}
11548 \newcommand*\wrappingoff{%
11549   \def\xobeysp{\leavevmode\penalty\@M\ }%
11550   \def\wrapright{}}
11551 \wrappingoff
```

11552

30.2 Writing and boxing verbatim

This bunch of code is from the `moreverb` package.

`verbatimoutput` `\begin{verbatimoutput}{\filename}` writes all text in its body to a file, the name of which it is given as an argument. (The code was written by Rainer Schöpf but the environment was called `verbatimwrite`). In the code below, uncommenting

`\catcode'\^^I=12`

will result in TABs being written as `^^I`.

```
11553 \newwrite \verbatim@out
11554 \def\verbatimoutput#1{%
11555   \@bsphack
11556   \immediate\openout \verbatim@out #1
11557   \let\do\@makeother\dospecials
11558   \catcode'\^^M\active %% \catcode'\^^I=12
11559   \def\verbatim@processline{%
11560     \immediate\write\verbatim@out
11561     {\the\verbatim@line}}%
11562   \verbatim@start}
11563 \def\endverbatimoutput{%
11564   \immediate\closeout\verbatim@out
11565   \@esphack}
```

`fboxverbatim` `fboxverbatim` puts the contents of a verbatim environment in a framing box. (PW: This was originally called `boxedverbatim`).

(Written by Victor Eijkhout.)

Bug fix (supplied by David Carlisle) 1995/12/28, marked `%%DPC%%`

First, redefine 'processline' to produce only a line as wide as the natural width of the line

```
11566 \def\fboxverbatim{\begingroup%
11567   \tabsoff %% PW otherwise box fills the width
11568   \def\verbatim@processline{%
11569     {\setbox0=\hbox{\the\verbatim@line}}%
11570     \hsize=\wd0 \the\verbatim@line\par}}%
```

Now save the verbatim code in a box

```
11571 \@minipagetrue%%DPC%%
11572 \@tempwattrue%%DPC%%
11573 \setbox0=\vbox\bgroup \verbatim
11574 }
```

At the end of the environment, we (umm) simply have to stick the results into a frame.

```
11575 \def\endfboxverbatim{%
11576   \endverbatim
11577   \unskip\setbox0=\lastbox %%DPC%%}
```

Now everything's in the box, so we can close it...

```
11578 \egroup
```

To change the code for centering, the next line needs a spot of hacking.

```
11579 \fbox{\box0}% <<<=== change here for centering,...
```

```
11580 \endgroup}
```

30.3 The shortvrb package

The following is the shortvrb package code (from doc.dtx by Frank Mittelbach).

It has been so useful to me that I wanted to include it in the class.

CODE AND COMMENTARY IS BY FRANK MITTELBACH

`\MakeShortVerb`

```
11581 \def\MakeShortVerb#1{%
11582   \expandafter\ifx\csname cc\string#1\endcsname\relax
11583     \@shortvrbinfo{Made }{#1}%
11584     \add@special{#1}%
```

Then the character's current catcode is stored in `\cc\langle c \rangle`.

```
11585   \expandafter
11586   \xdef\csname cc\string#1\endcsname{\the\catcode'#1}%
```

The character is spliced into the definition using the same trick as used in `\verb` (for instance), having activated `~` in a group.

```
11587   \begingroup
11588     \catcode'\~\active \lccode'\~'#1%
11589     \lowercase{%
```

The character's old meaning is recorded in `\ac\langle c \rangle` prior to assigning it a new one.

```
11590     \global\expandafter\let
11591     \csname ac\string#1\endcsname~%
11592     \gdef~{\verb~}}%
11593   \endgroup
```

Finally the character is made active.

```
11594   \global\catcode'#1\active
```

If we suspect that `\langle c \rangle` is already a short reference, we tell the user. Now he or she is responsible if anything goes wrong...

```
11595   \else
11596     \@shortvrbinfo\@empty{#1 already}%
11597   \fi}
```

`\DeleteShortVerb` Here's the means of undoing a `\MakeShortVerb`, for instance in a region where you need to use the character outside a verbatim environment. It arranges for `\dospecials` and `\@sanitiz` to be altered appropriately, restores the saved catcode and, if necessary, the character's meaning (as stored by

`\MakeShortVerb`). If the catcode wasn't stored in `\cc\<c>` (by `\MakeShortVerb`) the command is silently ignored.

```
11598 \def\DeleteShortVerb#1{%
11599   \expandafter\ifx\csname cc\string#1\endcsname\relax
11600   \else
11601     \@shortvrbinf{Deleted }{#1 as}%
11602     \rem@special{#1}%
11603     \global\catcode'#1\csname cc\string#1\endcsname
```

We must not forget to reset `\cc\<c>`, otherwise the check in `\MakeShortVerb` for a repeated definition will not work.

```
11604   \global \expandafter\let \csname cc\string#1\endcsname \relax
11605   \ifnum\catcode'#1=\active
11606     \begingroup
11607       \catcode'\~\active \lccode'\~'#1%
11608       \lowercase{%
11609         \global\expandafter\let\expandafter~%
11610         \csname ac\string#1\endcsname}%
11611     \endgroup \fi \fi}
```

`\@shortvrbinf` Helper function for info messages.

```
11612 \def\@shortvrbinf#1#2{%
11613   \ClassInfo{memoir}{%
11614     #1\expandafter\@gobble\string#2 a short reference
11615     for \string\verb}}
```

`\add@special` This helper macro adds its argument to the `\dospecials` macro which is conventionally used by verbatim macros to alter the catcodes of the currently active characters. We need to add `\do\<c>` to the expansion of `\dospecials` after removing the character if it was already there to avoid multiple copies building up should `\MakeShortVerb` not be balanced by `\DeleteShortVerb` (in case anything that uses `\dospecials` cares about repetitions).

```
11616 \def\add@special#1{%
11617   \rem@special{#1}%
11618   \expandafter\gdef\expandafter\dospecials\expandafter
11619   {\dospecials \do #1}%
```

Similarly we have to add `\@makeother\<c>` to `\@sanitize` (which is used in things like `\index` to re-catcode all special characters except braces).

```
11620   \expandafter\gdef\expandafter\@sanitize\expandafter
11621   {\@sanitize \@makeother #1}}
```

`\rem@special` The inverse of `csadd@special` is slightly trickier. `\do` is re-defined to expand to nothing if its argument is the character of interest, otherwise to expand simply to the argument. We can then re-define `\dospecials` to be the expansion of itself. The space after `=##1` prevents an expansion to `\relax`!

```
11622 \def\rem@special#1{%
11623   \def\do##1{%
```

```

11624 \ifnum'#1='##1 \else \noexpand\do\noexpand##1\fi}%
11625 \xdef\dospecials{\dospecials}%

```

Fixing \@sanitize is the same except that we need to re-define \@makeother which obviously needs to be done in a group.

```

11626 \begingroup
11627 \def\@makeother##1{%
11628 \ifnum'#1='##1 \else \noexpand\@makeother\noexpand##1\fi}%
11629 \xdef\@sanitize{\@sanitize}%
11630 \endgroup

```

END OF MITTELBAACH CODE AND COMMENTARY.

30.4 General verbatim boxing and line numbering

A while ago I wrote a package that I never released. Here it is now, updated and improved. The package was based on code originally posted to CTT by Donald Arseneau on 13 July 2000.

This is DA's posted code.

```

\RequirePackage{verbatim}
\def\boxverbflag{14 }

\def\boxedverbatim{%
  \fboxsep=1em
  \def\verbatim@processline{\leavevmode
    \vrule\vbox{\advance\hsize-.8p@ \@@line
      {\strut\kern\fboxsep\the\verbatim@line\hss}%
      \kern\fboxsep}\vrule\par}%
  \@verbatim % but make some replacement settings
% \ifdim\@totalleftmargin>\fboxsep \fboxsep\@totalleftmargin \fi
  \leftskip\x@skip \rightskip\z@skip
  \interlinepenalty\boxverbflag
  \parfillskip\z@ plus\p@ minus\p@
  \lineskip-\fboxsep \baselineskip\z@skip
  \frenchspacing\@vobeyspaces
  \boxverb@toprule
  \verbatim@start}

\def\endboxedverbatim{\hrule\endtrivlist}

\@namedef{boxedverbatim*}{\let\frenchspacing\@gobble \boxedverbatim}
\@namedef{endboxedverbatim*}{\hrule\endtrivlist}

\output=\expandafter{\expandafter\boxverb@split \the\output}

\def\boxverb@toprule{\hrule \nobreak \vskip-.1\p@
  \@@line{\vrule height2\fboxsep \hss \vrule}}

\def\boxverb@split{\ifnum\outputpenalty=\boxverbflag

```

```

\ifdim\dp\@cclv=\z@
\setbox\@cclv\vbox{\unvbox\@cclv\hrule\kern-.4pt}%
\null \kern-.7\topskip \boxverb@toprule
\fi\fi}

```

That's the end of DA's posted code
I have extended this to provide:

- A (multipage) boxed verbatim
- A (multipage) boxed verbatim input
- Normally each 'page' is boxed, but start/end rules can be switched off at page boundaries
- A heading can be put at the start of 'continuation' pages
- Lines of verbatim text can be numbered
- Can be 'downgraded' to normal verbatim environment

OK, on to the main code.

```

\boxverbflag
\bvboxsep 11631 \def\boxverbflag{14 }
          11632 \newlength{\bvboxsep}      % user can change this
          11633 \setlength{\bvboxsep}{1em}
          11634

          11635 \newif\ifbvperpage % start/end lines on every page of multipage verbatim
          11636 \bvperpagetrue
          11637

\bvtopofpage Can use \bvtopofpage to put a heading above continued verbatims. For example
\begin{bvt} \bvtopofpage{\begin{center}\normalfont (Continued)\end{center}}
It only works for \bvperpagetrue.
          11638 \newcommand{\bvtopofpage}[1]{%
          11639   \long\def\bvttop{#1}}
          11640 \def\bvttop{} % used in \boxverb@split for heading
          11641

\c@memfbvline Counter for adjusting the starting line number for boxed verbatims and for the
\c@bvlinectr line number itself..
          11642 \newcounter{memfbvline}
          11643 \c@memfbvline=\z@
          11644 \newcounter{bvlinectr}

```

`\setbvlینums` `\setbvlینums{<firstline>}{<startnumsat>}` sets the first line number to `<firstline>` and the first line number to be printed is `<startnumsat>`.

```
11645 \newcommand*{\setbvlینums}[2]{%
11646   \c@bvlینectr #1\relax \advance\c@bvlینectr \m@ne
11647   \ifnum\z@<\linemodnum%   we are printing line numbers
11648     \@tempcnta #2\relax
11649     \divide\@tempcnta\linemodnum
11650     \multiply\@tempcnta\linemodnum
11651     \c@memfbvline #2\relax
11652     \advance\c@memfbvline-\@tempcnta
11653   \fi}
11654
```

`\theb@vlinenumber` `\resetbvlینumber` reinitializes the line numbering.

```
\resetbvlینumber11655 \def\theb@vlinenumber{\getthelinenumber{bvlینectr}{memfbvline}}
11656 \newcommand*{\resetbvlینumber}{\setcounter{bvlینectr}{0}}
11657
```

`\b@vdocount` Increment line number if counting. A line number is printed in a space width

`\bvnumlength` `\bvnumlength`, which is given a temporary value here.

```
11658 \def\b@vdocount{\ifbvcounlines\stepcounter{bvlینectr}\fi}
11659 \newlength{\bvnumlength}
11660 %% \settowidth{\bvnumlength}{\vlvnumfont 9999}
11661 \settowidth{\bvnumlength}{\normalfont 999}
11662
```

`\ifbvcountryside` Flag and commands for positioning the numbers. Default is to print them inside

`\bvnumbersinside` the box.

```
\bvnumbersoutside11663 \newif\ifbvcountryside % TRUE if line numbers inside box
11664   \bvcountryside true
11665 \newcommand*{\bvnumbersinside}{\bvcountryside true}
11666 \newcommand*{\bvnumbersoutside}{\bvcountryside false}
11667
```

`\b@vdoinside` Print numbers inside (outside) the box.

```
\b@vdooutside11668 \def\b@vdoinside{%
11669   \ifbvcounlines\ifbvcountryside%
11670     \makebox[\bvnumlength][r]{%
11671       \vlvnumfont \theb@vlinenumber\space}%
11672   \fi\fi}
11673
11674 \def\b@vdooutside{%
11675   \ifbvcounlines\ifbvcountryside\else%
11676     \llap{\makebox[\bvnumlength][r]{%
11677       \vlvnumfont \theb@vlinenumber\space}}%
11678   \fi\fi}
11679
```

```

\@@m@oline \@@line is defined as \def\@@line{\hb@xt@\hsize} and I need an equivalent
for \linewidth instead of \hsize for use in \setupboxverb@line.
11680 \newcommand*\@@m@oline{\hb@xt@\linewidth}
11681

\setupboxverb@line Use a macro for the first main part of DA's code for \boxedverbatim, adding
lots of hooks.
Per Starbäck had problems with the combination of adjustwidth with
boxedverbatim (see CTT Re: [memoir] adjustwidth + boxedverbatim,
2007/02/10) and Lars Madsen came up with a suggestion to use \linewidth
instead of \hsize.
11682 \newcommand{\setupboxverb@line}{%
11683   \par
11684   \ifbvperpage
11685     \output=\expandafter{\expandafter\boxverb@split \the\output}
11686   \fi

\verbatim@processline
11687   \def\verbatim@processline{\leavevmode
11688     \b@vdocount%
11689     \b@vleftsidehook\vb@x{\advance% \hsize-.8\p@ \@@line % changed to \linewidth
11690                                   \linewidth-.8\p@ \@@line
11691       {\b@vdooutside\strut\kern\bvboxsep%
11692         \b@vdoinside%
11693         \ift@bs
11694           \tabverbatim@processline
11695         \else
11696           \the\verbatim@line
11697         \fi
11698         \hss}%
11699     \kern\bvboxsep}\b@vrightsidehook\par}}
11700

\setupbox@verb Use a macro for the second main part of DA's code, and integrate it with the
other verbatim codes (e.g., include \@maybeobeytags).
11701 \newcommand{\setupbox@verb}{%
11702   \leftskip\z@skip \rightskip\z@skip
11703   \interlinepenalty\boxverbflag
11704   \parfillskip\z@ plus\p@ minus\p@
11705   \lineskip-\bvboxsep \baselineskip\z@skip
11706   \frenchspacing\@vobeyspaces\@maybeobeytabs
11707   \boxverb@toprule}
11708

\boxedverbatim Given the two macros above, we can write a briefer version of DA's
\endboxedverbatim \boxedverbatim(*). As noted by Zarko Cucej30, have to handle a following
\boxedverbatim* (no-) paragraph.
\endboxedverbatim*

```

³⁰zarko.cucej@uni-mb.si


```

11709 \def\boxedverbatim{\begingroup
11710   \let\@line\@m@line%      new from mempatch v4.9
11711   \setupboxverb@line
11712   \@verbatim
11713   \setupbox@verb
11714   \verbatim@start}
11715 \def\endboxedverbatim{\bvendrulehook\endtrivlist\endgroup\@doendpe}
11716 \@namedef{boxedverbatim*}{\let\frenchspacing\@gobble \boxedverbatim}
11717 \@namelet{endboxedverbatim*}\endboxverbatim
11718
11719 \def\boxverb@toprule{\bvtoprulehook
11720   \@line{\bvleftsidehook \bvtopmidhook \bvrighsidehook}}
11721

```

`\bvendofpage` `\bvendofpage{<spec>}` sets the boxed verbatim ‘end of page’ to `<spec>`.

```

11722 \newcommand*{\bvendofpage}[1]{%
11723   \def\boxverb@botpage{#1}}
11724 %%%\bvendofpage{\hrule\kern-.4pt}
11725 \bvendofpage{\hrule width\linewidth\kern-.4pt}
11726

```

`\boxverb@split`

```

11727 \def\boxverb@split{\ifnum\outputpenalty=\boxverbflag
11728   \ifdim\dp\@cclv=\z@
11729   %%% \setbox\@cclv\vbox{\unvbox\@cclv\hrule\kern-.4pt}%
11730   \setbox\@cclv\vbox{\unvbox\@cclv\boxverb@botpage}%
11731   \null \kern-.7\topskip \b@vtop \boxverb@toprule
11732   \fi
11733 \fi}
11734

```

`\bvtoprulehook` The new hooks, for the top, bottom, left and right of the box.

```

\bvendrulehook 11735 \def\bvtoprulehook{\hrule width\linewidth \nobreak\vskip-.1p@}
\bvleftsidehook 11736 \def\bvendrulehook{\hrule width\linewidth}
\bvrighsidehook 11737 \def\bvleftsidehook{\vrule}
\bvtopmidhook 11738 \def\bvrighsidehook{\vrule}
11739 \def\bvtopmidhook{\rule{0p@}{2\bvboxsep} \hss}
11740

```

`\boxedverbatiminput` `\boxedverbatiminput{<filename>}` read in filename contents as verbatim

```

\boxedverbatim@input 11741 \newcommand{\boxedverbatiminput}{\begingroup
11742   \@ifstar{\let\frenchspacing\@gobble
11743     \boxedverbatim@input\relax}%
11744   {\boxedverbatim@input{\frenchspacing\@vobeyspaces}}}
11745
11746 \def\boxedverbatim@input#1#2{%
11747   \setupboxverb@line
11748   \IfFileExists{#2}{\@verbatim #1\relax

```

```

11749 \setupbox@verb
11750 \verbatim@readfile{\@filef@und}%
11751 \bvendrulehook\endtrivlist\endgroup\@doendpe}%
11752 {\typeout {No file #2.}\endgroup}}
11753

```

Some prepackaged boxing styles.

`\bvbox` Original (default) boxing.

```

11754 \newcommand{\bvbox}{%
11755 \bvperpagetrue%
11756 \renewcommand{\bvtoprulehook}{\hrule \nobreak \vskip-.1\p@}%
11757 \renewcommand{\bvleftsidehook}{\vrule}%
11758 \renewcommand{\bvrightsidehook}{\vrule}%
11759 \renewcommand{\bvendrulehook}{\hrule}}
11760

```

`\nobvbox` No boxing

```

11761 \newcommand{\nobvbox}{%
11762 \bvperpagefalse%
11763 \renewcommand{\bvtoprulehook}{}%
11764 \renewcommand{\bvleftsidehook}{}%
11765 \renewcommand{\bvrightsidehook}{}%
11766 \renewcommand{\bvendrulehook}{}%
11767

```

`\bvtopandtail` Head and foot horizontal lines only

```

11768 \newcommand{\bvtopandtail}{%
11769 \bvperpagefalse%
11770 \renewcommand{\bvtoprulehook}{\hrule \nobreak \vskip-.1\p@}%
11771 \renewcommand{\bvleftsidehook}{}%
11772 \renewcommand{\bvrightsidehook}{}%
11773 \renewcommand{\bvendrulehook}{\hrule}}
11774

```

`\bvsides` Side vertical lines only

```

11775 \newcommand{\bvsides}{%
11776 \bvperpagefalse%
11777 \renewcommand{\bvtoprulehook}{\vskip 3ex}%
11778 \renewcommand{\bvleftsidehook}{\vrule}%
11779 \renewcommand{\bvrightsidehook}{\vrule}%
11780 \renewcommand{\bvendrulehook}{}%
11781

```

30.5 The framed package

The following code is the framed package [Ars03] by Donald Arseneau.

```

\framed
\endframed1782 \let\framed\relax \let\endframed\relax
\shaded1783 \let\shaded\relax \let\endshaded\relax
\endshaded1784

```

Way back when DA sent me a pre-release copy of v0.6 of his framed package [Ars03]. We also discussed good ways of embedding it into the class. One result being that I defined some items `\AtBeginDocument`, but this is no longer required as DA has changed framed to cooperate with memoir. We continue to have sporadic email conversations about the interaction of memoir and framed.

Here is a modified version of framed.sty. In particular I moved some of DA's description of the code into the general commentary instead of comments within the code.

```

framed.sty  v 0.95    2007/10/04
Copyright (C) 1992-2007 by Donald Arseneau (asnd@triumf.ca)
These macros may be freely transmitted, reproduced, or modified
provided that this notice is left intact.

```

===== Begin Instructions =====

```

framed.sty
~~~~~

```

Create framed, shaded, or differently highlighted regions that can break across pages. The environments defined are

```

framed    -- ordinary frame box (\fbox) with edge at margin
shaded    -- shaded background (\colorbox) bleeding into margin
snugshade -- similar
leftbar   -- thick vertical line in left margin
to be used like
\begin{framed}
copious text
\end{framed}

```

But the more general purpose of this package is to facilitate the creation of environments that enable page breaking within arbitrary decorations using a simple new-environment definition incorporating `\FrameCommand` and `\begin{MakeFramed}{settings} ... \end{MakeFramed}`

The "framed" environment uses `"\fbox"` as its `"\FrameCommand"` with the additional settings `\fboxrule=\FrameRule` and `\fboxsep=\FrameSep`. You can change these lengths (using `\setlength`) and you can change the definition of `\FrameCommand` to use much fancier boxes.

In fact, the "shaded" environment just redefines `\FrameCommand` to be `\colorbox{shadecolor}` (and you have to define the color "shadecolor": `\definecolor{shadecolor}...`).

A page break is allowed, and even encouraged, before the framed environment. If you want to attach some text (a box title) to the frame, then the text should be inserted by `\FrameCommand`.

The contents of the framed regions are restricted:
 Floats, footnotes, marginpars and head-line entries will be lost.
 (Some of these may be handled in a later version.)
 This package will not work with the page breaking of `multicol.sty`,
 or other systems that perform column-balancing.

The `MakeFramed` environment does the work. Its "settings" argument should contain any adjustments to the text width (applied to `\hsize`, and using the "`\width`" of the frame itself) as well as a "restore" command -- `\@parboxrestore` or `\FrameRestore` or something similar; as an example, the `snugshade` environment shows how to suppress excess spacing within the box, copying the code from `minipage`.

Expert commands:

```
\MakeFramed, \endMakeFramed: the "MakeFramed" environment
\FrameCommand: command to draw the frame around its argument
\FrameRestore: restore some text settings, but fewer than \@parboxrestore
\FrameRule: length register; \fboxrule for default "framed".
\FrameSep: length register; \fboxsep for default "framed".
\FrameHeightAdjust: macro; height of frame above baseline at top of page
```

This is still a 'pre-production' version because I can think of many features/improvements that should be made. Nevertheless, starting with version 0.5 it should be bug-free.

ToDo:

```
Test more varieties of list
Improve and correct documentation
Propagation of \marks
Handle footnotes (how??) floats (?) and marginpars.
Stretchability modification.
```

===== End Instructions =====

```
\ProvidesPackage{framed}[2007/10/04 v 0.95:
  framed or shaded text with page breaks]
```

```
\newenvironment{framed}% using default \FrameCommand
  {\MakeFramed {\advance\hsize-\width \FrameRestore}}%
  {\endMakeFramed}
```

```
\newenvironment{shaded}{%
  \def\FrameCommand{\fboxsep=\FrameSep \colorbox{shadecolor}}%
  \MakeFramed {\FrameRestore}}%
  {\endMakeFramed}
```

```

\newenvironment{leftbar}{%
  \def\FrameCommand{\vrule width 3pt \hspace{10pt}}%
  \MakeFramed {\advance\hsize-\width \FrameRestore}}%
{\endMakeFramed}

\newenvironment{snugshade}{%
  \def\FrameCommand{\colorbox{shadecolor}}%
  \MakeFramed {\FrameRestore\@setminipage}}%
{\par\unskip\endMakeFramed}

```

Now for the package code.

```

11785 \chardef\FrameRestore=\catcode'\ | % for debug
11786 \catcode'\ |=\catcode'\% % (debug: insert space after backslash)
11787

\MakeFramed
11788 \def\MakeFramed#1{\par
  measure added width and height; call result \width and \height
11789 \fb@sizeofframe\FrameCommand
11790 \let\width\fb@frw \let\height\fb@frh

  insert pre-penalties and skips
11791 \begingroup
11792 \skip@\lastskip
11793 \if@nobreak\else
11794   \penalty9999 % updates \page parameters
11795   \ifdim\pagefilstretch=\z@ \ifdim\pagefillstretch=\z@

    not infinitely stretchable, so encourage a page break here
11796     \edef\@tempa{\the\skip@}%
11797     \ifx\@tempa\zero@glue \penalty-30
11798     \else \vskip-\skip@ \penalty-30 \vskip\skip@
11799     \fi\fi\fi
11800     \penalty\z@

    Give a stretchy breakpoint that will always be taken in preference to the
    \penalty 9999 used to update page parameters. The cube root of 10000/100
    indicates a multiplier of 0.21545, but the maximum calculated badness is really
    8192, not 10000, so the multiplier is 0.2301.
11801     \advance\skip@ \z@ plus-.5\baselineskip
11802     \advance\skip@ \z@ plus-.231\height
11803     \advance\skip@ \z@ plus-.231\skip@
11804     \advance\skip@ \z@ plus-.231\topsep
11805     \vskip-\skip@ \penalty 1800 \vskip\skip@
11806   \fi
11807   \addvspace{\topsep}%
11808 \endgroup

```

clear out pending page break

```

11809 \penalty\@M \vskip 2\baselineskip \vskip\height
11810 \penalty9999 \vskip -2\baselineskip \vskip-\height
11811 \penalty9999 % updates \pagetotal
11812 \message{After clearout, \pagetotal=\the\pagetotal,
11813 | \pagegoal=\the\pagegoal. }%
11814 \fb@adjheight
11815 \setbox\@tempboxa\vbox\bgroup
11816 #1% Modifications to \hsize (can use \width and \height)
11817 \textwidth\hsize \columnwidth\hsize}
11818

```

\endMakeFramed

```

11819 \def\endMakeFramed{\par
11820 \kern\z@
11821 \hrule\@width\hsize\@height\z@
11822 \penalty-100 % put depth into height
11823 \egroup
11824 % {\showoutput\showbox\@tempboxa}%
11825 \begingroup
11826 \fb@put@frame\FrameCommand\FirstFrameCommand
11827 \endgroup}
11828

```

\fb@put@frame \fb@put@frame{<nosplit>}{<split>} takes the contents of \@tempboxa and puts all, or a piece, of it on the page with a frame (\FrameCommand, \FirstFrameCommand, \MidFrameCommand, or \LastFrameCommand). It recurses until all of \@tempboxa has been used up. (\@tempboxa must have zero depth.)
 <nosplit> = attempted framing command, if no split
 <split> = framing command if split.
 First iteration: Try to fit with \FrameCommand. If it does not fit, split for \FirstFrameCommand.
 Later iteration: Try to fit with \LastFrameCommand. If it does not fit, split for \MidFrameCommand.

```

11829 \def\fb@put@frame#1#2{\relax
11830 \ifdim\pagegoal=\maxdimen \pagegoal\vsize \fi
11831 \ifinner
11832 \fb@putboxa#1%
11833 \fb@afterframe
11834 \else
11835 \dimen@\pagegoal \advance\dimen@-\pagetotal % natural space left on page
11836 \ifdim\dimen@<2\baselineskip % Too little room on page
11837 | \message{Page has only \the\dimen@\space room left; eject. }%
11838 \eject \fb@adjheight \fb@put@frame#1#2%
11839 \else % there's appreciable room left on the page
11840 \fb@sizeofframe#1%
11841 | \message{\string\pagetotal=\the\pagetotal,
11842 | \string\pagegoal=\the\pagegoal,

```

```

11843 |         \string\pagestretch=\the\pagestretch,
11844 |         \string\pageshrink=\the\pageshrink,
11845 |         \string\fb@frh=\fb@frh. \space}
11846 |     \message{Box of size \the\ht\@tempboxa\space + \fb@frh}%
11847 |     \begingroup % temporarily set \dimen@ to be...
11848 |     \advance\dimen@.8\pageshrink % maximum space available on page
11849 |     \advance\dimen@-\fb@frh\relax % space available for frame's contents
11850 |     \expandafter\endgroup

    expand \ifdim, then restore \dimen@ to real room left on page
11851 |     \ifdim\dimen@>\ht\@tempboxa % whole box does fit
11852 |         \message{fits in \the\dimen@. }%

    Use vsplit anyway to capture the marks
    !!!???!!! MERGE THIS WITH THE else CLAUSE!!!
11853 |         \fb@putboxa#1%
11854 |         \fb@afterframe
11855 |     \else % box must be split
11856 |         \message{must be split to fit in \the\dimen@. }%

    update frame measurement to use \FirstFrameCommand or \MidFrameCommand
11857 |         \fb@sizeofframe#2%
11858 |         \setbox\@tempboxa\vbox{% simulate frame and flexibility of the page:
11859 |             \vskip \fb@frh \@plus\pagestretch \@minus.8\pageshrink
11860 |             \kern137sp\kern-137sp\penalty-30
11861 |             \unvbox\@tempboxa}%
11862 |         \edef\fb@resto@set{\boxmaxdepth\the\boxmaxdepth
11863 |             \splittopskip\the\splittopskip}%
11864 |         \boxmaxdepth\z@ \splittopskip\z@
11865 |         \message{Padded box of size \the\ht\@tempboxa\space split
11866 |             to \the\dimen@}%

    Split box here
11867 |         \setbox\@tw@ \vsplit\@tempboxa to\dimen@
11868 |         \toks99\expandafter{\splitfirstmark}%
11869 |         \toks98\expandafter{\splitbotmark}%
11870 |         \message{Marks are: \the\toks99, \the\toks98. }%
11871 |         \setbox\@tw@ \vbox{\unvbox\@tw@}% natural-sized
11872 |         \message{Natural height of split box is \the\ht\@tw@, leaving
11873 |             \the\ht\@tempboxa\space remainder. }%

    If the split-to size > (\vsize-\topskip), then set box to full size
11874 |         \begingroup
11875 |         \advance\dimen@\topskip
11876 |         \expandafter\endgroup
11877 |         \ifdim\dimen@>\pagegoal
11878 |             \message{Frame is big -- Use up the full column. }%
11879 |             \dimen@ii\pagegoal
11880 |             \advance\dimen@ii -\topskip
11881 |             \advance\dimen@ii \FrameHeightAdjust\relax
11882 |         \else % suspect this is wrong:

```

If the split-to size > feasible room-on-page, rebox it smaller.

```

11883      \advance\dimen@.8\pageshrink
11884      \ifdim\ht\tw@>\dimen@
11885 |      \message{Box too tall; rebox it to \the\dimen@. }%
11886      \dimen@ii\dimen@
11887      \else % use natural size
11888      \dimen@ii\ht\tw@
11889      \fi
11890      \fi

```

Re-box contents to desired size \dimen@ii

```

11891      \advance\dimen@ii -\fb@frh
11892      \setbox\tw@\vbox to\dimen@ii \bgroup

```

remove simulated frame and page flexibility:

```

11893      \vskip -\fb@frh \@plus-\pagestretch \@minus-.8\pageshrink
11894      \unvbox\tw@ \unpenalty\unpenalty
11895      \ifdim\lastkern=-137sp % whole box went to next page
11896 |      \message{box split at beginning! }%
11897      % need work here???
11898      \egroup \fb@resto@set \eject % (\vskip for frame size was discarded)
11899      \fb@adjheight
11900      \fb@put@frame#1#2% INSERTED ???
11901      \else % Got material split off at the head
11902      \egroup \fb@resto@set
11903      \ifvoid@tempboxa % it all fit after all
11904 |      \message{box split at end! }%
11905      \setbox\@tempboxa\box\tw@
11906      \fb@putboxa#1%
11907      \fb@afterframe
11908      \else % it really did split
11909 |      \message{box split as expected. Its reboxed height
11910 |      is \the\ht\tw@. }%
11911      \ifdim\wd\tw@>\z@
11912      \wd\tw@\wd\@tempboxa

```

PW: the next line was:

```
\centerline{#2{\box\tw@}}% ??
```

b

ad

idea?

but I have changed it as follows so that \centerline can be changed from elsewhere if useful.

```

11913      \memfblineboxtwo{#2{\box\tw@}}% ??? \centerline bad idea? \
11914      \else
11915 |      \message{Zero width means likely blank.
11916 |      Don't frame it (guess)}%
11917      \box\tw@
11918      \fi
11919      \hrule \@height\z@ \@width\hsize
11920      \eject

```



```

11921             \fb@adjheight
11922             \fb@put@frame\LastFrameCommand\MidFrameCommand
11923 \fi\fi\fi\fi\fi}
11924

\memfblineboxtwo PW: this is an addition of mine. There is a question mark against the use of
\memfblineboxa \centerline within the framed code. I have created these two macros which are
used in the code instead of \centerline, so that those parts of the code can be
easily altered. are used instead

11925 \newcommand{\memfblineboxtwo}[1]{\centerline{#1}}
11926 \newcommand{\memfblineboxa}[1]{\centerline{#1}}
11927

\fb@putboxa
11928 \def\fb@putboxa#1{%
11929 \ifvoid\@tempboxa
11930 %%% PackageWarning{framed}{Boxa is void -- discard it. }%
11931 \@memwarn{Boxa is void -- discard it. }%
11932 \else
11933 | \message{Frame and place boxa. }%
11934 | %{\showoutput\showbox\@tempboxa}%
11935 %%%\centerline{#1{\box\@tempboxa}}%
11936 \memfblineboxa{#1{\box\@tempboxa}}%
11937 \fi}
11938

\fb@afterframe
11939 \def\fb@afterframe{%
11940 \nointerlineskip \null %{\showoutput \showlists}
11941 \penalty-30 \vskip\topsep \relax}
11942

\fb@frw \fb@sizeofframe{<framecommand>} measures width and height added by frame
\fb@frh (<framecommand>) and call the results \fb@frw and \fb@frh.
\fb@sizeofframe 11943 \newdimen\fb@frw
11944 \newdimen\fb@frh
11945 \def\fb@sizeofframe#1{\begingroup
11946 \setbox\z@\vbox{\vskip-5in \hbox{\hskip-5in
11947 #1{\hbox{\vrule \@height 4.7in \@depth.3in \@width 5in}}}%
11948 \vskip\z@skip}%
11949 | \message{Measuring frame addition for \string#1 in \@currentenv\space
11950 | gives ht \the\ht\z@\space and wd \the\wd\z@. }%
11951 %{\showoutput\showbox\z@}%
11952 \global\fb@frw\wd\z@ \global\fb@frh\ht\z@
11953 \endgroup}
11954

\fb@adjheight
11955 \def\fb@adjheight{%

```

```

11956 \vbox to\FrameHeightAdjust{}% get proper baseline skip from above.
11957 \penalty\@M \nointerlineskip
11958 \vskip-\FrameHeightAdjust
11959 \penalty\@M} % useful for tops of pages
11960

\zero@glue
11961 \edef\zero@glue{\the\z@skip}
11962
11963 \catcode'\|=\FrameRestore
11964

\FrameCommand Provide configuration commands:
\FrameRule 11965 \providecommand\FrameCommand{%
\FrameSep 11966 \setlength\fbboxrule{\FrameRule}\setlength\fbboxsep{\FrameSep}%
\FirstFrameCommand 11967 \fbbox}
\MidFrameCommand 11968 \@ifundefined{FrameRule}{\newdimen\FrameRule \FrameRule=\fbboxrule}{}
\LastFrameCommand 11969 \@ifundefined{FrameSep}{\newdimen\FrameSep \FrameSep =3\fbboxsep}{}
11970 \providecommand\FirstFrameCommand{\FrameCommand}
11971 \providecommand\MidFrameCommand{\FrameCommand}
11972 \providecommand\LastFrameCommand{\FrameCommand}

\FrameHeightAdjust Height of frame above first baseline when frame starts a page:
11973 \providecommand\FrameHeightAdjust{6pt}
11974

\FrameRestore \FrameRestore has parts of \@parboxrestore, performing a similar but less
complete restoration of a default layout. See how it is used in the "settings"
argument of \MakeFrame. Though not a parameter, \hsize should be set to the
desired total line width available inside the frame before invoking
\FrameRestore.
11975 \def\FrameRestore{%
11976 \let\if@nobreak\iffalse
11977 \let\if@noskipsec\iffalse
11978 \let\-\@dischyph
11979 \let'\@acci\let'\@accii\let\=\@acciii
11980 % \message{FrameRestore:
11981 % \totalleftmargin=\the \totalleftmargin,
11982 % \rightmargin=\the\rightmargin,
11983 % \listdepth=\the\listdepth. }%

Test if we are in a list (or list-like paragraph)
11984 \ifnum \ifdim\totalleftmargin>\z@ 1\fi
11985 \ifdim\rightmargin>\z@ 1\fi
11986 \ifnum\listdepth>0 1\fi 0>\z@
11987 % \message{In a list: \linewidth=\the\linewidth,
11988 % \totalleftmargin=\the\totalleftmargin,
11989 % \parshape=\the\parshape, \columnwidth=\the\columnwidth,
11990 % \hsize=\the\hsize,
11991 % \labelwidth=\the\labelwidth. }%

```

Now try to propagate changes of width from `\hsize` to list parameters. This is deficient, but a more advanced way to indicate modification to text dimensions is not (yet) provided; in particular, no separate left/right adjustment. (PW. I have provided a macro for the next few lines as I think that I may have a reason to allow them to be easily changed).

```

11992 %%      \@setminipage % snug fit around the item
11993 %%      \advance\linewidth-\columnwidth \advance\linewidth\hsize
11994 %%      \parshape\@ne \@totalleftmargin \linewidth
11995      \memfblistfixparams
11996      \else % Not in list
11997          \linewidth=\hsize
11998          %\message{No list, set \string\linewidth=\the\hsize. }%
11999      \fi
12000      \sloppy}
12001
12002 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
12003

```

`\memfblistfixparams` An extract of some code from `\FrameRestore`. Try to propagate changes of width from `\hsize` to list parameters. This is deficient, but a more advanced way to indicate modification to text dimensions is not (yet) provided; in particular, no separate left/right adjustment.

```

12004 \newcommand*{\memfblistfixparams}{%
12005     \@setminipage % snug fit around the item
12006     \advance\linewidth-\columnwidth \advance\linewidth\hsize
12007     \parshape\@ne \@totalleftmargin \linewidth}
12008

```

I have moved the following from the start of the framed package to here.

```

framed
shaded 2009 \newenvironment{framed}% using default \FrameCommand
leftbar 2010 {\MakeFramed {\advance\hsize-\width \FrameRestore}}%
snugshade 2011 {\endMakeFramed}
12012
12013 \newenvironment{shaded}{%
12014     \def\FrameCommand{\fboxsep=\FrameSep \colorbox{shadecolor}}%
12015     \MakeFramed {\FrameRestore}}%
12016     {\endMakeFramed}
12017
12018 \newenvironment{leftbar}{%
12019     \def\FrameCommand{\vrule width 3pt \hspace{10pt}}%
12020     \MakeFramed {\advance\hsize-\width \FrameRestore}}%
12021     {\endMakeFramed}
12022
12023 \newenvironment{snugshade}{%
12024     \def\FrameCommand{\colorbox{shadecolor}}%
12025     \MakeFramed {\FrameRestore\@setminipage}}%
12026     {\par\unskip\endMakeFramed}

```

12027

As the package used `\newenvironment` it will burp if it is used with the class. There are two options: either prevent `framed` from being loaded, or kill the environments if it is loaded. I'm going for the latter as Donald will be updating his package at some point and author's may want to use it before it's updated here.

```
12028 \AtBeginPackage{framed}{%
12029   \let\framed\relax \let\endframed\relax
12030   \let\shaded\relax \let\endshaded\relax
12031   \let\leftbar\relax \let\endleftbar\relax
12032   \let\snugshade\relax \let\endsnugshade\relax}
12033
```

`qframe` This is from an email to me from DA shortly after he released `framed` v 0.95.

A frame more like a simple parbox. Notably useful for use inside a single list item — the frame matches the text of the other items.

He actually called it `pframe` and said that something similar could be in a later version of `framed`.

I have found it useful for framing `quote` or `adjustwidth` environments, or similar. Hence the name, from ‘*quote frame*’.

```
12034 \newenvironment{qframe}{%
12035   \def\FrameCommand##1{\fboxrule=\FrameRule\fboxsep=\FrameSep
12036     \hskip\@totalleftmargin\fbox{##1}% There is no \@totalrightmargin, so...
12037     \hskip-\linewidth \hskip-\@totalleftmargin \hskip\columnwidth}%
12038   \MakeFramed{\advance\hsize-\width
12039     \advance\hsize \FrameSep
12040     \@totalleftmargin\z@ \linewidth=\hsize}}%
12041   {\endMakeFramed}
12042
```

A nd here is my corresponding version for `shaded`.

```
12043 \newenvironment{qshade}{%
12044   \def\FrameCommand##1{\fboxsep=\FrameSep
12045     \hskip\@totalleftmargin
12046     \hskip -1\FrameSep
12047     \colorbox{shade color}{##1}%
12048     \hskip-\linewidth \hskip-\@totalleftmargin
12049     \hskip -1\FrameSep
12050     \hskip\columnwidth}%
12051   \MakeFramed{\advance\hsize-\width
12052     \advance\hsize 3\FrameSep
12053     \@totalleftmargin\z@ \linewidth=\hsize}}%
12054   {\endMakeFramed}
12055
```

30.6 The newfile package

The following code is from the newfile package. To try and avoid name clashes with other packages, each internal macro in this package includes the character string ‘stre@m’.

\newoutputstream **\newoutputstream{<stream>}** creates a new output stream called <stream>. Different files may be associated with the <stream>. Note that TeX permits no more than 16 output streams.

```

12056 \newcommand*{\newoutputstream}[1]{%
12057   \@ifundefined{#1outstre@m}%
12058     {\expandafter\newwrite\csname #1outstre@m\endcsname
12059      \csname newif\expandafter\endcsname
12060       \csname ifstre@m#1open\endcsname
12061       \global\csname stre@m#1openfalse\endcsname
12062       \expandafter\ifx\csname atstreamopen#1\endcsname\relax
12063         \global\@namedef{atstreamopen#1}{}%
12064       \fi
12065       \expandafter\ifx\csname atstreamclose#1\endcsname\relax
12066         \global\@namedef{atstreamclose#1}{}%
12067       \fi
12068     }%
12069     {\@memwarn{Output stream #1\space is already defined}}}
12070
```

\newinputstream **\newinputstream{<stream>}** creates a new input stream called <stream>. Different files may be associated with the <stream>. Note that TeX permits no more than 16 input streams.

```

12071 \newcommand*{\newinputstream}[1]{%
12072   \@ifundefined{#1instre@m}%
12073     {\expandafter\newread\csname #1instre@m\endcsname
12074      \csname newif\expandafter\endcsname
12075       \csname ifstre@m#1open\endcsname
12076       \global\csname stre@m#1openfalse\endcsname
12077       \expandafter\ifx\csname atstreamopen#1\endcsname\relax
12078         \global\@namedef{atstreamopen#1}{}%
12079       \fi
12080       \expandafter\ifx\csname atstreamclose#1\endcsname\relax
12081         \global\@namedef{atstreamclose#1}{}%
12082       \fi
12083     }%
12084     {\@memwarn{Input stream #1\space is already defined}}}
12085
```

Some checking macros will be useful as some of the checks occur in multiple places.

\IfStreamOpen **\IfStreamOpen{<stream>}{<TRUE code>}{<FALSE code>}** checks if stream <stream> is currently open.

```

12086 \newcommand{\IfStreamOpen}[3]{%
12087   \csname ifstre@m#1open\endcsname#2\else#3\fi

\instre@mandopen \instre@mandopen{<stream>}{<TRUE code>} checks if <stream> is an input
stream and is open. If so, it executes <TRUE code>.

12088 \newcommand{\instre@mandopen}[2]{%
12089   \ifundefined{#1instre@m}{%
12090     \@memwarn{#1\space is not an input stream}}%
12091     {\IfStreamOpen{#1}{#2}{%
12092       \@memwarn{Input stream #1\space is not open}}}%
12093

\instre@mandclosed \instre@mandclosed{<stream>}{<TRUE code>} checks if <stream> is an input
stream and is closed (not open). If so, it executes <TRUE code>.

12094 \newcommand{\instre@mandclosed}[2]{%
12095   \ifundefined{#1instre@m}{%
12096     \@memwarn{#1\space is not an input stream}}%
12097     {\IfStreamOpen{#1}{#2}{%
12098       \@memwarn{Input stream #1\space is open}}{#2}}}%
12099

\outstre@mandopen \outstre@mandopen{<stream>}{<TRUE code>} checks if <stream> is an output
stream and is open. If so, it executes <TRUE code>.

12100 \newcommand{\outstre@mandopen}[2]{%
12101   \ifundefined{#1outstre@m}{%
12102     \@memwarn{#1\space is not an output stream}}%
12103     {\IfStreamOpen{#1}{#2}{%
12104       \@memwarn{Output stream #1\space is not open}}}%
12105

\outstre@mandclosed \outstre@mandclosed{<stream>}{<TRUE code>} checks if <stream> is an output
stream and is closed (not open). If so, it executes <TRUE code>.

12106 \newcommand{\outstre@mandclosed}[2]{%
12107   \ifundefined{#1outstre@m}{%
12108     \@memwarn{#1\space is not an output stream}}%
12109     {\IfStreamOpen{#1}{#2}{%
12110       \@memwarn{Output stream #1\space is open}}{#2}}}%
12111

\openoutputfile \openoutputfile{<filename>}{<stream>} opens the file called <filename> and
attaches it to the stream <stream> for writing. However, if the \nofiles
command has been given the file is not attached to the stream. No more than
one file can be attached to a stream at any given time.

12112 \newcommand*\openoutputfile[2]{%
12113   \outstre@mandclosed{#2}{%
12114     \global\@namedef{#1@filename}{#1}%
12115     \if@files
12116       \immediate\openout\@nameuse{#2outstre@m}=\@nameuse{#1@filename}%

```

```

12117 \fi
12118 \global\csname stre@m#2openttrue\endcsname%
12119 \@nameuse{atstreamopen#2}}
12120

```

`\closeoutputstream` `\closeoutputstream{<stream>}` closes the stream `<stream>`.

```

12121 \newcommand*{\closeoutputstream}[1]{%
12122 \outstre@mmandopen{#1}{%
12123 \@nameuse{atstreamclose#1}%
12124 \immediate\closeout\@nameuse{#1outstre@m}%
12125 \global\csname stre@m#1openfalse\endcsname}}
12126

```

`\openinputfile` `\openinputfile{<filename>}{<stream>}` opens the file called `<filename>` and attaches it to the stream `<stream>` for reading. The file is added to the list of files. No more than one file can be attached to a stream at any given time.

```

12127 \newcommand{\openinputfile}[2]{%
12128 \IfFileExists{#1}{% file exists
12129 \instre@mmandclosed{#2}{%
12130 \@addtofilelist{#1}%
12131 \global\@namedef{#1@filename}{#1}%
12132 \immediate\openin\@nameuse{#2instre@m}=\@nameuse{#1@filename}%
12133 \global\csname stre@m#2openttrue\endcsname%
12134 \@nameuse{atstreamopen#2}}}%
12135 {% file not found
12136 \typeout{No file #1.}
12137 }}
12138

```

`\closeinputstream` `\closeinputstream{<stream>}` closes the stream `<stream>`.

```

12139 \newcommand{\closeinputstream}[1]{%
12140 \instre@mmandopen{#1}{%
12141 \@nameuse{atstreamclose#1}%
12142 \immediate\closein\@nameuse{#1instre@m}%
12143 \global\csname stre@m#1openfalse\endcsname}}
12144

```

`writeverbatim` `\begin{writeverbatim}{<stream>}` writes the contents of the environment as verbatim text to the given `<stream>`.

```

12145 \def\writeverbatim#1{%
12146 \@bsphack
12147 \let\do\@makeother\dospecials
12148 \catcode'\^~M\active
12149 \def\verbatim@processline{%
12150 \immediate\write\@nameuse{#1outstre@m}{\the\verbatim@line}}%
12151 \verbatim@start}
12152 \def\endwriteverbatim{\@esphack}
12153

```

`\addtostream` `\addtostream{<stream>}{<text>}` writes <text> to the given <stream>.

```

12154 \newcommand{\addtostream}[2]{%
12155   \@bsphack
12156   \outstre@mandopen{#1}{%
12157     {\let\protect\string
12158       \immediate\write\@nameuse{#1outstre@m}{#2}%
12159     }}%
12160   \@esphack}
12161

```

`\ifstre@mnoteof` `\checkstre@mnoteof{<stream>}` sets `\ifstre@mnoteof` to TRUE if <stream> is
`\checkstre@mnoteof` not at the end of the file (i.e., it is the opposite of `\ifeof`).

```

12162 \newif\ifstre@mnoteof
12163 \newcommand{\checkstre@meof}[1]{%
12164   \stre@mnoteoftrue\ifeof\@nameuse{#1instre@m}\stre@mnoteoffalse\fi}
12165

```

`\readstream` `\readstream{<stream>}` reads the contents of the given <stream> as `\input` text.

```

12166 \def\readstream#1{
12167   \instre@mandopen{#1}{%
12168     \loop \checkstre@meof{#1} \ifstre@mnoteof
12169       \read\@nameuse{#1instre@m} to\temptokstre@m
12170       \temptokstre@m
12171     \repeat
12172   }}
12173

```

`\readaline` `\readaline{<stream>}` reads what TeX considers to be one line from the given
 <stream> as `\input` text.

```

12174 \def\readaline#1{
12175   \instre@mandopen{#1}{%
12176     \ifeof\@nameuse{#1instre@m}
12177       \@memwarn{No more to read from stream #1}
12178     \else
12179       \read\@nameuse{#1instre@m} to\temptokstre@m
12180       \temptokstre@m
12181     \fi}}
12182

```

`\readverbatim` `\readverbatim{<stream>}` reads the contents of the given <stream> as verbatim
 text.

The read verbatim code is a slight variation on code from the `verbatim` package. Most of the setup is done by the macros `\stre@mverb@input{<setup>}{<stream>}` and `\verbatim@readstre@m{<stream>}`. Finally, `\verbatim@read@file` is a verbatim package macro.

```

12183 \def\readverbatim{\begingroup
12184   \ift@bs
12185   \def\verbatim@processline{\tabverbatim@processline}%

```



```

12186 \fi
12187 \@ifstar{\stre@mverb@input{\@maybeobeytabs}}%
12188      {\stre@mverb@input{\frenchspacing\@vobeyspaces\@maybeobeytabs}}
12189

```

`\stre@mverb@input` `\stre@mverb@input{⟨setup⟩}{⟨stream⟩}` is a stream version of `\@verbinstre@m` `\verbatim@input`. It defines `\@verbinstre@m` to be `⟨stream⟩`.

```

12190 \newcommand{\stre@mverb@input}[2]{%
12191   \IfStreamOpen{#2}%
12192     {\@verbatim #1\relax
12193      \def\@verbinstre@m{\@nameuse{#2instre@m}}
12194      \verb@readstre@m\endtrivlist\endgroup\@doendpe}%
12195     {\@memwarn{Stream #2\space is not open}\endgroup}}
12196

```

`\verb@readstre@m` `\verb@readstre@m` is the stream analogue to `\verbatim@readfile`.

```

12197 \newcommand{\verb@readstre@m}{%
12198   \verbatim@startline
12199   \expandafter\endlinechar\expandafter\m@ne
12200   \expandafter\verbatim@read@stre@m
12201   \expandafter\endlinechar\the\endlinechar\relax
12202   \verbatim@finish}
12203

```

`\verbatim@read@stre@m` `\verbatim@read@stre@m` is the analogue to `\verbatim@readfile`. It gets its input from the stream `\@verbinstre@m`, which is defined in the `\stre@mverb@input` macro.

```

12204 \newcommand{\verbatim@read@stre@m}{%
12205   \read\@verbinstre@m to\next
12206   \ifeof\@verbinstre@m
12207     \else
12208       \expandafter\verbatim@addtoline\expandafter{\next}%
12209       \verbatim@processline
12210       \verbatim@startline
12211       \expandafter\verbatim@read@stre@m
12212   \fi}
12213

```

`\readboxedverbatim` `\readboxedverbatim{⟨stream⟩}` is the equivalent of `\readverb@tim` except that it generates a boxed verbatim.

```

12214 \newcommand{\readboxedverbatim}{\begingroup
12215   \@ifstar{\stre@mbvin\relax}%
12216       {\stre@mbvin{\frenchspacing\@vobeyspaces}}}
12217

```

`\stre@mbvin` `\stre@mbvin{⟨setup⟩}{⟨stream⟩}` is the workhorse for `\readboxedverbatim`, and in its turn it uses `\verb@readstre@m`.

```

12218 \newcommand{\stre@mbvin}[2]{%
12219   \IfStreamOpen{#2}%

```

```

12220 {\setupboxverb@line
12221 \verbatimim #1\relax
12222 \def\@verbinstre@m{\@nameuse{#2instre@m}}%
12223 \setupbox@verb
12224 \verb@readstre@m\bvendrulehook\endtrivlist\endgroup\@doendpe}%
12225 {\@memwarn{Stream #2\space is not open}\endgroup}}
12226

```

31 Utilities

31.1 Extra ‘provide’ commands

`\provideenvironment` Works like `\providecommand` but for environments.

```

12227 \newcommand{\provideenvironment}{\@star@or@long\m@mprovenv}

```

`\m@mprovenv` This checks if the environment has been defined. If not it calls the kernel code for `\newenvironment` otherwise it calls code to discard arguments.

```

12228 \newcommand{\m@mprovenv}[1]{\@ifundefined{#1}%
12229 {\new@environment{#1}}% % create new environment
12230 {\@memwarn{Environment ‘#1’ already defined}%
12231 \m@gobbleoptsandtwo}}

```

`\m@gobbleoptsandtwo` `\m@gobbleoptsandtwo` gobbles the `[opt][opt]{begin}{end}` arguments to `\provideenvironment`. It is actually more general than that as it will gobble any sequence of any number of optional arguments followed by two required arguments. If there are no optional arguments it disposes of the two required ones, otherwise it calls a recursive procedure to discard any number of optionals, then two required ones.

`\m@gobbleoptandtwo` recursively discards optional arguments, then finally two required arguments.

```

12232 \newcommand{\m@gobbleoptsandtwo}{%
12233 \ifnextchar [{\m@gobbleoptandtwo}{\@gobbletwo}}
12234 \def\m@gobbleoptandtwo[#1]{%
12235 \ifnextchar [{\m@gobbleoptandtwo}{\@gobbletwo}}
12236

```

`\providecounter` Works like the other `\provide...` commands, but for counters. Code is based on `\newcounter` in `ltcounts.dtx`.

```

12237 \newcommand*{\providecounter}[1]{%
12238 \@ifundefined{c@#1}%
12239 {\newcounter{#1}}%
12240 {\@memwarn{Counter ‘#1’ already defined}%
12241 \@ifnextchar [{\m@gobbleopt}{}}
12242

```

`\m@gobbleopt` Gobble an optional argument.

```

12243 \def\m@gobbleopt[#1]{
12244

```

`\providelength` Works like the other `\provide...` commands, but for lengths. Code is based on `\provide@command` in `ltdefns.dtx`.

```

12245 \newcommand*{\providelength}[1]{%
12246   \begingroup
12247   \escapechar\m@ne\xdef\@gtempa{\string#1}%
12248   \endgroup
12249   \expandafter\@ifundefined\@gtempa
12250     {\newlength{#1}}%
12251     {\@memwarn{Length #1 already defined}}
12252
```

`\newloglike` For defining a new log-like math function. `\newloglike{\fun}{string}` or `\m@mnewlog` `\newloglike*{\fun}{string}`. The starred version is for functions with limits `\m@mnewlogs` (like an integral sign).

```

12253 \newcommand*{\newloglike}{\@ifstar{\m@mnewlogs}{\m@mnewlog}}
12254 \newcommand*{\m@mnewlogs}[2]{%
12255   \newcommand*{#1}{\mathop{\operator@font #2}}}
12256 \newcommand*{\m@mnewlog}[2]{%
12257   \newcommand*{#1}{\mathop{\operator@font #2}\nolimits}}
12258
```

`\provideloglike` The provide version of `\newloglike`.

```

\m@mprovlogs 12259 \newcommand*{\provideloglike}{\@ifstar{\m@mprovlogs}{\m@mprovlog}}
\m@mprovlog 12260 \newcommand*{\m@mprovlogs}[2]{%
12261   \providecommand*{#1}{\mathop{\operator@font #2}}}
12262 \newcommand*{\m@mprovlog}[2]{%
12263   \providecommand*{#1}{\mathop{\operator@font #2}\nolimits}}
12264
```

31.2 Changing counters

In LaTeX, a new counter called, say ‘ctr’, is created by the command `\newcounter{ctr}[within]`. If the optional within argument is given the counter ‘ctr’ is reset to zero each time the counter ‘within’ changes. The command `\thectr` typesets the value of the counter ctr. This is automatically defined by `\newcounter` and is initialised to typeset arabic numerals. Sometimes it may be desirable to change the counter definitions. The following code is based on the `chngcntr` package [Wil01e].

`\@removefromreset` This code uses David Carlisle’s `\@removefromreset` command as specified in the `remreset` package [Car98b]. It is provided here as a convenience to the user, and with David Carlisle’s permission.

START OF DAVID CARLISLE’S CODE

```

12265 \providecommand{\@removefromreset}[2]{%
12266   \expandafter\let\c@#1\endcsname\@removefromreset
12267   \def\@elt##1{%
12268     \expandafter\ifx\c@##1\endcsname\@removefromreset
12269     \else
```

```

12270      \noexpand\@elt{##1}%
12271      \fi}%
12272      \expandafter\xdef\csname cl@#2\endcsname{%
12273      \csname cl@#2\endcsname}}
12274

```

END OF DAVID CARLISLE'S CODE

`\ifbothcnters` This is called as
`\ifbothcnters{<ctr>}{<within>}{<code when both are counters>}`. This tests if
both `<ctr>` and `<within>` are counters.

```

12275 \newcommand{\ifbothcnters}[3]{%
12276   \ifundefined{c@#1}{% counter undefined
12277     \@memerror{#1 is not a counter}{\@eha}}%
12278   {% else counter is defined
12279     \ifundefined{c@#2}{% within undefined
12280       \@memerror{#2 is not a counter}{\@eha}}%
12281     {% else both counter and within are defined
12282       #3}}
12283

```

`\counterwithin` It is sometimes desirable to change a counter that has been defined by
`\counterwithin*` `\newcounter{ctr}` to act as though it had been defined as
`\@csinstar` `\newcounter{ctr}[within]`. The command `\counterwithin{ctr}{within}`
`\@csin` accomplishes this. By default, it also redefines the `\thectr` command so that it
typesets values in the style `\thewithin.\arabic{ctr}`. The starred version of
the command suppresses the redefinition of `\thectr`.

```

12284 \newcommand{\counterwithin}{\@ifstar{\@csinstar}{\@csin}}
12285 \newcommand{\@csinstar}[2]{%
12286   \ifbothcnters{#1}{#2}{\@addtoreset{#1}{#2}}}
12287 \newcommand{\@csin}[2]{%
12288   \ifbothcnters{#1}{#2}{\@addtoreset{#1}{#2}%
12289     \@namedef{the#1}{\@nameuse{the#2}.\arabic{#1}}}}
12290

```

`\counterwithout` Likewise, the command `\counterwithout{ctr}{within}` changes a counter that
`\counterwithout*` has been created by `\newcounter{ctr}[within]` to act as though it had been
`\@csoutstar` created by `\newcounter{ctr}`. By default it also redefines the `\thectr`
`\@csout` command so that it just typesets an arabic numeral. The starred version of the
command suppresses the redefinition of `\thectr`.

```

12291 \newcommand{\counterwithout}{\@ifstar{\@csoutstar}{\@csout}}
12292 \newcommand{\@csoutstar}[2]{%
12293   \ifbothcnters{#1}{#2}{\@removefromreset{#1}{#2}}}
12294 \newcommand{\@csout}[2]{%
12295   \ifbothcnters{#1}{#2}{\@removefromreset{#1}{#2}%
12296     \@namedef{the#1}{\arabic{#1}}}}
12297

```

Any number of `\counterwithin{ctr}{...}` and `\counterwithout{ctr}{...}`
commands can be issued for a given counter, `ctr`, if you wish to toggle between

the two styles. The current value of `ctr` is unaffected by `\counterwithin` and `\counterwithout`. If you want to change the value after one of these commands, use `\setcounter{ctr}{...}`, and to change the typesetting style use `\renewcommand{\thectr}{...}`.

`\letcountercounter` At times it is handy to ‘let’ one counter act as if it was a different counter. Say you have two constructions, each with their own counter A and B, now you want them to cooperate, counting in unison. This can be done using the `\letcountercounter`.

`\letcountercounter{<counterA>}{<counterB>} \lets` (make the same `<counterA>` to `<counterB>`). The original of `<counterA>` is kept in `\c@m@morig@ctr<counterA>`. `\unletcounter{<counterA>}` restores `<counterA>` to its un`\let` condition. The code is based on Donald Arseneau’s ‘Re: `\let`, `\csname` and `\endcsname`’, CTT, 2004/12/08. The long version, where `\xp` is actually `\expandafter` is along the lines of:

```
\xp\xp\xp\let\xp\csname\xp c@#1\xp\endcsname\csname c@#2\endcsname
```

which employs 6 `\expandafters`. DA’s version uses only 2. The technique can be applied to other kinds of `\lets`.

```
12298 \newcommand*\letcountercounter}[2]{%
12299   \expandafter\let\csname c@m@morig@ctr#1\expandafter\endcsname%
12300   \csname c@#1\endcsname
12301   \expandafter\let\csname c@#1\expandafter\endcsname%
12302   \csname c@#2\endcsname}
12303 \newcommand*\unletcounter}[1]{%
12304   \expandafter\let\csname c@#1\expandafter\endcsname%
12305   \csname c@m@morig@ctr#1\endcsname}
12306
```

31.3 Odd/even page checking

It is difficult to check robustly if the current page is odd or even. This code, which is based on the `chnpage` package [Wil01b], provides a method that has proved (so far) to be robust. It works by writing out a label and on the next LaTeX run checks the page reference for the label.

`\ifoddpag` The boolean `\ifoddpag` is TRUE if the checked page is odd. The boolean `\oddpagetrue` `\ifstrictpagecheck` is for turning on (TRUE) and off (FALSE) the strictest method of page checking.

```
\ifstrictpagecheck 12307 \newif\ifoddpag
\strictpagechecktrue 12308 \newif\ifstrictpagecheck
\strictpagecheckfalse 12309 \strictpagecheckfalse
\strictpagecheck 12310 \newcommand*\strictpagecheck{\strictpagechecktrue}
\easypagecheck 12311 \newcommand*\easypagecheck{\strictpagecheckfalse}
12312
```

`\c@cp@cntr` The counter `cp@cntr` is used to make unique labels, which start with `\cplabel`.

```

\cplabel12313 \newcounter{cp@cntr}
12314 \newcommand{\cplabel}{^_}

```

`\checkoddpag` This is the user level command to check for odd/even page. It does a robust check for `\strictpagecheck` otherwise the simple minded check. It sets `\ifoddpag` to TRUE if the page is odd, otherwise it sets it to FALSE. This is now fixed so that it should work for non-arabic page numbering. It uses a new label/pageref variant based on the page counter value, not its printed representation. This also gets rid of worrying about hyperref!! The problem was discovered by Bastiaan Niels Veelo

```

12315 \DeclareRobustCommand{\checkoddpag}{%
12316   \oddpagfalse%
12317   \ifstrictpagecheck%
12318     \stepcounter{cp@cntr}\pmemlabel{\cplabel\thecp@cntr}%
12319     \@memcnta=\pmemlabelref{\cplabel\thecp@cntr}\relax
12320     \ifodd\@memcnta\oddpagtrue\fi
12321   \else
12322     \ifodd\c@page\oddpagtrue\fi
12323   \fi}
12324

```

`\thepmemc@page` The value of the page counter.

```

12325 \gdef\thepmemc@page{\the\c@page}
12326

```

`\pmemprotected@write` `\pmemprotected@write` is a modified version of the kernel's `\protected@write`. This version of the macro was provided by Romano Giannetti (romano@dea.ica.i.upco.es) on 15 July 2003.

```

12327 \long\def\pmemprotected@write#1#2#3{%
12328   \begingroup
12329   \let\thepmemc@page\relax
12330   #2%
12331   \let\protect\@unexpandable@protect
12332   \edef\reserved@a{\write#1{#3}}%
12333   \reserved@a
12334   \endgroup
12335   \if@nobeak\ifvmode\nobeak\fi\fi}
12336

```

`\pmemlabel` A version of `\label` that uses `\pmemprotected@write`.

```

12337 \newcommand{\pmemlabel}[1]{\@bsphack
12338   \pmemprotected@write\@auxout{%
12339     {\string\newpmemlabel{#1}{\thepmemc@page}}%
12340   \@esphack}

```

`\newpmemlabel` We have to be able to cope with a particular label not (yet) being in the the

`\pmemlabelref` .aux file when we come to check the page number.

```

12341 \newcommand{\newpmemlabel}[2]{\global\@namedef{m@#1}{#2}}
12342 \newcommand{\pmemlabelref}[1]{%
12343   \expandafter\ifx\csname m@#1\endcsname\relax
12344     0%
12345   \else
12346     \csname m@#1\endcsname
12347   \fi}
12348

```

31.4 Checking for empty arguments

Like page checking, testing for an empty macro argument is more difficult than it might appear at first sight.

The following code is from the ifmtarg package [ArWi00].

```

\@ifmtarg \@ifmtarg{<arg>}{<code when empty>}{<code when arg not empty>}
\@ifnotmtarg \@ifnotmtarg{<arg>}{<code when arg not empty>}

12349 \begin{group}
12350 \catcode'\Q=3
12351 \long\gdef\@ifmtarg#1{\@xifmtarg#1QQ\@secondoftwo\@firstoftwo\@nil}
12352 \long\gdef\@xifmtarg#1#2Q#3#4#5\@nil{#4}
12353 \long\gdef\@ifnotmtarg#1{\@xifmtarg#1QQ\@firstofone\@gobble\@nil}
12354 \end{group}
12355

```

Some example uses:

```

\newcommand{\isempty}[1]{%
  \@ifmtarg{#1}{\typeout{YES}}{\typeout{NO}}}
\newcommand{\isnotempty}[1]{%
  \@ifnotmtarg{#1}{\typeout{YES}}}

\isempty{}      -> YES  \isnotempty{}      ->
\isempty{ }    -> YES  \isnotempty{ }    ->
\isempty{A}    -> NO   \isnotempty{A}    -> YES
\isempty{ A } -> NO   \isnotempty{ A } -> YES

```

31.5 Changing the page layout in the document

You should not do this, but...

The following code is essentially from the chngpage package [Wil01b].

```

\ch@ngetext This macro sets the page output parameters.

12356 \DeclareRobustCommand{\ch@ngetext}{%
12357   \setlength{\@colht}{\textheight}\setlength{\@colroom}{\textheight}%
12358   \setlength{\vsize}{\textheight}\setlength{\columnwidth}{\textwidth}%
12359   \if@twocolumn%
12360     \advance\columnwidth-\columnsep \divide\columnwidth\tw@%

```

```

12361 \firstcolumntrue%
12362 \fi%
12363 \setlength{\hspace}{\columnwidth}%
12364 \setlength{\linewidth}{\hspace}}
12365

```

`\changetext` `\changetext{<H>}{<W>}{<E>}{<O>}{<G>}` adds the given lengths to 5 main bits of the page layout. An empty argument means ‘no change’.

```

12366 \DeclareRobustCommand{\changetext}[5]{%
12367 \ifmtarg{#1}{\addtolength{\textheight}{#1}}%
12368 \ifmtarg{#2}{\addtolength{\textwidth}{#2}}%
12369 \ifmtarg{#3}{\addtolength{\evensidemargin}{#3}}%
12370 \ifmtarg{#4}{\addtolength{\oddsidemargin}{#4}}%
12371 \ifmtarg{#5}{\addtolength{\columnsep}{#5}}%
12372 \ch@ngetext}
12373

```

Any given change lasts until another `\changetext` command is given. A `\changetext` command should only be issued between two paragraphs. `\changetext` should not be used by anyone unless they really know what they are doing. If you really know what you are doing then you should know enough not to use it.

`\changeage` Change the page layout, but DON’T.

```

12374 \DeclareRobustCommand{\changeage}[9]{%
12375 \ifmtarg{#1}{\addtolength{\textheight}{#1}}%
12376 \ifmtarg{#2}{\addtolength{\textwidth}{#2}}%
12377 \ifmtarg{#3}{\addtolength{\evensidemargin}{#3}}%
12378 \ifmtarg{#4}{\addtolength{\oddsidemargin}{#4}}%
12379 \ifmtarg{#5}{\addtolength{\columnsep}{#5}}%
12380 \ch@ngetext
12381 \ifmtarg{#6}{\addtolength{\topmargin}{#6}}%
12382 \ifmtarg{#7}{\addtolength{\headheight}{#7}}%
12383 \ifmtarg{#8}{\addtolength{\headsep}{#8}}%
12384 \ifmtarg{#9}{\addtolength{\footskip}{#9}}%
12385

```

31.6 Temporarily changing the text width

`adjustwidth` `\begin{adjustwidth}{<left>}{<right>}` adds the given lengths to the left and right hand margins. A positive value will shorten the text and a negative value will widen it. The starred version of the environment will cause the margin changes to switch between odd and even pages.

This code is based on the `chnpage` package.

```

12386 \newenvironment{adjustwidth}[2]{%
12387 \begin{list}{}{}%
12388 \topsep\z@%
12389 \listparindent\parindent%

```



```

12390 \parsep\parskip%
12391 \@ifmtarg{#1}{\setlength{\leftmargin}{\z@}}%
12392           {\setlength{\leftmargin}{#1}}%
12393 \@ifmtarg{#2}{\setlength{\rightmargin}{\z@}}%
12394           {\setlength{\rightmargin}{#2}}%
12395 }
12396 \item[]{\end{list}}
12397
12398 \newenvironment{adjustwidth*}[2]{%
12399   \begin{list}{}{%
12400     \topsep\z@%
12401     \listparindent\parindent%
12402     \parsep\parskip%
12403     \checkoddpage
12404     \ifoddpage % odd numbered page
12405       \@ifmtarg{#1}{\setlength{\leftmargin}{\z@}}%
12406               {\setlength{\leftmargin}{#1}}%
12407       \@ifmtarg{#2}{\setlength{\rightmargin}{\z@}}%
12408               {\setlength{\rightmargin}{#2}}%
12409     \else % even numbered page
12410       \@ifmtarg{#2}{\setlength{\leftmargin}{\z@}}%
12411               {\setlength{\leftmargin}{#2}}%
12412       \@ifmtarg{#1}{\setlength{\rightmargin}{\z@}}%
12413               {\setlength{\rightmargin}{#1}}%
12414     \fi
12415   }
12416   \item[]{\end{list}}
12417

```

The environments only work for complete paragraphs.

31.7 Centering text

\calccentering This macro calculates the amount to be added to the spine margin (and subtracted from the foreedge margin) in order to center the textblock. Call as `\calccentering{length}` and it sets `length` to the required value. Then use

as

```
\begin{adjustwidth*}{length}{-length}...
```

```

12418 \newcommand{\calccentering}[1]{
12419   #1 = \paperwidth
12420   \advance #1 by -\textwidth
12421   \divide #1 by \tw@
12422   \advance #1 by -\spinemargin}
12423

```

vplace `\begin{vplace}[decimal number]` centers its body vertically. The optional argument *decimal number* increases (> 1.0) or decreases (< 1.0) the space above the body with respect to the space below. This is more robust than the method proposed in the manual.

```

12424 \newenvironment{vplace}[1][1]{%
12425   \par\vspace{\stretch{#1}}%
12426 }{%
12427   \vspace*{\stretch{1}}%
12428   \par}
12429

```

31.8 Moving from the current page

Much of this code is taken from the `nextpage` package [Wil00c].

Clear to next page is `\clearpage`.

`\cleartoevenpage` `\cleartoevenpage[text]` clears to the next even numbered page, putting the optional *text* on the skipped page, if any.

```

12430 \newcommand{\cleartoevenpage}[1][\@empty]{%
12431   \clearpage%
12432   \ifodd\c@page\hbox{#1\clearpage\fi}
12433

```

Move to the next page without flushing floats is `\newpage`.

`\movetoevenpage` `\movetoevenpage[text]` moves to the next even numbered page without flushing floats, putting the optional *text* on the skipped page, if any.

```

12434 \newcommand{\movetoevenpage}[1][\@empty]{%
12435   \newpage%
12436   \ifodd\c@page\hbox{#1\newpage\fi}
12437

```

Clear to next odd numbered page is `\cleardoublepage`.

`\cleartooddpage` `\cleartooddpage[text]` clears to the next odd numbered page, putting the optional *text* on the skipped page, if any.

```

12438 \newcommand{\cleartooddpage}[1][\@empty]{%
12439   \clearpage%
12440   \ifodd\c@page\else\hbox{#1\clearpage\fi}
12441

```

`\movetooddpage` `\movetooddpage[text]` moves to the next odd numbered page without flushing floats, putting the optional *text* on the skipped page, if any.

```

12442 \newcommand{\movetooddpage}[1][\@empty]{%
12443   \newpage%
12444   \ifodd\c@page\else\hbox{#1\newpage\fi}
12445

```

Example uses:

```

\cleartooddpage           % same as \cleardouble page
\cleartooddpage[\thispagestyle{empty}] % No headings on the skipped page
\cleartoevenpage          % go to next even numbered page

```

Next example puts text on a skipped page
`\cleartoevenpage[\vspace*{\hfill}THIS PAGE LEFT BLANK\vspace*{\hfill}]`

31.9 Needing space at the bottom of a page

`\needspace` `\needspace{<length>}` checks if there is `<length>` amount of vertical space left on the page. If there is not it will start a new page. The code is a variant of code thought of for the `needspace` package [Wil00d].

```
12446 \newcommand{\needspace}[1]{\begingroup\setlength{\dimen@}{#1}%
12447 \vskip\z@\@plus\dimen@\penalty -100\vskip\z@\@plus-\dimen@
12448 \vskip\dimen@\penalty 9999\vskip-\dimen@\endgroup}
12449
```

`\Needspace` `\Needspace` and `\Needspace*` are more robust versions of `\needspace`, which depends on penalties. If either `\needspace` or `\Needspace` produce a short page it will be ragged bottom, even if `\flushbottom` is in effect. The `\Needspace*` version honours the `\dotsbottom` declaration.

```
12450 \newcommand{\Needspace}{\@ifstar{\M@needspace}{\M@needspace}}
```

`\M@needspace` These implement `\Needspace*` and `\needspace` respectively.

```
\M@needspace 12451 \newcommand{\M@needspace}[1]{\par \penalty-100\begingroup
12452 \setlength{\dimen@}{#1}%
12453 \dimen@ii\pagegoal \advance\dimen@ii-\pagetotal
12454 \ifdim \dimen@>\dimen@ii
12455 \break
12456 \fi\endgroup}
12457 \newcommand{\M@needspace}[1]{\par \penalty-100\begingroup
12458 \setlength{\dimen@}{#1}%
12459 \dimen@ii\pagegoal \advance\dimen@ii-\pagetotal
12460 \ifdim \dimen@>\dimen@ii
12461 \ifdim \dimen@ii>\z@
12462 \vfil
12463 \fi
12464 \break
12465 \fi\endgroup}
12466
```

31.10 Overlong lines

LaTeX provides `\fussy` and `\sloppy` to control the amount of slack in a line while trying to make justified lines. Their definitions are:

```
\def\fussy{%
  \emergencystretch\z@
  \tolerance 200%
  \hfuzz .1\p@
  \vfuzz\hfuzz}
\def\sloppy{%
```

```

\tolerance 9999%
\emergencystretch 3em%
\hfuzz .5\p@
\vfuzz\hfuzz}

```

`\midsloppy` Somewhere between `\fussy` and `\sloppy`.

```

12467 \newcommand*\midsloppy{%
12468   \tolerance 5000%
12469   \hbadness 4000%
12470   \emergencystretch 1.5em%
12471   \hfuzz .1\p@
12472   \vfuzz\hfuzz}

```

`midsloppypar` A paragraph form of `\midsloppy`; equivalent to `\par \midsloppy ... \par`

```

12473 \newenvironment{midsloppypar}{\par\midsloppy}{\par}
12474

```

31.11 Text spacing commands

While we're at it, new text spacing commands to supplement the kernel's `\`, (`\thinspace`) = 3/18 em

`\medspace` Medium space `\:` (`\medspace`) = 4/18 em

```

\!12475 \newcommand{\medspace}{\kern .22222em }
12476 \DeclareRobustCommand{\:}{%
12477   \relax\ifmmode\mskip\medmuskip\else\medspace\fi}

```

`\!` Negative thin space (- 3/18 em). Patrik Nyman (private email 2003/10/06) pointed out I had missed the final `\fi` !!!!.

```

12478 \DeclareRobustCommand{\!}{%
12479   \relax\ifmmode\mskip-\thinmuskip\else\negthinspace\fi}
12480

```

31.12 Fractions and subscripts

Styles for fractions like 3/4.

`\slashfracstyle` `\slashfracstyle` based on the kernel `\textsuperscript` macro

```

12481 \DeclareRobustCommand*\slashfracstyle[1]{%
12482   {\m@th\ensuremath{\mbox{\fontsize\sf@size\z@\selectfont #1}}}}

```

`\slashfrac` `\slashfrac` based on TeXbook exercise 11.6

```

12483 \DeclareRobustCommand*\slashfrac[2]{\leavevmode
12484   \raise.5ex\hbox{\slashfracstyle{#1}}\kern-.13em/%
12485   \kern-.15em\lower.25ex\hbox{\slashfracstyle{#2}}}
12486

```

`\textsubscript` `\textsubscript`, for text subscripts, based on the kernel `\textsuperscript`
`\@textsubscript` macro.

```
12487 \DeclareRobustCommand*\textsubscript[1]{%
12488   \@textsubscript{\selectfont#1}}
12489 \newcommand*\@textsubscript[1]{%
12490   {\m@th\ensuremath{_{\mbox{\fontsize\sf@size\z@#1}}}}}
12491
```

31.13 Numbers to names

```
12492 %%%%%%%%%%%%%%%%%%%%%%%%% number formatting
12493
```

The macros here convert an integer number in the range 0–2147483647 (TeX’s maximum) into the natural language name of the number.

`\iflowernumtoname` A flag for lowercasing the initial letters.

```
12494 \newif\iflowernumtoname
12495 \lowernumtonamefalse
```

`\ifpriornum` Some new booleans

```
\ifminusnumber 12496 \newif\ifpriornum
\ifnotnumtonameallcaps 12497 \newif\ifminusnumber
\ifmakeordinal 12498 \newif\ifnotnumtonameallcaps
12499 \newif\ifmakeordinal
12500
```

`\namenumberand` Some macros representing characters/words for number formatting.

```
\namenumbercomma 12501 \newcommand*\namenumberand{{ and }
\lcmminusname 12502 \newcommand*\namenumbercomma}{, }
\ucminusname 12503 \newcommand*\lcmminusname}{minus }
\minusname 12504 \newcommand*\ucminusname}{Minus }
12505 \let\minusname\lcmminusname
```

`\fnumbersep` More characters/words for formatting

```
\tensunitsep 12506 \newcommand*\fnumbersep{,}
\nthstring 12507 \newcommand*\tensunitsep{-}
\iststring 12508 \newcommand*\nthstring{th} % nth
\iindstring 12509 \newcommand*\iststring{st} % 1st
\iiirdstring 12510 \newcommand*\iindstring{nd} % 2nd
\tiethstring 12511 \newcommand*\iiirdstring{rd} % 3rd
\teenstring 12512 \newcommand*\tiethstring{tieth} % tieth
12513 \newcommand*\teenstring{teen} % teen
```

`\ordscript` How to format the ordinal string

```
12514 \newcommand*\ordscript[1]{#1}
12515
```

`\m@mten` An internal shorthand for 10

```
12516 \chardef\m@mten=10 % shorthand for 10
12517
```

```

\c@ism@mctr Counters for digits (units to ten thousands) in a number
\c@xsm@mctr\2518 \newcounter{ism@mctr} % units
\c@csm@mctr\2519 \newcounter{xsm@mctr} % tens
\c@ksm@mctr\2520 \newcounter{csm@mctr} % hundreds
\c@xksm@mctr\2521 \newcounter{ksm@mctr} % thousands
12522 \newcounter{xksm@mctr} % ten thousands

\c@cksm@mctr Counters for digits (hundred thousands to billions) in a number
\c@msm@mctr\2523 \newcounter{cksm@mctr} % hundred thousands
\c@xmsm@mctr\2524 \newcounter{msm@mctr} % millions
\c@cmsm@mctr\2525 \newcounter{xmsm@mctr} % ten millions
\c@bsm@mctr\2526 \newcounter{cmsm@mctr} % hundred millions
12527 \newcounter{bsm@mctr} % billions

\c@workm@mctr A 'work' counter.
12528 \newcounter{workm@mctr}
12529

\numdigits \numdigits{<number>} splits the <number> into individual digits. It sets
\minusnumbertrue if the number is negative, otherwise it is false.
12530 \newcommand*{\numdigits}[1]{%
12531 \setcounter{ism@mctr}{0}%
12532 \setcounter{xsm@mctr}{0}%
12533 \setcounter{csm@mctr}{0}%
12534 \setcounter{ksm@mctr}{0}%
12535 \setcounter{xksm@mctr}{0}%
12536 \setcounter{cksm@mctr}{0}%
12537 \setcounter{msm@mctr}{0}%
12538 \setcounter{xmsm@mctr}{0}%
12539 \setcounter{cmsm@mctr}{0}%
12540 \setcounter{bsm@mctr}{0}%
12541 \setcounter{workm@mctr}{#1}%
12542 \minusnumberfalse
12543 \ifnum \c@workm@mctr < \z@ % negative
12544 \minusnumbertrue
12545 \c@workm@mctr = -\c@workm@mctr
12546 \fi
12547 \ifnum \c@workm@mctr > \m@ne % units
12548 \c@ism@mctr = \c@workm@mctr
12549 \divide \c@workm@mctr by \m@ten
12550 \multiply \c@workm@mctr by \m@ten
12551 \advance \c@ism@mctr by -\c@workm@mctr
12552 \divide \c@workm@mctr by \m@ten
12553 \fi
12554 \ifnum \c@workm@mctr > \z@ % tens
12555 \c@xsm@mctr = \c@workm@mctr
12556 \divide \c@workm@mctr by \m@ten
12557 \multiply \c@workm@mctr by \m@ten
12558 \advance \c@xsm@mctr by -\c@workm@mctr

```

```

12559     \divide \c@workm@mctr by \m@ten
12560 \fi
12561 \ifnum \c@workm@mctr > \z@    % hundreds
12562     \c@csm@mctr = \c@workm@mctr
12563     \divide \c@workm@mctr by \m@ten
12564     \multiply \c@workm@mctr by \m@ten
12565     \advance \c@csm@mctr by -\c@workm@mctr
12566     \divide \c@workm@mctr by \m@ten
12567 \fi
12568 \ifnum \c@workm@mctr > \z@    % thousands
12569     \c@ksm@mctr = \c@workm@mctr
12570     \divide \c@workm@mctr by \m@ten
12571     \multiply \c@workm@mctr by \m@ten
12572     \advance \c@ksm@mctr by -\c@workm@mctr
12573     \divide \c@workm@mctr by \m@ten
12574 \fi
12575 \ifnum \c@workm@mctr > \z@    % ten thousands
12576     \c@xksm@mctr = \c@workm@mctr
12577     \divide \c@workm@mctr by \m@ten
12578     \multiply \c@workm@mctr by \m@ten
12579     \advance \c@xksm@mctr by -\c@workm@mctr
12580     \divide \c@workm@mctr by \m@ten
12581 \fi
12582 \ifnum \c@workm@mctr > \z@    % hundred thousands
12583     \c@cksm@mctr = \c@workm@mctr
12584     \divide \c@workm@mctr by \m@ten
12585     \multiply \c@workm@mctr by \m@ten
12586     \advance \c@cksm@mctr by -\c@workm@mctr
12587     \divide \c@workm@mctr by \m@ten
12588 \fi
12589 \ifnum \c@workm@mctr > \z@    % millions
12590     \c@msm@mctr = \c@workm@mctr
12591     \divide \c@workm@mctr by \m@ten
12592     \multiply \c@workm@mctr by \m@ten
12593     \advance \c@msm@mctr by -\c@workm@mctr
12594     \divide \c@workm@mctr by \m@ten
12595 \fi
12596 \ifnum \c@workm@mctr > \z@    % ten millions
12597     \c@xmsm@mctr = \c@workm@mctr
12598     \divide \c@workm@mctr by \m@ten
12599     \multiply \c@workm@mctr by \m@ten
12600     \advance \c@xmsm@mctr by -\c@workm@mctr
12601     \divide \c@workm@mctr by \m@ten
12602 \fi
12603 \ifnum \c@workm@mctr > \z@    % hundred millions
12604     \c@cmsm@mctr = \c@workm@mctr
12605     \divide \c@workm@mctr by \m@ten
12606     \multiply \c@workm@mctr by \m@ten
12607     \advance \c@cmsm@mctr by -\c@workm@mctr
12608     \divide \c@workm@mctr by \m@ten

```

```

12609 \fi
12610 \ifnum \c@workm@mctr > \z@% billions
12611   \c@bsm@mctr = \c@workm@mctr
12612   \divide \c@workm@mctr by \m@ten
12613   \multiply \c@workm@mctr by \m@ten
12614   \advance \c@bsm@mctr by -\c@workm@mctr
12615 \fi}
12616

```

`\form@tnumber` `\form@tnumber{<number>}` formats <number> as digits.

```

12617 \newcommand*{\form@tnumber}[1]{%
12618   \numdigits{#1}%
12619   \ifminusnumber-\fi
12620   \priornumfalse
12621   \ifnum \c@bsm@mctr > \z@ % billions
12622     \priornumtrue
12623     \thebsm@mctr\fnumbersep
12624   \fi
12625   \ifpriornum           % hundred millions
12626     \thecsm@mctr
12627   \else
12628     \ifnum \c@csm@mctr > \z@
12629       \priornumtrue
12630       \thecsm@mctr
12631     \fi
12632   \fi
12633   \ifpriornum           % ten millions
12634     \thexsm@mctr
12635   \else
12636     \ifnum \c@xsm@mctr > \z@
12637       \priornumtrue
12638       \thexsm@mctr
12639     \fi
12640   \fi
12641   \ifpriornum           % millions
12642     \themsm@mctr\fnumbersep
12643   \else
12644     \ifnum \c@msm@mctr > \z@
12645       \priornumtrue
12646       \themsm@mctr\fnumbersep
12647     \fi
12648   \fi
12649   \ifpriornum           % hundred thousands
12650     \thecksm@mctr
12651   \else
12652     \ifnum \c@cksm@mctr > \z@
12653       \priornumtrue
12654       \thecksm@mctr
12655     \fi
12656   \fi

```



```

12657 \ifpriornum           % ten thousands
12658   \the xsm@mctr
12659 \else
12660   \ifnum \c@xsm@mctr > \z@
12661     \priornumtrue
12662     \the xsm@mctr
12663   \fi
12664 \fi
12665 \ifpriornum           % thousands
12666   \the xsm@mctr \fnumbersep
12667 \else
12668   \ifnum \c@ksm@mctr > \z@
12669     \priornumtrue
12670     \the xsm@mctr \fnumbersep
12671   \fi
12672 \fi
12673 \ifpriornum           % hundreds
12674   \the csm@mctr
12675 \else
12676   \ifnum \c@csm@mctr > \z@
12677     \priornumtrue
12678     \the csm@mctr
12679   \fi
12680 \fi
12681 \ifpriornum           % tens
12682   \the xsm@mctr
12683 \else
12684   \ifnum \c@xsm@mctr > \z@
12685     \priornumtrue
12686     \the xsm@mctr
12687   \fi
12688 \fi
12689 \the ism@mctr}          % units
12690

\cardinal \cardinal{<number>} prints <number> unformatted.
12691 \newcommand*{\cardinal}[1]{%
12692   \begingroup
12693   \let\fnumbersep\relax
12694   \form@tnumber{#1}%
12695   \endgroup}

\fcardinal \fcardinal{<number>} prints <number> formatted.
12696 \newcommand*{\fcardinal}[1]{%
12697   \begingroup
12698   \form@tnumber{#1}%
12699   \endgroup}
12700

\ordinal \ordinal{<number>} prints <number> as an unformatted ordinal.

```

```

12701 \newcommand*{\ordinal}[1]{%
12702   \begingroup
12703   \let\fnnumbersep\relax
12704   \form@tnumber{#1}%
12705   \let\ordstring\nthstring
12706   \ifnum \c@xsm@mctr=\@ne\else
12707     \ifcase \c@ism@mctr
12708       \or \let\ordstring\iststring%    1st
12709       \or \let\ordstring\iindstring%  2nd
12710       \or \let\ordstring\iiirdstring% 3rd
12711     \fi
12712   \fi
12713   \ordscript{\ordstring}%
12714   \endgroup
12715 }

```

`\fordinal` `\fordinal{<number>}` prints `<number>` as a formatted ordinal.

```

12716 \newcommand*{\fordinal}[1]{%
12717   \begingroup
12718   \form@tnumber{#1}%
12719   \let\ordstring\nthstring
12720   \ifnum \c@xsm@mctr=\@ne\else
12721     \ifcase \c@ism@mctr
12722       \or \let\ordstring\iststring%    1st
12723       \or \let\ordstring\iindstring%  2nd
12724       \or \let\ordstring\iiirdstring% 3rd
12725     \fi
12726   \fi
12727   \ordscript{\ordstring}%
12728   \endgroup
12729 }
12730

```

The next, tedious, code is for translating numbers into names.

`\nNameo` Names of major numbers: 0, 10^2 , 10^3 , 10^6 , and 10^9 .

```

\nNameo 2731 \newcommand*\nNameo{\iflowernumtoname z\else Z\fi ero}
\nNameh 2732 \newcommand*\nNameh{\iflowernumtoname h\else H\fi undred}
\nNamem 2733 \newcommand*\nNamem{\iflowernumtoname t\else T\fi housand}
\nNamev 2734 \newcommand*\nNamev{\iflowernumtoname m\else M\fi illion}
12735 \newcommand*\nNamev{\iflowernumtoname b\else B\fi illion}
12736

```

`\nNamei` These are the names for numbers 1 to 5.

```

\nNamei 2737 \newcommand*\nNamei{\iflowernumtoname o\else O\fi ne}
\nNameii 2738 \newcommand*\nNameii{\iflowernumtoname t\else T\fi wo}
\nNameiii 2739 \newcommand*\nNameiii{\iflowernumtoname t\else T\fi hree}
\nNameiv 2740 \newcommand*\nNameiv{\iflowernumtoname f\else F\fi our}
12741 \newcommand*\nNamev{\iflowernumtoname f\else F\fi ive}

```

`\Namevi` These are the names for numbers 6 to 10.

```

\Namevii2742 \newcommand*\Namevi{\iflowernumtoname s\else S\fi ix}
\Nameviii2743 \newcommand*\Namevii{\iflowernumtoname s\else S\fi even}
\Nameix2744 \newcommand*\Nameviii{\iflowernumtoname e\else E\fi ight}
\Namex2745 \newcommand*\Nameix{\iflowernumtoname n\else N\fi ine}
12746 \newcommand*\Nameix{\iflowernumtoname t\else T\fi en}

```

`\Namexi` These are the names for numbers 11 to 15.

```

\Namexii2747 \newcommand*\Namexi{\iflowernumtoname e\else E\fi leven}
\Namexiii2748 \newcommand*\Namexii{\iflowernumtoname t\else T\fi welve}
\Namexiv2749 \newcommand*\Namexiii{\iflowernumtoname t\else T\fi hir\teenstring}
\Namexv2750 \newcommand*\Namexiv{\iflowernumtoname f\else F\fi our\teenstring}
12751 \newcommand*\Namexv{\iflowernumtoname f\else F\fi if\teenstring}

```

`\Namexvi` These are the names for numbers 16 to 20.

```

\Namexvii2752 \newcommand*\Namexvi{\iflowernumtoname s\else S\fi ix\teenstring}
\Namexviii2753 \newcommand*\Namexvii{\iflowernumtoname s\else S\fi even\teenstring}
\Namexix2754 \newcommand*\Namexviii{\iflowernumtoname e\else E\fi igh\teenstring}
\Namexx2755 \newcommand*\Namexix{\iflowernumtoname n\else N\fi ine\teenstring}
12756 \newcommand*\Namexx{\iflowernumtoname t\else T\fi wenty}

```

`\Namexxx` These are the names for numbers 30 to 70.

```

\Namexl2757 \newcommand*\Namexxx{\iflowernumtoname t\else T\fi hirty}
\Namel2758 \newcommand*\Namexl{\iflowernumtoname f\else F\fi orty}
\Namelx2759 \newcommand*\Namel{\iflowernumtoname f\else F\fi ifty}
\Namelxx2760 \newcommand*\Namelx{\iflowernumtoname s\else S\fi ixty}
12761 \newcommand*\Namelxx{\iflowernumtoname s\else S\fi eventy}

```

`\Namelxxx` These are the names for numbers 80 to 90.

```

\Nameqx2762 \newcommand*\Namelxxx{\iflowernumtoname e\else E\fi ighty}
12763 \newcommand*\Namexc{\iflowernumtoname n\else N\fi inety}
12764

```

`\unitnumbername` Get the name of a unit (0 – 9).

```

12765 \newcommand*\unitnumbername[1]{%
12766 \ifcase #1 \Nameo\or
12767 \Namei\or
12768 \Nameii\or
12769 \Nameiii\or
12770 \Nameiv\or
12771 \Namev\or
12772 \Namevi\or
12773 \Namevii\or
12774 \Nameviii\or
12775 \Nameix\fi}
12776

```

`\teennumbername` Get the name of a 'teen number (10 – 19)

```

12777 \newcommand*\teennumbername[1]{%

```

```

12778 \ifcase #1 \nName\or
12779 \nNamei\or
12780 \nNameii\or
12781 \nNameiii\or
12782 \nNameiv\or
12783 \nNamev\or
12784 \nNamevi\or
12785 \nNamevii\or
12786 \nNameviii\or
12787 \nNameix\fi}
12788

```

`\tensnumbername` Get the name of a tens number (20 – 90)

```

12789 \newcommand*\tensnumbername[2]{%
12790 \ifnum #1=\@ne
12791 \teennumbername{#2}\ifnotnumtonameallcaps\lowernumtonametrue\fi
12792 \else
12793 \ifcase #1
12794 \or
12795 \or \nNamexx
12796 \or \nNamexxx
12797 \or \nNamexl
12798 \or \nName1
12799 \or \nName1x
12800 \or \nName1xx
12801 \or \nName1xxx
12802 \or \nNamexc
12803 \fi
12804 \ifnotnumtonameallcaps\lowernumtonametrue\fi
12805 \ifnum #2 > \z@ \tensunitsep\unitnumbername{#2}\fi
12806 \fi}
12807

```

Names of small ordinals. The use of `\nthstring` instead of ‘th’ saves some tokens.

```

12808 \newcommand*\nthNameo{\nNameo\nthstring}
12809 \newcommand*\nthNamei{\iflowernumtoname f\else F\fi irst}
12810 \newcommand*\nthNameii{\iflowernumtoname s\else S\fi econd}
12811 \newcommand*\nthNameiii{\iflowernumtoname t\else T\fi hird}
12812 \newcommand*\nthNameiv{\nNameiv\nthstring}
12813 \newcommand*\nthNamev{\iflowernumtoname f\else F\fi if\nthstring}
12814 \newcommand*\nthNamevi{\nNamevi\nthstring}
12815 \newcommand*\nthNamevii{\nNamevii\nthstring}
12816 \newcommand*\nthNameviii{\iflowernumtoname e\else E\fi igh\nthstring}
12817 \newcommand*\nthNameix{\iflowernumtoname n\else N\fi in\nthstring}
12818 \newcommand*\nthNameixii{\iflowernumtoname t\else T\fi welf\nthstring}
12819

```

`\unitordinalname` Get the ordinal name of a unit (0 – 9)

```

12820 \newcommand*{\unitordinalname}[1]{%
12821   \ifcase #1 \nthNameo\or
12822   \nthNamei\or
12823   \nthNameii\or
12824   \nthNameiii\or
12825   \nthNameiv\or
12826   \nthNamev\or
12827   \nthNamevi\or
12828   \nthNamevii\or
12829   \nthNameviii\or
12830   \nthNameix\fi}
12831

```

`\teenordinalname` Get the ordinal name of a 'teen number (10 – 19). Using `\nthstring` instead of 'th' to save some tokens.

```

12832 \newcommand*{\teenordinalname}[1]{%
12833   \ifcase #1 \nName\ nthstring\or
12834   \nNamexi\ nthstring\or
12835   \nthNamexii\or
12836   \nNamexiii\ nthstring\or
12837   \nNamexiv\ nthstring\or
12838   \nNamexv\ nthstring\or
12839   \nNamexvi\ nthstring\or
12840   \nNamexvii\ nthstring\or
12841   \nNamexviii\ nthstring\or
12842   \nNamexix\ nthstring\fi}
12843

```

`\tensordinalname` Get the ordinal name of a tens number (20 – 90) (Mathew Dafilis (mpd@swin.edu.au) sent Email on 2003/11/14 saying that `\ordinaltoname` didn't work for 20, 30, etc. He was correct. It is now fixed.

```

12844 \newcommand*{\tensordinalname}[2]{%
12845   \ifnum #1=\@ne
12846     \teenordinalname{#2}\ifnotnumtonameallcaps\lowernumtonametrue\fi
12847   \else
12848     \ifnum #2> \z@
12849       \ifcase #1
12850       \or
12851       \or \nNamexx
12852       \or \nNamexxx
12853       \or \nNamexl
12854       \or \nNameel
12855       \or \nNameelx
12856       \or \nNameelxx
12857       \or \nNameelxxx
12858       \or \nNameexc
12859       \fi
12860     \ifnotnumtonameallcaps\lowernumtonametrue\fi
12861     \tensunitsep\unitordinalname{#2}

```

```

12862 \else
12863 \ifcase #1
12864 \or
12865 \or \nthNameexx
12866 \or \nthNameexxx
12867 \or \nthNameexl
12868 \or \nthNameel
12869 \or \nthNameelx
12870 \or \nthNameelxx
12871 \or \nthNameelxxx
12872 \or \nthNameexc
12873 \fi
12874 \ifnotnumtonameallcaps\lowernumtonametrue\fi
12875 \fi
12876 \fi}
12877

```

The names of tens ordinals. The use of `\tiethstring` instead of ‘tieth’ saves some tokens.

```

12878 \newcommand*\nthNameexx{\iflowernumtoname t\else T\fi wen\tiethstring}
12879 \newcommand*\nthNameexxx{\iflowernumtoname t\else T\fi hir\tiethstring}
12880 \newcommand*\nthNameexl{\iflowernumtoname f\else F\fi or\tiethstring}
12881 \newcommand*\nthNameel{\iflowernumtoname f\else F\fi if\tiethstring}
12882 \newcommand*\nthNameelx{\iflowernumtoname s\else S\fi ix\tiethstring}
12883 \newcommand*\nthNameelxx{\iflowernumtoname s\else S\fi even\tiethstring}
12884 \newcommand*\nthNameelxxx{\iflowernumtoname e\else E\fi igh\tiethstring}
12885 \newcommand*\nthNameexc{\iflowernumtoname n\else N\fi ine\tiethstring}
12886

```

`\n@me@number` `\n@me@number{⟨number⟩}` is an internal macro to convert a `⟨number⟩` to names.

```

12887 \newcommand*\n@me@number[1]{%
12888 \begingroup
12889 \numdigits{#1}%
12890 \ifminusnumber\minusname\fi
12891 \priornumfalse
12892 %% billions
12893 \ifnum \c@bsm@mctr > \z@
12894 \unitnumbername{\thebsm@mctr}\space
12895 \ifnotnumtonameallcaps\lowernumtonametrue\fi\nNamemmm
12896 \priornumtrue
12897 \fi
12898 %% hundred millions
12899 \ifnum \c@cmsm@mctr > \z@
12900 \ifpriornum\namenumberscomma\fi
12901 \unitnumbername{\thecmsm@mctr}\space
12902 \ifnotnumtonameallcaps\lowernumtonametrue\fi\nNamec
12903 \priornumtrue
12904 \fi
12905 %% tens/units millions
12906 \ifnum \c@xmsm@mctr > \z@

```

```

12907     \ifpriorium
12908         \ifnum\c@cmsm@mctr>\z@\namenumberand\else\namenumbercomma\fi
12909     \fi
12910     \tensnumbername{\thexsm@mctr}{\themsm@mctr}%
12911     \prioriumtrue
12912 \else
12913     \ifnum \c@msm@mctr > \z@
12914         \ifpriorium
12915             \ifnum\c@cmsm@mctr>\z@\namenumberand\else\namenumbercomma\fi
12916         \fi
12917         \unitnumbername{\themsm@mctr}%
12918         \ifnotnumtonameallcaps\lowernumtonametrue\fi
12919         \prioriumtrue
12920     \fi
12921 \fi
12922 \ifnum \c@cmsm@mctr > \z@
12923     \ifpriorium\space\fi
12924     \nNamemm
12925 \else
12926     \ifnum \c@xsm@mctr > \z@
12927         \ifpriorium\space\fi
12928         \nNamemm
12929     \else
12930         \ifnum \c@msm@mctr > \z@
12931             \ifpriorium\space\fi
12932             \nNamemm
12933         \fi
12934     \fi
12935 \fi
12936 %% hundred thousands
12937     \ifnum \c@cksm@mctr > \z@
12938         \ifpriorium\namenumbercomma\fi
12939         \unitnumbername{\thecksm@mctr}\space
12940         \ifnotnumtonameallcaps\lowernumtonametrue\fi\nNamec
12941         \prioriumtrue
12942     \fi
12943 %% tens/units thousands
12944     \ifnum \c@xksm@mctr > \z@
12945         \ifpriorium
12946             \ifnum\c@cksm@mctr>\z@\namenumberand\else\namenumbercomma\fi
12947         \fi
12948         \tensnumbername{\thexksm@mctr}{\theksm@mctr}%
12949         \prioriumtrue
12950     \else
12951         \ifnum \c@ksm@mctr > \z@
12952             \ifpriorium
12953                 \ifnum\c@cksm@mctr>\z@\namenumberand\else\namenumbercomma\fi
12954             \fi
12955             \unitnumbername{\theksm@mctr}%
12956             \ifnotnumtonameallcaps\lowernumtonametrue\fi

```

```

12957         \priornumtrue
12958     \fi
12959 \fi
12960 \ifnum \c@cksm@mctr > \z@
12961     \ifpriornum\space\fi
12962     \nNamem
12963 \else
12964     \ifnum \c@xksm@mctr > \z@
12965         \ifpriornum\space\fi
12966         \nNamem
12967     \else
12968         \ifnum \c@ksm@mctr > \z@
12969             \ifpriornum\space\fi
12970             \nNamem
12971         \fi
12972     \fi
12973 \fi
12974 %% hundreds
12975     \ifnum \c@csm@mctr > \z@
12976         \ifpriornum\namenumberscomma\fi
12977         \unitnumbername{\thecsm@mctr}\space
12978         \ifnotnumtonameallcaps\lowernumtonametrue\fi\nNamec
12979     \priornumtrue
12980 \fi
12981 %% tens/units
12982 \ifmakeordinal
12983     \ifnum \c@xsm@mctr > \z@
12984         \ifpriornum\namenumbersand\fi
12985         \tensordinalname{\thexsm@mctr}{\theism@mctr}%
12986     \else
12987         \ifnum \c@ism@mctr > \z@
12988             \ifpriornum\namenumbersand\fi
12989             \unitordinalname{\theism@mctr}%
12990         \else
12991             \ifpriornum\nthstring\else\unitordinalname{\theism@mctr}\fi
12992         \fi
12993     \fi
12994 \else % not ordinal
12995     \ifnum \c@xsm@mctr > \z@
12996         \ifpriornum\namenumbersand\fi
12997         \tensnumbername{\thexsm@mctr}{\theism@mctr}%
12998     \else
12999         \ifnum \c@ism@mctr > \z@
13000             \ifpriornum\namenumbersand\fi
13001             \unitnumbername{\theism@mctr}%
13002         \else
13003             \ifpriornum\else\unitnumbername{\theism@mctr}\fi
13004         \fi
13005     \fi
13006 \fi % end ifmakeordinal

```



```
13007 \endgroup}
13008
```

`\numtoname` Lowercase all names

```
13009 \DeclareRobustCommand{\numtoname}[1]{%
13010   \makeordinalfalse
13011   \notnumtonameallcapstrue%
13012   \lowernumtonametrue%
13013   \n@me@number{#1}}
13014
```

`\numtoName` Uppercase first letter of first name (all else lowercase).

```
13015 \DeclareRobustCommand{\numtoName}[1]{%
13016   \makeordinalfalse
13017   \notnumtonameallcapstrue%
13018   \lowernumtonamefalse%
13019   \n@me@number{#1}}
13020
```

`\NumToName` Uppercase first letter of all names (all else lowercase).

```
13021 \DeclareRobustCommand{\NumToName}[1]{%
13022   \makeordinalfalse
13023   \notnumtonameallcapsfalse%
13024   \lowernumtonamefalse%
13025   \n@me@number{#1}}
13026
```

`\ordinaltoname` Lowercase all ordinal names

```
13027 \DeclareRobustCommand{\ordinaltoname}[1]{%
13028   \makeordinaltrue
13029   \notnumtonameallcapstrue%
13030   \lowernumtonametrue%
13031   \n@me@number{#1}}
13032
```

`\ordinaltoName` Uppercase first letter of first ordinal name (all else lowercase).

```
13033 \DeclareRobustCommand{\ordinaltoName}[1]{%
13034   \makeordinaltrue
13035   \notnumtonameallcapstrue%
13036   \lowernumtonamefalse%
13037   \n@me@number{#1}}
13038
```

`\OrdinalToName` Uppercase first letter of all ordinal names (all else lowercase).

```
13039 \DeclareRobustCommand{\OrdinalToName}[1]{%
13040   \makeordinaltrue
13041   \notnumtonameallcapsfalse%
13042   \lowernumtonamefalse%
13043   \n@me@number{#1}}
13044
```

31.14 A fix for two column headings

This is from a posting by Donald Arseneau to CTT on 23 April 2001 to fix a problem — ‘When I use `\onecolumn\chapter...` its headline is printed lower on the page than for two-column chapters’

DA and I had previously discussed this in relation to the Index and I had put a hack into the `tocbibind` package to fix the Index.

Donald posted the following.

`\vspace*` gives bad spacing after a pagebreak and `\makechapterhead` starts with `\vspace*`. The biggest problem is the definition of `\@topnewpage`, which is used for the two-column spanning text. Here is a redefinition:

`\@topnewpage`

```

13045 \long\def \@topnewpage [#1]{%
13046   \@nodoocument
13047   \@next\@currbox\@freelist{}}}%
13048   \global \setbox\@currbox
13049     \vbox {%
13050       \break
13051       \prevdepth\z@
13052       \begingroup
13053       \normalcolor
13054       \hsize\textwidth
13055       \@parboxrestore
13056       \col@number \@ne
13057       #1%
13058       \vskip -\dbltextfloatsep
13059       \endgroup
13060       \null % ordinary \baselineskip
13061       \vskip -\topskip
13062   }%
13063   \begingroup %% \showbox\@currbox
13064     \splitmaxdepth\maxdepth \splittopskip\topskip
13065     \setbox\@tempboxa \vsplit\@currbox to \z@
13066   \endgroup %% \showbox\@currbox
13067   \ifdim \ht\@currbox>\textheight
13068     \ht\@currbox \textheight
13069   \fi
13070   \global \count\@currbox \tw@
13071   \@tempdima -\ht\@currbox
13072   \advance \@tempdima -\dbltextfloatsep
13073   \global \advance \@colht \@tempdima
13074   \ifx \@dbltoplist \@empty
13075   \else
13076     \@latexerr{Float(s) lost}\@ehb
13077     \let \@dbltoplist \@empty
13078   \fi
13079   \@cons \@dbltoplist \@currbox
13080   \global \@dbltopnum \m@ne

```

```

13081 \ifdim \@colht<2.5\baselineskip
13082 \latex@warning@no@line {Optional argument of \noexpand\twocolumn
13083     too tall on page \thepage}%
13084 \@emptycol
13085 \if@firstcolumn
13086 \else
13087     \@emptycol
13088 \fi
13089 \else
13090 \global \vsize \@colht
13091 \global \@colroom \@colht
13092 \floatplacement
13093 \fi}
13094

```

The original version of `\@topnewpage` is in `ltouput.dtx`, line 159.

31.15 Time of day

William Adams (2006/08/28) supplied a basis for `\printtime`, which he needed for `\quarkmarks` but I have used one from *TeX for the Impatient* as it saves some counters.

`\m@mcalthm` Calculate the hours and minutes from `\time`.

```

13095 \newcommand*{\m@mcalthm}{%
13096     \count0 = \time \divide \count0 by 60\relax
13097     \count2 = \count0\relax%      the hour
13098     \count4 = \time \multiply\count0 by 60\relax
13099     \advance\count4 by -\count0\relax% the minute
13100     \ifnum\count4<10 \toks1 = {0}% make a leading zero
13101     \else \toks1 = {}%
13102     \fi}

```

`\hmpunct` User format controls for `\printtime`

```

\amname 3103 %%% punctuation, am and pm for \printtime
\pmname 3104 \newcommand*{\hmpunct}{:}% hours minutes separator
13105 \newcommand*{\amname}{am}% ante meridiem
13106 \newcommand*{\pmname}{pm}% post meridiem
13107

```

`\printtime` Print the time of day as 24 hour clock or 12 hour clock `\printtime` prints the
`\printtime*` time par 24 hour clock and `\printtime*` per 12 hour clock.

```

13108 \newcommand*{\printtime}{%
13109     \@ifstar{\m@msprtime}{\m@mprtime}}

```

`\m@mprtime` These implement time printing: `\m@mprtime` as 24 hour clock and `\m@msprtime`
`\m@msprtime` as 12 hour clock

```

13110 \newcommand*{\m@mprtime}{\begingroup
13111     \m@mcalthm

```

```

13112 \number\count2\hmpunct\the\toks1 \number\count4
13113 \endgroup}
13114 \newcommand*{\m@msprtime}{\begingroup
13115 \m@mcalthm
13116 \def\@mpm{\pmname}%
13117 \ifnum\count2<1\relax% early in the morning
13118 \count2=12\relax
13119 \ifnum\count4>0\relax% not midnight
13120 \def\@mpm{\amname}%
13121 \fi
13122 \else
13123 \ifnum\time<721\relax% noon or earlier
13124 \def\@mpm{\amname}%
13125 \else
13126 \ifnum\time>779\relax% 1300 hrs or later
13127 \advance\count2 by -12\relax
13128 \fi
13129 \fi
13130 \fi
13131 \number\count2\hmpunct\the\toks1 \number\count4\ \@mpm
13132 \endgroup}
13133

```

31.16 Sequential sheet (page) numbers

`\c@sheetsequence` Peter Heslin asked for the ability to add a sequential page number (1 for the first page, N for the last page, no matter what value the page counter has) to the trimming marks.

`\c@sheetsequence` is a new counter for pages starting at the beginning and independent of the standard page counter. This should not be reset by anything. The counter increment has to be added to the output routine.

This may also be useful for page N of M numbering.

```

13134 \newcounter{sheetsequence}
13135 \setcounter{sheetsequence}{1}
13136 \renewcommand{\thesheetsequence}{\@arabic\c@sheetsequence}
13137 \g@addto@macro{\@outputpage}{\stepcounter{sheetsequence}}
13138

```

`\c@lastsheet` While we're at it, might as well provide for lastpage and lastsheet counters

```

\c@lastpage 13139 \newcounter{lastsheet}
13140 \setcounter{lastsheet}{0}
13141 \newcounter{lastpage}
13142 \setcounter{lastpage}{0}

```

`\dol@stsheet` These two macros write the values of lastsheet and lastpage to the aux file. They have to be called at the end of the document after a `\clearpage` to flush out any floats.

```

13143 \newcommand{\dol@stsheet}{%

```

```

13144 \if@filesw
13145   \addtocounter{sheetsequence}{-1}%
13146   \immediate\write\@auxout%
13147     {\string\memsetcounter{lastsheet}{\the\c@sheetsequence}}%
13148   \stepcounter{sheetsequence}%
13149 \fi}
13150 \newcommand{\dol@stpage}{%
13151   \if@filesw
13152     \addtocounter{page}{-1}%
13153     \immediate\write\@auxout%
13154       {\string\memsetcounter{lastpage}{\the\c@page}}%
13155     \stepcounter{page}%
13156   \fi}

```

I originally used this:

`\AtEndDocument{\clearpage\dol@stsheet\dol@stpage}` but following the CTT thread *AtEndDocument produces unwanted page break*, 2004/03/11, and in particular Dan Luecking's response I now try as hard as possible to do it right at the end, but even this is not 100% reliable. The only way to ensure reliability is to modify the kernel `\enddocument` which doesn't seem to be a particularly wise thing to do.

```

13157 \AtBeginDocument{\AtEndDocument{\clearpage\dol@stsheet\dol@stpage}}
13158

```

31.17 Leaves per gathering

Traditionally books are assembled in terms of gatherings, or signatures, with perhaps 8 or 16 pages (or leaves) per grouping (a leaf has two pages, recto and one verso). At the request of Alan Ristow code for this has been included in the class.

`\ifcntrmod` `\iscntrmod{⟨counter⟩}{⟨number⟩}` returns `\cntrmod(true|false)` and `\ifnotcntrmod` `\notcntrmod(false|true)` if `⟨counter⟩` is a multiple of `⟨number⟩`.

```

\iscntrmod{
13159 \newif\ifcntrmod
13160 \newif\ifnotcntrmod
13161 \newcommand*{\iscntrmod}[2]{
13162   \@tempcnta=\@nameuse{c@#1}%
13163   \@tempcntb=\@tempcnta
13164   \divide\@tempcnta #2\relax
13165   \multiply\@tempcnta #2\relax
13166   \advance\@tempcntb-\@tempcnta
13167   \ifnum\@tempcntb=0\relax
13168     \cntrmodtrue
13169   \notcntrmodfalse
13170 \else
13171   \cntrmodfalse
13172   \notcntrmodtrue
13173 \fi}
13174

```

`\@memensuresigpages` Output enough (empty) pages to make up a complete final signature.

```

13175 \newcommand*{\@memensuresigpages}{%
13176   \ifnum\@mempagespersig<\@one
13177   \else
13178     \iscntrmod{sheetsequence}{\@mempagespersig}
13179     \ifcntrmod
13180     \else
13181       \clearpage
13182       \pagestyle{empty}
13183       \mbox{}
13184     \loop
13185       \iscntrmod{sheetsequence}{\@mempagespersig}
13186       \ifnotcntrmod
13187       \clearpage
13188       \pagestyle{empty}
13189       \mbox{}
13190     \repeat
13191   \fi
13192 \fi}
13193

```

`\leavespergathering` `\leavespergathering{<num>}` is the user command for specifying that there must be `<num>` leaves per gathering (`2<num>` pages per gathering). For `<num>` more than one the total number of pages output is exactly divisible by `2<num>`. `<num>` less than two (the default) has no effect.

```

13194 \newcommand*{\leavespergathering}[1]{\@memcnta=#1\relax
13195   \ifnum\@memcnta<\tw@
13196     \def\@mempagespersig{-1}%
13197   \else
13198     \multiply\@memcnta \tw@
13199     \edef\@mempagespersig{\@memcnta}%
13200   \fi}
13201 \leavespergathering{0}
13202

```

Finally, make sure that the requested number of pages is output.

```

13203 \AtEndDocument{\@memensuresigpages}
13204

```

32 Initialization

32.1 Words and phrases

This document class is for documents prepared in the English language. To prepare a version for another language, various English words and phrases must be replaced. The English elements that require replacement are defined below in command names.

`\abstractname` This list is for titles of document sections.

`\contentsname` 13205 `\newcommand*{\abstractname}{Abstract}`

`\listfigurename` 13206 `\newcommand*{\contentsname}{Contents}`

`\listtablename` 13207 `\newcommand*{\listfigurename}{List of Figures}`

`\bookname` 13208 `\newcommand*{\listtablename}{List of Tables}`

`\partname` 13209 `\newcommand*{\bookname}{Book}`

`\chaptername` 13210 `\newcommand*{\partname}{Part}`

`\appendixname` 13211 `\newcommand*{\chaptername}{Chapter}`

`\appendixtocname` 13212 `\newcommand*{\appendixname}{Appendix}`

`\appendixpagename` 13213 `\newcommand*{\appendixtocname}{Appendices}`

`\bibname` 13214 `\newcommand*{\appendixpagename}{Appendices}`

`\indexname` 13215 `\newcommand*{\bibname}{Bibliography}`

`\glossaryname` 13216 `\newcommand*{\indexname}{Index}`

13217 `\newcommand*{\glossaryname}{Glossary}`

`\figurename` These are the names and phrases used for general elements.

`\tablename` 13218 `\newcommand*{\figurename}{Figure}`

`\figurerefname` 13219 `\newcommand*{\tablename}{Table}`

`\tablerefname` 13220 `\newcommand*{\figurerefname}{Figure}`

`\pagename` 13221 `\newcommand*{\tablerefname}{Table}`

`\pagerefname` 13222 `\newcommand*{\pagename}{page}`

13223 `\newcommand*{\pagerefname}{page}`

`\bookrefname` More names for referencing.

`\partrefname` 13224 `\newcommand*{\bookrefname}{Book~}`

`\chapterrefname` 13225 `\newcommand*{\partrefname}{Part~}`

`\sectionrefname` 13226 `\newcommand*{\chapterrefname}{Chapter~}`

`\appendixrefname` 13227 `\newcommand*{\sectionrefname}{S}`

13228 `\newcommand*{\appendixrefname}{Appendix~}`

13229

32.2 Date

`\today` This macro uses the TeX primitives `\month`, `\day` and `\year` to provide the date of the L^AT_EX-run.

```
13230 \newcommand{\today}{\ifcase\month\or
13231   January\or February\or March\or April\or May\or June\or
13232   July\or August\or September\or October\or November\or December\fi
13233   \space\number\day, \number\year}
```

32.3 Two column mode

`\columnsep` This gives the distance between two columns in two column mode.

```
13234 \setlength\columnsep{10\p@}
```

`\columnseprule` This gives the width of the rule between two columns in two column mode. We have no visible rule.

```
13235 \setlength\columnseprule{0\p@}
```

32.4 The page style and counters

We use the page style *headings* by default and arabic page numbering.

```
13236 \pagestyle{headings}
13237 \pagenumbering{arabic}
13238
```

We set the sectional counters to zero, the `tocdepth` to one (sections and above listed), the `secnumdepth` to two (sections and above numbered), and `\maxsecnumdepth` to the same.

```
13239 \setcounter{part}{0}
13240 \setcounter{chapter}{0}
13241 \setcounter{tocdepth}{1}
13242 \setcounter{secnumdepth}{2}
13243 \maxsecnumdepth{section}
13244
```

Set the `\linenumberfrequency` to zero to prohibit line numbering and also set the font for line numbers. Can now set the final space for boxed verbatim line numbers.

```
13245 \linenumberfrequency{0}
13246 \linenumberfont{\small\rmfamily}
13247 \settowidth{\bvnumlength}{\vlvnumfont 9999}
13248
```

32.5 Single or double sided printing

Unless the `twoside` wasn't specified, We do not try to make each page of equal height.

```
13249 \if@twoside
13250 \else
13251   \raggedbottom
13252 \fi
```

When the `twocolumn` option was specified we call `\twocolumn` to activate this mode. We try to make each column as long as the others, but call `\sloppy` to make our life easier.

```
13253 \if@twocolumn
13254   \twocolumn
13255   \sloppy
13256   \flushbottom
```

Normally we call `\onecolumn` to initiate typesetting in one column.

```
13257 \else
13258   \onecolumn
13259 \fi
13260
```


32.6 Floats

Here are the implementations of the figure and table environments and their accompanying List of...

figure This is the definition of the actual environment. The form with the * is used for double column figures. We use `\newfloat` to set it. In this class figures are numbered per chapter, but we need to change the default definition of `\thefigure` if a figure is in a pre-numbered chapter.

```
13261 \newfloat[chapter]{figure}{lof}{\figurename}
13262 %%% \kill@lastcounter{lofdepth}
13263 \renewcommand{\thefigure}{\thechapter.\@arabic\c@figure}
13264
```

\listoffigures These macros request that LaTeX produces a list of figures. The LoF heading is added to the ToC unless the starred version is used.

```
13265 \newlistof{listoffigures}{lof}{\listfigurename}
13266 %%% \kill@lastcounter{lofdepth}
```

\l@figure `\l@figure{<title>}{<page>}` typesets the LoF entry for a `\figure` caption heading.

```
13267 \newlistentry[chapter]{figure}{lof}{0}
13268 \cftsetindents{figure}{0em}{2.3em}
13269 % \kill@lastcounter{lofdepth}
13270
```

table The definition for tables is almost identical.

```
13271 \newfloat[chapter]{table}{lot}{\tablename}
13272 %%% \kill@lastcounter{lotdepth}
13273 \renewcommand{\thetable}{\thechapter.\@arabic\c@table}
13274
```

\listoftables These macros request that LaTeX produces a list of tables. The LoT heading is added to the ToC unless the starred version is used.

```
13275 \newlistof{listoftables}{lot}{\listtablename}
13276 %%% \kill@lastcounter{lotdepth}
```

\l@table `\l@table{<title>}{<page>}` typesets the LoT entry for a `\table` caption heading.

```
13277 \newlistentry[chapter]{table}{lot}{0}
13278 \cftsetindents{table}{0em}{2.3em}
13279 % \kill@lastcounter{lotdepth}
13280
```

The subfigure package defines `lofdepth` and `lotdepth` counters. If is not used, then we have to define them. The subfig package replaced subfigure in 2005, and this only defined the counters if they were not previously defined. It now seems sensible to ignore any use of the subfigure package. This makes life a lot simpler.

```
13281 %%%\AtBeginDocument{%
```

```

13282 %%% \ifundefined{c@lofdepth}%
13283 %%%      {\newcounter{lofdepth}\setcounter{lofdepth}{1}}{}
13284 %%% \ifundefined{c@lotdepth}%
13285 %%%      {\newcounter{lotdepth}\setcounter{lotdepth}{1}}{}
13286

```

32.7 The article option

The article option requires changes to the default chapterstyle, and the numbering of floats, etc.

Emanuele Vicentini (2003/07/21) suggested making `\maketitle` more closely match the real article's appearance. Alan Budden³¹ commented on 2003/12/18 that the equation counter should be continuous.

```

13287 \ifartopt
13288   \chapterstyle{article}
13289   \counterwithout{figure}{chapter}
13290   \counterwithout{table}{chapter}
13291   \counterwithout{footnote}{chapter}
13292   \counterwithout{equation}{chapter}
13293   \renewcommand{\chaptername}{}
13294   \renewcommand{\maketitlehookb}{%
13295     \vskip -1.5\topsep\vskip -1.5\partopsep}
13296   \renewcommand{\maketitlehookc}{%
13297     \vskip -1.5\topsep\vskip -1.5\partopsep}
13298 \fi
13299

```

32.8 The ms option

This should be done last as it makes various changes to the defaults.

`\msdoublespacing` These do nothing unless the ms option is used; then they change the
`\mssinglespacing` `\baselinestretch`.

```

13300 \newcommand{\msdoublespacing}{}
13301 \newcommand{\mssinglespacing}{}

13302 \ifmsdoc
13303   \renewcommand{\msdoublespacing}{%
13304     \renewcommand{\baselinestretch}{1.6}\large\normalsize}
13305   \renewcommand{\mssinglespacing}{%
13306     \renewcommand{\baselinestretch}{1.0}\large\normalsize}
13307   \renewcommand{\familydefault}{\cmtt}
13308   \renewcommand{\rmdefault}{\cmtt}
13309   \renewcommand{\sfdefault}{\cmtt}
13310   \renewcommand{\bfdefault}{\m}
13311   \renewcommand{\itdefault}{\n}
13312   \renewcommand{\sldefault}{\n}

```

³¹alan.s.budden@bristol.ac.uk

```

13313 \renewcommand{\scdefault}{n}
13314 \renewcommand{\baselinestretch}{1.6}
13315 \@twocolumnfalse
13316 \onecolumn
13317 \sloppy
13318 \@twosidefalse
13319 \raggedbottom
13320 \pagestyle{plain}
13321 \fi
13322

```

32.9 Emulated packages

Many of the ‘emulations’ are extensions and integration of package facilities. In some cases an ‘emulated’ package just won’t work with the class, so I’ve added it here to prevent it from being loaded. In any case, most of the packages are mine.

```

13323 \EmulatedPackage{abstract}[2008/07/23]
13324 \EmulatedPackage{appendix}[2008/07/23]
13325 \EmulatedPackage{array}[2008/07/23]
13326 \EmulatedPackage{booktabs}[2008/07/23]
13327 \EmulatedPackage{ccaption}[2008/07/23]
13328 \EmulatedPackage{changepage}[2008/07/23]
13329 \EmulatedPackage{chngcntr}[2008/07/23]
13330 \EmulatedPackage{chngepage}[2008/07/23]
13331 \EmulatedPackage{crop}
13332 \EmulatedPackage{dcolumn}[2008/07/23]
13333 \EmulatedPackage{delarray}[2008/07/23]
13334 \EmulatedPackage{enumerate}[2008/07/23]
13335 \EmulatedPackage{epigraph}[2008/07/23]
13336 %%%\EmulatedPackage{framed}[2008/07/23]
13337 \EmulatedPackage{ifmtarg}[2008/07/23]
13338 \ifm@mifetex\EmulatedPackage{ifetex}[2008/07/23]\fi
13339 \ifm@mifluatex\EmulatedPackage{ifluatex}[2008/07/23]\fi
13340 \ifm@mifpdf\EmulatedPackage{ifpdf}[2008/07/23]\fi
13341 \ifm@mifxetex\EmulatedPackage{ifxetex}[2008/07/23]\fi
13342 \EmulatedPackage{index}[2008/07/23]
13343 \EmulatedPackage{makeidx}[2008/07/23]
13344 \EmulatedPackage{moreverb}[2008/07/23]
13345 \EmulatedPackage{mparhack}[2008/07/23]
13346 \EmulatedPackage{needspace}[2008/07/23]
13347 \EmulatedPackage{newfile}[2008/07/23]
13348 \EmulatedPackage{nextpage}[2008/07/23]
13349 \EmulatedPackage{pagenote}[2008/07/23]
13350 \EmulatedPackage{parskip}[2008/07/23]
13351 \EmulatedPackage{patchcmd}[2008/07/23]
13352 \EmulatedPackage{setspace}[2008/07/23]
13353 \EmulatedPackage{shortvrb}[2008/07/23]
13354 \EmulatedPackage{showidx}[2008/07/23]
13355 \EmulatedPackage{tabularx}[2008/07/23]

```

```

13356 \EmulatedPackage{titleref}[2008/07/23]
13357 \EmulatedPackage{titling}[2008/07/23]
13358 \EmulatedPackage{tocbibind}[2008/07/23]
13359 \EmulatedPackage{tocloft}[2008/07/23]
13360 \EmulatedPackage{tocvsec2}[2008/07/23]
13361 \EmulatedPackage{verbatim}[2008/07/23]
13362 \EmulatedPackage{verse}[2008/07/23]
13363

```

32.10 Interaction with the caption package

Although the author of the caption package has, over the years, made it work with many other classes he has not extended it to either recognise or work with the memoir class. Some authors want to use the caption package, hence... Kill changes to the caption macros when the caption package is used. The caption package checks the definitions of the `\@makecaption`, `\caption` and `\@caption` macros. These need to be identical to the definitions in the standard classes for the package to disbelieve that the memoir class is being used.

```

\@makecaption
  \caption13364 %% revert changes to captioning macros if the caption package is used.
  \@caption13365 \AtBeginPackage{caption}{
    13366 \ClassWarningNoLine{memoir}{%
    13367   You are using the caption package with the memoir \MessageBreak
    13368   class. This may cause unexpected or inconsistent \MessageBreak
    13369   results if you use any of memoir's captioning facilities}
    13370
    13371 \long\def\@makecaption##1##2{%
    13372   \vskip\abovecaptionskip
    13373   \sbox\@tempboxa{##1: ##2}%
    13374   \ifdim \wd\@tempboxa >\hsize
    13375     ##1: ##2\par
    13376   \else
    13377     \global \@minipagefalse
    13378     \hb@xt@\hsize{\hfil\box\@tempboxa\hfil}%
    13379   \fi
    13380   \vskip\belowcaptionskip}
    13381
    13382 \def\caption{%
    13383   \ifx\@capttype\@undefined
    13384     \@latex@error{\noexpand\caption outside float}\@ehd
    13385     \expandafter\@gobble
    13386   \else
    13387     \refstepcounter\@capttype
    13388     \expandafter\@firstofone
    13389   \fi
    13390   {\@dblarg{\@caption\@capttype}}%
    13391 }
    13392

```

```

13393 \long\def\caption##1[##2]##3{%
13394   \par
13395   \addcontentsline{\csname ext@##1\endcsname}{##1}%
13396   {\protect\numberline{\csname the##1\endcsname}{\ignorespaces ##2}}%
13397   \begingroup
13398     \@parboxrestore
13399     \if@minipage
13400       \setminipage
13401     \fi
13402     \normalsize
13403     \@makecaption{\csname fnum@##1\endcsname}{\ignorespaces ##3}\par
13404   \endgroup}
13405 }
13406

```

32.11 Interaction with the float package

The float package also defines `\newfloat`, so kill memoir's version when the float package gets used.

```

13407 \AtBeginPackage{float}{\let\newfloat\relax}
13408

```

32.12 Patch file

At the suggestion of Danie Els (DanieEls@sun.ac.za), Dan Leucking (luecking@uark.edu) and others, input a patch file, if one exists, as the final act. (This is preferable to my series of `memfixa.sty`, `memfixb.sty` packages, and so on, for each release of the class.)

```

13409 \IfFileExists{mempatch.sty}{%
13410   \RequirePackage{mempatch}}{}
13411

```

The end of the class definitions.

```

13412 \</class>

```

33 Glossary Makeindex style file

Here is the basic style (configuration) file for Makeindex for use with the default glossary setup.

```

13413 <*gst>
13414 %%%% basic.gst      basic makeindex glossary configuration file for memoir
13415 %%%% Output style parameters
13416 preamble "\begin{theglossary}"
13417 postamble "\n\\end{theglossary}\n"
13418 group_skip "\n\\glossaryspace\n"
13419 item_0     "\n\\glossitem"
13420 delim_0    "{\\memglonum{"

```

```

13421 encap_suffix "}}}"
13422 %%% Input style parameters
13423 keyword "\glossaryentry"
13424
13425 \gst)

```

The end of the configuration file code

References

- [ABH90] Paul W. Abrahams, Karl Berry and Kathryn A. Hargreaves. *TeX for the Impatient*. Addison-Wesley, Reading, Massachusetts, 1990. (Available from CTAN in `info/impatient`)
- [Ars01a] Donald Arseneau. *Titleref package (version 3.1)*. April 2001. (Available from CTAN as `macros/latex/contrib/misc/titleref.sty`)
- [Ars01b] Donald Arseneau. *Chapterbib package (version 1.9)*. September 2001. (Available from CTAN as `macros/latex/contrib/misc/chapterbib.sty`)
- [Ars03] Donald Arseneau. *Framed package (version 0.8a)*. July 2003. (Available from CTAN as `macros/latex/contrib/misc/framed.sty`)
- [ArWi00] Donald Arseneau and Peter Wilson. *The ifmtarg package*. March, 2000. (Available from CTAN in `/macros/latex/contrib/misc`)
- [Car94] David Carlisle. *The delarray package*. March 1994. (Available from CTAN in `/macros/latex/required/tools`)
- [Car98a] David Carlisle. *The enumerate package*. August, 1998. (Available from CTAN in `/macros/latex/required/tools`)
- [Car98b] David Carlisle. *The remreset package*. August, 1998. (Available from CTAN in `/macros/latex/contrib/carlisle`)
- [Car99] David Carlisle. *The tabularx package*. January 1999. (Available from CTAN in `/macros/latex/required/tools`)
- [Car01] David Carlisle. *The dcolumn package*. May 2001. (Available from CTAN in `/macros/latex/required/tools`)
- [Coc02] Steven Douglas Cochran. *The subfigure package*. March, 2002. (Available from CTAN in `/macros/latex/contrib/subfigure`)
- [Dal99] Patrick W. Daly. *Natural Sciences Citations and References*. May, 1999. (Available from CTAN in `/macros/latex/contrib/natbib`)

- [Dow00] Michael J. Downes. *The patchcmd package*. July 2000. (Available from CTAN in `/macros/latex/contrib/patchcmd`)
- [Fai98] Robin Fairbairns. *The moreverb package*. December, 1998. (Available from CTAN in `/macros/latex/contrib/moreverb`)
- [Fai03] Robin Fairbairns. *footmisc — a portmanteau package for customising footnotes in LaTeX*. February 2003. (Available from CTAN in `macros/latex/contrib/footmisc`)
- [Fea03] Simon Fear. *Publication quality tables in L^AT_EX*. March, 2003. (Available from CTAN in `macros/latex/contrib/booktabs`)
- [Fra00] Melchior Franz. *The crop package*. February, 2000. (Available from CTAN in `/macros/latex/contrib/crop`)
- [GMS94] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Companion*. Addison-Wesley Publishing Company, 1994.
- [Knu84] Donald E. Knuth. *The T_EXbook*. Addison-Wesley, Reading, Massachusetts, 1984.
- [Lam94] Leslie Lamport. *L^AT_EX — A Document Preparation System*. Addison-Wesley, Reading, Massachusetts, 1994.
- [LMB99] Leslie Lamport, Frank Mittelbach and Johannes Braams. *Standard Document Classes for LaTeX version 2e*. September, 1999. (Available from CTAN as `/macros/latex/base/classes.dtx`)
- [MC98] Frank Mittelbach and David Carlisle. *A new implementation of LaTeX's tabular and array environment*. May 1998. (Available from CTAN in `/macros/latex/required/tools`)
- [Oos96] Piet van Oostrum. *Page layout in LaTeX*. June, 1996. (Available from CTAN in `/macros/latex/contrib/fancyhdr`)
- [Rah01] Sebastian Rahtz. *Section name references in LaTeX*. January 2001. (Available from CTAN in `/macros/latex/contrib/hyperref`)
- [Rah02] Sebastian Rahtz. *Hypertext marks in LaTeX*. March 2002. (Available from CTAN in `/macros/latex/contrib/hyperref`)
- [Sch98] Martin Schröder. *The everyshi package*. August, 1998. (Available from CTAN in `/macros/latex/contrib/ms`)
- [SRR01] Rainer Schöpf, Bernd Raichle and Chris Rowley. *A new implementation of LaTeX's verbatim and verbatim* environments*. March, 2001. (Available from CTAN in `/macros/latex/required/tools`)

- [Wil99] Peter Wilson. *The tocvsec2 package*. January, 1999. (Available from CTAN in `/macros/latex/contrib/tocvsec2`)
- [Wil00a] Peter Wilson. *The epigraph package*. February, 2000. (Available from CTAN in `/macros/latex/contrib/epigraph`)
- [Wil00b] Peter Wilson. *LaTeX files for typesetting ISO standards*. February, 2000. (Available from CTAN in `/macros/latex/contrib/isostds/iso`)
- [Wil00c] Peter Wilson. *The nextpage package*. February, 2000. (Available from CTAN in `/macros/latex/contrib/misc`)
- [Wil00d] Peter Wilson. *The needspace package*. March, 2000. (Available from CTAN in `/macros/latex/contrib/misc`)
- [Wil01a] Peter Wilson. *The abstract package*. February, 2001. (Available from CTAN in `/macros/latex/contrib/abstract`)
- [Wil01b] Peter Wilson. *The chngpage package*. February, 2001. (Available from CTAN in `/macros/latex/contrib/misc`)
- [Wil01c] Peter Wilson. *The appendix package*. March, 2001. (Available from CTAN in `/macros/latex/contrib/appendix`)
- [Wil01d] Peter Wilson. *The ccaption package*. March, 2001. (Available from CTAN in `/macros/latex/contrib/ccaption`)
- [Wil01e] Peter Wilson. *The chngcntr package*. March, 2001. (Available from CTAN in `/macros/latex/contrib/misc`)
- [Wil01f] Peter Wilson. *The hanging package*. March, 2001. (Available from CTAN in `/macros/latex/contrib/hanging`)
- [Wil01g] Peter Wilson. *The titling package*. March, 2001. (Available from CTAN in `/macros/latex/contrib/titling`)
- [Wil01h] Peter Wilson. *The tocbibind package*. April, 2001. (Available from CTAN in `/macros/latex/contrib/tocbibind`)
- [Wil01i] Peter Wilson. *The tocloft package*. April, 2001. (Available from CTAN in `/macros/latex/contrib/tocloft`)
- [Wil01j] Peter Wilson. *Typesetting simple verse with LaTeX*. August, 2001. (Available from CTAN in `/macros/latex/contrib/verse`)
- [Wil03] Peter Wilson. *ledmac: A presumptuous attempt to port EDMAC and TABMAC to LaTeX*. August 2003. (Available from CTAN in `macros/latex/contrib/ledmac`)
- [Wil07] Peter Wilson. ‘Glisterings’ *TUGboat*, 28(2):229–232, 2007.
- [Wil08] Peter Wilson. ‘Glisterings’ *TUGboat*, 29(2):324–327, 2008.

Index

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

Symbols	8974,	9006,	\@addmarginpar ... 9321
\! .. 11411, 11414, <u>12478</u>	9013,	9018,	\@addspace . 6608, 6609
\# 6513, 6514	9036,	9043,	\@addtofilelist ...
\% 11530, 11786	9048,	9061, 9067	. 163, 11499, 12130
\(..... 237	\@wrrspindexhyp ...		\@addtopreamble ...
\) 237	... <u>9001</u> , <u>9029</u> , <u>9055</u>		... <u>6023</u> , 6066,
* 6504, 6513	\@Alph . 4892, 4940, 5340		6072, 6095,
\- 11978	\@BLTrule .. 6618, <u>6628</u>		6111, 6129,
\/ 226	\@BTcs . 6578, 6580, <u>6581</u>		6136, 6146, 6201
\: <u>12475</u>	\@BTendrule ... 6610,	\@afterheading 3666,	
\= 11979	6623, 6627,	3676, 3767,	
\@ 428, 438, 439, 442,	6634, <u>6636</u> , 6813	4445, 4452,	
457, 462, 9088, 9090	\@BTfns lone <u>6578</u>	4462, 4470,	
\@@ 6269, 6270,	\@BTfns lthree <u>6578</u>	4529, 4549,	
6480, 6482, 6484	\@BTfns ltwo <u>6578</u>	5854, 5860,	
\@@@ 6145, 6148	\@BTnormal ... 6617,	5925, 5982, 8255	
\@@@cmidrule 6667, <u>6669</u>	6619, <u>6622</u> , 6817	\@afterindentfalse .	
\@@@wrindexm@m	\@BTrule 6589, 6595, 3621,	
..... 8977, <u>8979</u>	6601, 6605, <u>6612</u>	4257, 4444,	
\@@BLTrule . 6628, <u>6629</u>	\@BTswitch .. 6617–6620	4461, 4528, 5891	
\@@array 6212, <u>6213</u> , <u>6362</u>	\@Esphack 7083	\@afterindenttrue .	
\@@cmidrule 6664	\@MM .. 9727, 9788, 10781 4255,	
\@@contcaption ... <u>7202</u>	\@Mii 7083	4451, 4469,	
\@@endpbox 6232	\@NCialign . <u>6758</u> , 6770	4548, 8316,	
\@@enum@ ... 5378, <u>5379</u>	\@NCtabarray 6756, <u>6760</u>	8392, 8454, 8517	
\@@glossary . 9202, <u>9203</u>	\@PTchs@def@ult .. <u>5951</u>	\@alph 5338	
\@line 10071, 11689,	\@PoemTitle . 5896, <u>5902</u>	\@ampacol 6228	
11690, 11710, 11720	\@Roman 3358, 3359	\@apppage .. 4896, <u>4900</u>	
\@m@mline <u>11680</u> , 11710	\@Zmaketitle <u>8244</u>	\@argarraycr 6189, <u>6190</u>	
\@par .. 4283, 9099,	\@aboverulesep	\@argtabularcr ... 6233	
11366, 11369, 11372	... <u>6569</u> , 6586,	\@array <u>6162</u> ,	
\@@sidecaption	6592, 6598,	6213, 6363, 6376	
..... <u>7725</u> , <u>7729</u>	6604, 6614, 6615	\@arrayacol 6231	
\@@sidecontcaption .	\@acci 11979	\@arrayclassiv ... 6229	
..... 7863, <u>7864</u>	\@accii 11979	\@arrayclassz 6229	
\@@sidellegend 7910, <u>7911</u>	\@acciii 11979	\@arraycr 6182,	
\@@sidenamedlegend .	\@acol .. <u>6071</u> , 6091,	<u>6186</u> , 6490, 6780	
..... 7888, <u>7889</u>	6093, 6102,	\@arrayleft	
\@@thesubX <u>7047</u>	6105, 6135, 6137	... 6175, <u>6223</u> ,	
\@wrglom@m . 9221, <u>9225</u>	\@acolampacol	6363, 6367, 6368	
\@wrindexhyp <u>8953</u> , <u>9055</u>	.. <u>6071</u> , 6101, 6103	\@arrayparboxrestore	
\@wrindexm@m . 8958,	\@addamp <u>6059</u> , 6073, 6156	
8963, 8966,	6102, 6106, 6199		

- \@arrayright .. 6223, 6363, 6367, 6369
- \@arrayrule . 6129, 6152
- \@arstrut 6172, 6185, 6204, 6772
- \@arstrutbox .. 6158, 6161, 6165, 6185, 6194, 6287, 6288, 6290, 6294, 6295, 6299, 6763
- \@author ... 3252, 3300
- \@auxout 7595, 7601, 7611, 7617, 7629, 8957, 8962, 8965, 9005, 9012, 9017, 9035, 9042, 9047, 9060, 9066, 9220, 12338, 13146, 13153
- \@beginparpenalty 5155, 11362
- \@belowrulesep 6569, 6587, 6593, 6599, 6604, 6609, 6644
- \@biblabel 8803, 8804, 8823
- \@book 3436, 3468
- \@botlist . 10464, 10466
- \@bs@thanks 5551, 5558, 5559
- \@bscontmark 3155, 3232
- \@bsfootnoterule 3196, 3203
- \@bslabeldelim 5491, 5494
- \@bsmarkseries 3151, 3213
- \@bsmtitleempty 3281, 3294
- \@bsonecolfalse .. 5470
- \@bsonecoltrue ... 5462
- \@bsphack 7594, 7610, 7627, 7739, 7869, 8152, 8915, 8917, 8924, 8925, 9199, 9484, 10918, 10958, 11033, 11041, 11047, 11053, 11400, 11555, 12146, 12155, 12337
- \@bspostauthor 3130, 3252
- \@bspostdate 3130, 3254
- \@bsposttitle 3130, 3250
- \@bspreauthor 3130, 3252
- \@bspredate . 3130, 3254
- \@bsprettitle 3130, 3250
- \@bsruninfalse 5468, 5472
- \@bsrunintitle 5493, 5544
- \@bsrunintrue 5473
- \@bsthanksheadpost 3157, 3215
- \@bsthanksheadpre 3157, 3215
- \@bstr@ctlist (environment) 5513
- \@caption 7148, 7760, 13364
- \@capttype 7069, 7072, 7199, 7200, 7237, 7255, 7257, 7259, 7276, 7277, 7281, 7296, 7301, 7304, 7306, 7321, 7324, 7334, 7386, 7388, 7390, 7398, 7400, 7402, 7403, 7431, 7433, 7434, 7445, 7447, 7461, 7477, 7478, 7535, 7584–7586, 7597, 7603, 7613, 7619, 7631, 7755, 7760, 7875, 7880, 7904, 13383, 13387, 13390
- \@cclv 10462, 10463, 10491, 10517, 10518, 10548, 10560, 10572, 10727, 10740, 11266, 11268, 11269, 11275, 11278, 11282, 11728–11730
- \@cellsincolumn 6819, 6887, 6892, 6893, 6911–6913, 6918, 6920
- \@cellstogo ... 6819, 6869, 6874, 6913, 6914, 6916, 6920, 6944, 6956–6958
- \@cftasnum 8292, 8293, 8322, 8398, 8413, 8460, 8478, 8521, 8546
- \@cftasnumb 8292, 8293, 8323, 8399, 8413, 8461, 8478, 8522, 8547
- \@cftbsnum 8292, 8293, 8321, 8397, 8413, 8459, 8478, 8520, 8546
- \@cftl@subfigtab 8585, 8592
- \@cftn@me 8287, 8292, 8324
- \@chapapp 2769, 2785, 2803, 2813, 3370, 3723, 3786, 3854, 3878, 4096, 4103, 4118, 4891, 4939, 7973–7975, 10947, 10951
- \@chapapp@head 8523, 8524, 8546
- \@chapter .. 3626, 3640
- \@chclass 6020, 6024, 6036, 6047, 6084, 6089, 6127

\@chnum 6020, 6025, 6032, 6043, 6111, 6152	\@contcaption . 7200, 7202, 7213, 7880	\@ctualm@m writetoglo 9231
\@chs@def@ult 3721	\@contcshortstyle 7104, 7171	\@currbox 7080, 7081, 9324, 9344, 13047, 13048, 13063, 13065–13068, 13070, 13071, 13079
\@chtocdok 8723, 8728, 8733, 8738, 8743, 8748, 8753, 8758, 8761	\@contcstyle 7101, 7104, 7176, 7182, 7187	\@currentHref . 7600, 7606, 7616, 7622
\@chtodok 8709, 8713, 8718	\@contcwidth 7114, 7165, 7166	\@currentlabel 8153, 8154, 8179, 9651, 9729, 9754, 9798, 9920, 9936, 10095, 10119, 10148, 10162, 10199, 10222, 10250, 10263, 10339, 10358, 10394, 10409, 10604, 10616, 10796
\@cite 8848	\@contcwtrue 7117	\@currenvir 6390, 11478, 11480, 11481, 11949
\@classi . . . 6084, 6151	\@contdelim . . . 7092, 7159, 7170, 7174, 7181, 7186	\@currlist 9323, 10468, 10471
\@classii . . 6084, 6141	\@contfinal . . . 7135, 7159, 7171, 7177, 7183, 7187	\@currname 151
\@classiii 6141	\@contfmark . 7157, 7161	\@currsize . 2985, 3035
\@classix 6124	\@contfnote . 7157, 7160	\@curtab 6064, 6067, 6726, 6738, 6807, 6808, 6811, 6856, 6943
\@classv 6085, 6144, 6153	\@conthangfalse 7124, 7125	\@cviipt 730, 1262, 1272, 1282
\@classvi 6085, 6127, 6134, 6151	\@conthangtrue . . . 7122	\@cxxxpt . 730, 1273, 1283
\@classvii 6086, 6131, 6138	\@contindentfalse 7122, 7125	\@cxxxiipt . . 730, 1284
\@classviii . 6086, 6126	\@contindenttrue . 7124	\@date 3254, 3301
\@classx 6087, 6088, 6099, 6108	\@contindw . 7121, 7182	\@dbfloat 7058
\@classz . . . 6084, 6108	\@contkeep 7384, 7400, 7431, 7447, 7461	\@dblarg 4233, 7237, 9483, 13390
\@clsextension 193, 195, 198	\@contmidbi 7128, 7278, 7297, 7322, 7335	\@dbldeferlist 10477, 10482, 10484
\@clubpenalty 8814	\@contnfont . . . 7095, 7159, 7170, 7174, 7181, 7186	\@dblfloat 7040, 7062, 7063
\@cmidla 6569, 6669– 6671, 6674, 6688	\@contpost . 7128, 7191	\@dblfloatplacement 10481
\@cmidlb 6569, 6630, 6673, 6674, 6688, 6694	\@contpre 7128, 7169, 7175, 7180, 7185	\@dblfpbot 1645
\@cmidrule 6664	\@contset 7384, 7430, 7460	\@dblfpsep 1645
\@cmidrulea . 6671, 6687	\@contsubcstyle 7339, 7579	\@dblfpstop 1645
\@cmidruleb 6632, 6672, 6687	\@conttfont . . . 7098, 7159, 7171, 7177, 7183, 7187	
\@colht 10467, 10479, 10505, 12357, 13073, 13081, 13090, 13091	\@csin 12284	
\@colroom 10467, 12357, 13091	\@csinstar 12284	
\@combinefloats . 10500	\@csout 12291	
\@cons . . 7081, 9323, 9324, 9875, 13079	\@csoutstar 12291	
\@contbotsubfalse 7468, 7472	\@ctabularstar . . . 6724	
\@contbotsubtrue 7392, 7438, 7454		

- \@dbltoplist 10477, 10478, 13074, 13077, 13079
- \@dbltopnum 13080
- \@deferlist . . . 7081, 10464, 10473, 10474
- \@del@array . . 6364, 6365
- \@depth 6167, 6195, 6765, 9105, 9374, 11947
- \@dischyph 11978
- \@docclearpage . . . 10458
- \@doendpe . . . 11390, 11526, 11715, 11751, 12194, 12224
- \@dotsep 8188
- \@dottedtocline . . 8905
- \@eha 12277, 12280
- \@ehb 10470, 13076
- \@ehc 2136, 5723, 5762, 5768, 5774, 5822, 6241, 7670, 7695, 7854
- \@ehd 417, 419, 459, 461, 1773, 1812, 1900, 2060, 2065, 2152, 2165, 2176, 2189, 2200, 2209, 9548, 10674, 13384
- \@elt 6407, 6409, 12267, 12270
- \@emptycol 13084, 13087
- \@emulated@package 136, 137, 144
- \@enGroup . . . 5356, 5366
- \@enLab 5348, 5352, 5355–5357, 5380, 5383, 5384, 5389
- \@enLabel 5350, 5360–5364
- \@enOther . . . 5357, 5368
- \@enQmark 5349, 5380, 5382
- \@enSpace 5354
- \@enSpace . . . 5354, 5365
- \@enThe 5351, 5380, 5382, 5385
- \@endbook 8925, 8951, 8970, 9023, 9053, 9063, 9069, 9208, 9223, 9508, 10956, 10959, 11038, 11045, 11051, 11057, 11401, 11565, 12152, 12160, 12340
- \@endcolumnactions 6867, 6907, 6954, 7008
- \@endfloatbox 7077
- \@endparpenalty . . . 5155
- \@endpart 3580, 3590, 3597, 4909, 4918
- \@endpbox 6075, 6119–6121, 6161
- \@enhook 5369, 5372
- \@enlab 5348
- \@enloop 5353, 5355–5357, 5358, 5381
- \@enloop@ 5358
- \@entemp 5358, 5360–5367
- \@enum@ 5367, 5378, 5381, 5390, 5391
- \@enumctr 5351, 5352, 5377, 5384–5386, 5391, 5392
- \@enumdepth 5375–5377, 5388
- \@epicenterfalse 7981, 7987
- \@epicentertrue 7977, 7985
- \@epichapapp 7972
- \@epidrop 8014, 8025, 8030, 8034
- \@epipos 7978, 7993, 8022
- \@epirhsfalse 7981, 7985
- \@epirhstrue 7976, 7987
- \@epirule 7942, 7945
- \@episource 7947, 7954, 7959
- \@epitemp 7991, 7996, 8001, 8005, 8020, 8025, 8030, 8034
- \@epitext 7943, 7954, 7959
- \@ept 7979, 7980, 7983, 7984
- \@eqnum 6013
- \@esphack 7607, 7623, 7633, 7739, 7869, 8156, 8924,
- \@evenfoot 2402, 2589, 2640, 8011
- \@evenhead 2402, 2587, 2636, 8009
- \@expast 6228
- \@fbreak 4454, 4455, 4475
- \@file@und 164, 11526, 11750
- \@fileswfalse 203
- \@finalstrut . . . 6161, 9615, 9656, 9804, 9925, 9941, 10621, 10802
- \@firstampfalse 6063, 6105, 6199
- \@firstamptrue 6074
- \@firstcolumntrue 12361
- \@firstofone 354, 6714, 6799, 12353, 13388
- \@firstoftwo 5647, 5648, 6710, 8167, 8183, 12351
- \@float 7039, 7058
- \@floatpenalty 7078, 7083
- \@floatplacement 13092
- \@flushglue 5033, 5065, 5069, 5070, 5078, 5100, 5101, 5111, 5112, 9099, 9423, 11365, 11538
- \@fontswitch 8107, 8119
- \@foot 9734
- \@footgroup 9868
- \@footgroupv@r 9774, 10064, 10086, 10190, 10325

\@footnotemark	9036, 9043, 8852, 8856
330, <u>9645</u> , 9709,	9048, 9061, 9067
9713, 9720, 9856	\@idxitem . . 8891, <u>8906</u>
\@footnotetext 329,	\@ifbothcntrs
330, 6422, 6464,	. 12275, 12286,
<u>9645</u> , 9839,	12288, 12293, 12295
9853, 10082,	\@ifclassloaded . . . 197
10186, 10321, <u>10598</u>	\@iffirstamp 6059
\@footstart 9867	\@ifmtarg 5730,
\@footstartv@r . . . 9774	5741, 7268,
\@for . . . 6858, 6906,	7279, 7288,
6947, 7007, 7540	7299, 7313,
\@formatsecmark@pp <u>4973</u>	10943, 10984,
\@fpbot <u>1585</u>	<u>12349</u> , 12367–
\@fpsep . <u>1585</u> , 1677,	12371, 12375–
1682, 1687,	12379, 12381–
1692, 1697, 1702	12384, 12391,
\@fptop . <u>1585</u> , 1676,	12393, 12405,
1681, 1686,	12407, 12410, 12412
1691, 1696, 1701	\@ifnextchar
\@freelist . . . 9323, 136, 3102,
9324, 10498, 13047	3113, 3626,
\@glossary . 9200, <u>9201</u>	4231, 4236,
\@gobblefour	4239, 5378,
. . 7267, 7287, 7312	5647, 5648,
\@gtempa 6671, 6672,	5668, 5670,
6678, 12247, 12249	5896, 6188,
\@halignto . . . 6171,	6212, 6251,
<u>6207</u> , 6210,	6362, 6589,
6214, 6217,	6595, 6601,
6737, 6748, 6770	6608, 6628,
\@hangfrom . . . <u>105</u> ,	6665, 6666,
3828, 4282, 4431	6725, 7105,
\@hangsubcapfalse 7346	7401, 7432,
\@hangsubcaptrue . 7345	7449, 7463,
\@hds@def@ult <u>4554</u>	7476, 7582,
\@height 2482, 2487,	7723, 7725,
6158, 6166,	7863, 7886,
6539, 6623,	8916, 9200,
6764, 6803,	9202, 9711,
6805, 9103,	9899, 9947,
9105, 9374,	9980, 10827,
11821, 11919, 11947	10839, 10852,
\@highpenalty . <u>1366</u> ,	12233, 12235, 12241
8380, 8442, 8572	\@ifnotmtarg <u>12349</u>
\@idxfile 8946, 8958,	\@ifpackageloaded .
8963, 8966,	189, 7074, 7592,
8975, 8983,	7608, 8592,
8995, 9006,	8850, 9056, 9079
9013, 9018,	\@ifpackagewith . . .
	\@ifstar 2972, 3373,
	3388, 3622,
	4264, 4438,
	4454, 4472,
	4510, 4896,
	4981, 4984,
	5227, 5251,
	5667, 5892,
	6187, 6508,
	7533, 7639,
	8175, 8224,
	10998, 11522,
	11742, 12187,
	12215, 12253,
	12259, 12284,
	12291, 12450, 13109
	\@ifundefined
 146, 168,
	170, 177, 180,
	297, 2412, 4967,
	6245, 7020,
	7031, 7060,
	7064, 7538,
	7539, 8221,
	8297, 8762,
	8939, 8988,
	9204, 11409,
	11968, 11969,
	12057, 12072,
	12089, 12095,
	12101, 12107,
	12228, 12238,
	12249, 12276,
	12279, 13282, 13284
	\@iiiminipage 3104, 3114
	\@iiiparbox 9836
	\@iiminipage 3114
	\@index 8916, <u>8938</u>
	\@indexbox
	. . <u>9082</u> , 9094, 9123
	\@input@ . . . 8927, 9256
	\@itemdepth . . . 5399,
	5400, 5402, 5404
	\@itemitem . 5404, 5407
	\@itemlabel 5171
	\@itempenalty 5155
	\@ivminipage 3103, <u>3112</u>
	\@ivpt <u>730</u> , 1130

- \@kludgeins 10502, 10590, 10755
- \@largefloatcheck 7079
- \@lastchclass 6020, 6025–6028, 6030, 6074, 6089, 6090, 6100, 6126, 6131, 6134
- \@lastruleclass 6569, 6615, 6616, 6637–6644, 6677, 6681, 6683, 6684
- \@latex@error ... 13384
- \@latex@warning .. 8818
- \@latex@warning@no@line 9349, 13082
- \@latexbug 9324
- \@latexerr 10470, 13076
- \@leftidx 9109, 9115, 9118
- \@legend 7237, 7240, 7904
- \@linestogo 6819, 6857, 6859, 6868, 6911, 6915, 6946, 6948, 6955
- \@listI 856, 5300
- \@listdepth 5162, 5165, 5170, 11983, 11986
- \@listi 856, 864, 876, 888, 900, 912, 924, 936, 946, 956, 966, 976, 986, 999, 1011, 1023, 1035, 1047, 1059, 1071, 1081, 1091, 1101, 1111, 1121, 5300
- \@listii 5308
- \@listiii 5308
- \@listiv 5308
- \@listv 5308
- \@listvi 5308
- \@lowpenalty 1366, 5155–5157
- \@lxxxpt 730, 850, 1212, 1228, 1238, 1248, 1258, 1268
- \@lxxiip 730, 1229, 1239, 1249, 1259, 1269, 1279
- \@lxxxivpt 730, 1240, 1250, 1260, 1270, 1280
- \@m@mPoemTitle 5892, 5894
- \@m@mchapter 3622, 3624
- \@m@msPoemTitle 5892, 5969
- \@m@mschapter 3622, 3737
- \@mainaux 210
- \@mainmatterfalse 3380, 3425
- \@mainmattertrue .. 82, 3394
- \@makePoemTitlehead 5924, 5938
- \@makecaption 7154, 7210, 7234, 7251, 13364
- \@makechapterhead 3665, 3672, 3674, 3698
- \@makecol . 10492, 10583
- \@makefcolumn 10473, 10474, 10482, 10484
- \@makefnmark 328, 3214, 9666, 9703, 9861
- \@makefntext .. 3216, 9654, 9690, 9731, 9756, 9802, 10607, 10619
- \@makeoother .. 11290, 11376, 11394, 11557, 11621, 11627, 11628, 12147
- \@makesPoemTitlehead 5981, 5985
- \@makeschapterhead . 3759, 3762, 3764, 3769, 8881, 9158
- \@makesidefnmark .. 10776, 10822
- \@makesidefntext .. 10799, 10886
- \@makespecialcolbox 10503
- \@makesubfloatcaption 7425, 7546
- \@maketitle 3219, 3221, 3228, 3241, 3283
- \@marbox 9323, 9344, 9347, 9356, 9358, 9359, 9361–9363, 9372
- \@maxdepth 1451, 9772, 10516, 10529, 10556, 10568, 10581, 10736, 10750
- \@maybeobeytabs 11324, 11384, 11389, 11522, 11523, 11706, 12187, 12188
- \@medpenalty 1366
- \@mem@gettitle 8158, 8185
- \@mem@bfwarn 8070
- \@mem@calwarn 8102
- \@mem@extranofeet . 9893, 10455, 10456, 10724
- \@mem@extranofeet 10449
- \@mem@itwarn 8078
- \@mem@mitwarn 8114
- \@mem@nestwarn 8159, 8166
- \@mem@nofootfalse . 9894, 10452–10454, 10761–10764
- \@mem@nofoottrue .. 10451, 10760
- \@mem@old@label 8151, 8155
- \@mem@old@ssect 4304, 4308
- \@mem@rmwarn 8046
- \@mem@scap@afterhook 7726, 7775
- \@mem@scap@beforehook 7726, 7744
- \@mem@scwarn 8094

\@mem@sfwarn	8054	\@memfront	3373, 3384	\@memsubcaplabel	7419, 7422
\@mem@slwarn	8086	\@memfront@floats	3374, 3382	\@memsubcaption	7408, 7410, 7414, 7517, 7521
\@mem@testifnofoot	10449, 10460, 10724, 10759	\@memlistsubcaptions	7069, 7072, 7215, 7219, 7535, 7537	\@memsubfig	7450, 7451, 7464, 7465, 7475
\@mem@theTR	8171, 8183	\@memmain	3388, 3406	\@memsubfigcaptionlist	7420, 7421, 7531, 7534, 7540, 7544
\@mem@titlefootkill	3236, 3242	\@memmain@floats	3389, 3404	\@memsubfloat	7477, 7478, 7480
\@mem@titleref	8169	\@memnegtest	2063, 2094, 2095, 2100, 2102, 2109, 2111, 2113, 2115, 2117, 9537	\@memtempa	97, 98, 5354, 5360–5368, 5371, 7154, 7156, 7170, 7177, 7181, 7186
\@mem@titlerefno link	8169	\@memold@caption	7148, 7152	\@memtempb	97, 98
\@mem@ttwarn	8062	\@memoldcaption	7143, 7144	\@memtempc	7275, 7276, 7295, 7296, 7320, 7321, 7333, 7334
\@memb@bchap	8826, 8835, 8854	\@memolddblfloat	7062, 7065	\@memwarn	90, 351, 2061, 2066, 3237, 5567, 5806, 5813, 6247, 6249, 6446, 8043, 8104, 8116, 8763, 8941, 8990, 9383, 9439, 9549, 9571, 10629, 11931, 12069, 12084, 12090, 12092, 12096, 12098, 12102, 12104, 12108, 12110, 12177, 12195, 12225, 12230, 12240, 12251
\@memb@bsec	8826, 8853, 8858	\@memoldedblfloat	7070, 7072	\@memznegtest	2058, 2096–2099, 2101, 2103–2108, 2110, 2112, 2114, 2116,
\@memb@bfloats	3409, 3428	\@memoldefloat	7067, 7069		
\@memcnom	7104	\@memoldfloat	7058, 7061		
\@memcnorm	7105, 7109	\@memoldfonterr	8039, 8051, 8059, 8067, 8075, 8083, 8091, 8099		
\@memcnta	72, 12319, 12320, 13194, 13195, 13198, 13199	\@memoldfontfalse	85		
\@memcontsubbody	7453, 7473	\@memoldfonttrue	690		
\@memcshort	7104	\@memoldfontwarn	8039, 8047, 8055, 8063, 8071, 8079, 8087, 8095		
\@memensuresigpages	13175, 13203	\@memoldlabel	7625, 7628		
\@memerror	90, 305, 418, 460, 749, 766, 1771, 1810, 1898, 2060, 2065, 2133, 2148, 2160, 2172, 2184, 2196, 2206, 2413, 2415, 2672, 2689, 2729, 5723, 5762, 5768, 5774, 5822, 7669, 7694, 7853, 8040, 8110, 8122, 8699, 8768, 9548, 10674, 12277, 12280	\@mempagespersig	13176, 13178, 13185, 13196, 13199		
\@memfakeusepackage	131, 466, 6533–6536, 6704	\@mempnum	2972, 2975		
		\@memptsize	67, 620–632, 748, 753, 755, 762, 765, 767		
		\@memsubbody	7437, 7469		
		\@memsubcap	7402, 7403, 7404, 7433, 7434		

- 7651, 7652,
9531, 9534,
10657, 10660, 10663
\@midlist . 10498, 10499
\@mincolumnwidth . .
... 6827, 6854,
6861, 6862,
6865, 6880,
6945, 6950,
6951, 6953, 6970
\@minipagefalse . .
. 7505, 9833, 13377
\@minipagerestore .
.. 5999, 9761, 9809
\@minipagetrue . 11571
\@minpagerestore . 3130
\@mkboth . . 7992, 8161
\@mkidx 9112,
9129, 9132, 9133
\@mklab 5172
\@mkpream 6074, 6170,
6201, 6768, 6769
\@mparbottom
.. 9345, 9354–9357
\@mpargs 9836
\@mparswitchfalse .
.... 16, 641, 9395
\@mparswitchtrue . .
..... 640, 9393
\@mpfn . . . 330, 6499,
9711, 9717,
9859, 9900,
9901, 9908, 9987
\@mpfootins
... 6004, 9748,
9750, 9794,
9795, 9816,
9826, 9827,
9830, 9869, 9885
\@mpfootinsv@r 9815,
10066, 10114,
10115, 10217,
10218, 10353, 10354
\@mpfootnotetext . .
.... 330, 9793,
9840, 9873,
10087, 10191, 10326
\@mpm 13116,
13120, 13124, 13131
\@mppara@footgroupv@r
..... 10328
\@mpparafootnotetext
.... 10326, 10352
\@mpthreecol@footgroupv@r
..... 10193
\@mpthreecolfootnotetext
.... 10191, 10216
\@mptwocol@footgroupv@r
..... 10089
\@mptwocolfootnotetext
.... 10087, 10113
\@empty 2450, 2454
\@multicnt
.. 6797, 6799–6801
\@multispan . 6798, 6802
\@name@p@xdef . 114,
9902, 9950, 9981
\@name@p@xdf 114
\@name@unresp@xdef .
.... 114, 9909,
9957, 9988, 10005
\@nameedef 125, 5720,
5721, 5740, 5753
\@namelet
. 114, 132, 158,
174, 175, 6715,
6716, 9848,
9871, 11391, 11717
\@namelongdef
.... 114, 9914,
9930, 10133,
10135, 10143,
10156, 10235,
10237, 10245,
10257, 10270,
10373, 10375,
10389, 10403, 10418
\@next 9323, 13047
\@nextchar
.. 6031, 6033–
6035, 6037–
6041, 6044–
6046, 6053,
6081, 6119–
6121, 6145, 6236
\@nmbrlistfalse . . 5173
\@nobreakfalse . . .
.. 8275, 8406, 8468
\@nobreaktrue
.. 4552, 8405, 8467
\@nodocument 8787, 13046
\@noitemerr . 8817, 11374
\@noligs 11357
\@nolnerr 5663
\@normalsize 781
\@notefile
. 10915, 10916,
10953, 10969,
11006, 11008, 11018
\@nowrindex
.. 8945, 8950, 8994
\@numptitlefalse . 5876
\@numptitletrue . . 5876
\@nyptcfile . 747,
748, 750, 752, 757
\@nyptsizeoptfalse . 634
\@nyptsizeopttrue . 634
\@oddfont 2402, 2590,
2642, 8010, 8011
\@oddfont 2402, 2588,
2638, 7995,
8000, 8004, 8009
\@onlypreamble
.... 153–155,
159, 10905, 10909
\@opcol . . . 10474, 10492
\@openbib@code
.. 682, 8807, 8822
\@openleftfalse . . .
.... 79, 671, 673
\@openlefttrue 670, 675
\@openrightfalse . .
..... 669, 673
\@openrighttrue . . .
.... 76, 668, 671
\@outputbox
... 9771, 9773,
10505, 10507,
10508, 10515,
10528, 10530,
10540, 10542,
10548, 10560,
10572, 10719,
10721, 10727, 10740
\@outputpage
.... 10483, 13137
\@pagedp . . . 9364, 9374
\@pageht . . . 9346, 9354

\@para@footgroupv@r 10325	\@preamsidefntext 10779	\@setclcnt 2702, 2716,
\@parafootfmt 10343, 10347, 10362	\@preamthreefmt ... 10205, 10213, 10271	2735, 8645, 8781, 8784,
\@parafootnotetext 10321, 10335	\@preamtwofmt 10102, 10110, 10170	8788, 8790, 8791
\@parboxrestore 7205, 7228, 7246, 9605, 9728, 9753, 9791, 9797, 9935, 10118, 10161, 10221, 10262, 10357, 10408, 10784, 10788, 13055, 13398	\@printpagenotes 10998, 11014	\@setclcntok .. 8646, 8650, 8655, 8660, 8665, 8670, 8675, 8680, 8685, 8690, 8695, 8698
\@part 3527, 3559	\@ptsize 67, 619, 621- 632, 755, 771, 772, 774, 2999, 3017	\@setfontsize 784, 790, 796, 802, 808, 814, 820, 826, 832, 838, 844, 850, 860, 872, 884, 896, 908, 920, 932, 942, 952, 962, 972, 982, 995, 1007, 1019, 1031, 1043, 1055, 1067, 1077, 1087, 1097, 1107, 1117, 1130, 1132, 1134-1141, 1144-1152, 1155-1162, 1164, 1170- 1176, 1178, 1179, 1186- 1191, 1193- 1195, 1203- 1207, 1209- 1212, 1221- 1229, 1232- 1240, 1243- 1251, 1254- 1262, 1265- 1273, 1276-1284
\@pbreak 4438, 4439, 4475	\@reargdef 6252	\@setminipage 7207, 7230, 7248, 11992, 12005, 12025, 13400
\@pfbreak .. 4472, 4473	\@reinserts 10587, 10724	\@setrulekerning 6629, 6646, 6676
\@pfbreakgap 4510, 4511	\@removefromreset . 12265, 12293, 12295	\@setsize 3033
\@pkgextension 145- 147, 149, 150, 158, 185, 187, 190	\@resets@pp . 4934, 4945	\@settopoint 2001, 2030,
\@pnumwidth ... 8188, 8279, 8365, 8368, 8432, 8438, 8498, 8567	\@resets@ppsub 4950, 4962	
\@pprestoresec 4922, 4946	\@restonecolfalse 3265, 8200, 8208, 8870, 9147	
\@ppsveapp . 4922, 4946	\@restonecoltrue 3263, 8198, 8206, 8872, 9149	
\@ppsvesec . 4922, 4936	\@reversemarginfalse 9400, 9402	
\@preamble 6023, 6074, 6171, 6172, 6182, 6204, 6223, 6770, 6773, 6784, 6789, 6884, 6889, 6896, 6904, 6975, 6981, 6984, 6993, 6996, 7004	\@reversemargintrue 9398, 9404	
\@preamerr ... 6047, 6094, 6095, 6097, 6127, 6132, 6200, 6234	\@rightidx ... 9109, 9115, 9118, 9120	
\@preamfntext . 9649, 9783, 9918, 10094, 10147, 10198, 10249, 10338, 10393, 10614	\@rightskip .. 5092- 5094, 11538-11540	
	\@roman 5339	
	\@ruledmarks 2849, 2886	
	\@sPoemTitle 5970, 5980	
	\@sanitize 8944, 8948, 8976, 8993, 8998, 9206, 9212, 9228, 10919, 10958, 11620, 11621, 11629	
	\@sanitizeat 9095	
	\@sapppage . 4896, 4910	
	\@sbook 3436, 3491	
	\@schapter . 3738, 3754	
	\@secCNTformat 105, 4275, 4432, 4434, 4436, 4967	
	\@secondoftwo ... 12351	
	\@secpenalty 4262	

2031,	2038,	\@sprintpagenotes .	6680,	6682,
2046,	2226–2228 10998, <u>11002</u>	6813,	6858,
\@setupbook .	3435, <u>3441</u>	\@sptheidx ... 8996,	6860,	6906,
\@setuppart ...	3526,	9008, 9015,	6907,	6947,
3532,	4901,	9019, 9038,	6949,	7007,
4911, 4991, 5001		9045, 9049, 9067	7008,	7540–
\@sfbreak		\@sptoken ... 5365, 6581	7542,	7678,
.. 4454, <u>4455</u> , 4478		\@sssect 4265, <u>4304</u>	7679,	7683,
\@sharp	<u>6055</u> ,	\@star@or@long	7687,	7691,
6075,	6095,	.. 358, 369, 12227	7845,	7846,
6178,	6203,	\@startpbox ... 6075,	7849,	7850,
6777,	6886,	6119–6121, <u>6155</u>	8163,	8164,
6891,	6897,	\@startsection	9259,	9260,
6899,	6977, 104, <u>4245</u> ,	9264,	9268,
6982,	6986,	4312, 4335,	9272,	11434,
6994, 6997, 7000		4358, 4385, 4407	11436, 11796, 11797	
\@shortsubcapfalse	7346	\@starttoc . 8240, <u>8264</u>	\@tempb	6517,
\@shortsubcaptrue	7344	\@subcapfont .. <u>7377</u> ,	6519,	6522,
\@shortvrbinfo	11583,	7549, 7572, 7575	6650,	6651,
11596, 11601, <u>11612</u>		\@subcaplabelfont .	6655,	6656,
\@showidx 8951, 8955,		... <u>7377</u> , 7549,	7660,	7661,
9003,	9033,	7568, 7574, 7641	7663,	7664,
9059, 9065, <u>9093</u>		\@subcapsize .. <u>7377</u> ,	7666,	7667,
\@sidecaption 7723, <u>7724</u>		7548, 7568, 7579	7678,	7679,
\@sidecontcaption .		\@svsec	7682,	7683,
.... 7861, <u>7862</u>		4275, 4282, 4294	7686,	7687,
\@sidefootnotemark .		\@svsechd	7690,	7691,
. 10815, 10830,		\@t@mp	7779,	7780,
10836, 10843, 10849		\@tabacol	7786,	7787,
\@sidefootnotetext .		\@tabarray	7845,	7846,
. 10786, 10830,		.. 6211, <u>6212</u> , 6222	7850,	9259,
10836, 10855, 10861		\@tabclassiv	9260,	9263,
\@sidempfn .. <u>10773</u> ,		\@tabclassz	9264,	9267,
10828, 10833, 10858		\@tabular	9268, 9271, 9272	
\@sidenamedlegend <u>7885</u>		.. 6214, 6217, <u>6218</u>	\@tempc 11449, 11462,	
\@sidepar	<u>9483</u>	\@tabularcr	11472, 11474,	
\@smemfront		\@tempa	11475, 11478, 11486	
.. 3373, <u>3378</u> , 3385		2417, 2418,	\@tempcntb	
\@smemmain		2682, 2685,	13163, 13166, 13167	
.. 3388, <u>3393</u> , 3407		2694, 6148,	\@tempd 6234, 6239, 6240	
\@smempnum		6149, 6399,	\@tempdimc	
.. 2972, <u>2973</u> , 2976		6400, 6498,	.. 1815, 1821–	
\@spaces	5383,	6500, 6509–	1823,	1827–
6414, 6478, 6483		6511, 6516,	1829,	1839–
\@spart 3527, <u>3582</u>		6520, 6610,	1841,	1845–
\@spbreak		6623, 6627,	1847,	1852–
.. 4438, <u>4439</u> , 4478		6634, 6637–	1854,	1864,
\@spfbreak . 4472, <u>4473</u>		6642, 6649,	1866,	1870,
\@spfbreakgap 4510, <u>4531</u>		6651, 6656,	1872,	1883,
\@spindex .. 8917, <u>8987</u>		6660, 6679,	1886,	1892,

1895,	1903,	\@thesubX	7047	\@twocolfootfmt . . .	
1933,	1942,	\@thisruleclass . . .		10098, 10109, 10122	
1952,	1958,	. . . 6569, 6588,		\@twocolfootnotetext	
1965,	4474,	6594, 6600,	 10082, 10091	
4475,	4477,	6604, 6609,		\@twocolumnfalse . .	
4478,	9447–	6614, 6637–6642	 642, 13315	
9449,	9451–	\@thisrulewidth 6569,		\@twocolumntrue . . .	643
9453,	9580–	6613, 6623,		\@twosidefalse	
9582,	9584–	6625, 6626,	 641, 13318	
9586,	10684–	6675, 6680,		\@twosidetrue	640
10686, 10688–10690		6689, 6691,		\@undefined	
\@tempkipa . . .	4254,	6695, 6697, 6811		254, 277, 320, 13383	
4256,	4257,	\@threecol@footgroupv@r		\@unexpandable@protect	
4262, 4278, 4279	 10190		8158, 10927, 12331	
\@tempskipb	7510	\@threecolfootfmt .		\@verbatim . . 11359,	
\@temptokena 361, 364,		10202, 10212, 10225		11384, 11389,	
372, 375, 429,		\@threecolfootnotetext		11525, 11712,	
441, 6076, 6082,	 10186, 10195		11748, 12192, 12221	
6255, 6264,		\@tightsubcapfalse 7363		\@verbfootnotetext .	
6265, 6278,		\@tightsubcaptrue 7373	 6000,	
11435, 11442,		\@title 3250, 3299		9713, 9720, 9722	
11447, 11453,		\@tocrmarg . . . 8188,		\@verbinstre@m 12190,	
11460, 11470, 11484		8280, 8312,		12205, 12206, 12222	
\@testpach . 6024, 6083		8313, 8388,		\@verbmpfootnotetext	
\@textbottom . . 9128,		8389, 8450,	 6000, 9747	
9131, 9136, 10510		8451, 8512,		\@vobeyspaces	
\@textsubscript . 12487		8513, 8629, 8630		. 11291, 11384,	
\@textsuperscript .		\@toodeep		11523, 11706,	
. 3214,		. . 5163, 5375, 5399		11744, 12188, 12216	
9703, 10000, 10776		\@toplist . 10464, 10465		\@vobeytabs 11304, 11331	
\@texttop		\@topnewpage		\@vsa 5792	
. 9127, 9138, 10506		. 3672, 3762, 13045		\@vsarrayname	
\@tfor . 6081, 6246, 6649		\@topnum . . . 3227, 3619		. . . 5727, 5729,	
\@thanks 3230, 3276, 3298		\@totalleftmargin .		5732, 5739, 5740	
\@thefnmark . . . 3162,	 5180,		\@vsarraytostring .	
9652, 9680,		5181, 6768,	 5782, 5784	
9684, 9686,		11363, 11364,		\@vscentercr 5662, 5685	
9704, 9707,		11541, 11542,		\@vsend 5735, 5738, 5746	
9712, 9718,		11981, 11984,		\@vsicentercr	
9729, 9754,		11988, 11994,		. . 5668, 5670, 5671	
9799, 10096,		12007, 12036,		\@vsifbang . 5647, 5668	
10120, 10200,		12037, 12040,		\@vsifgt . . . 5648, 5665	
10223, 10340,		12045, 12048, 12053		\@vsinext	
10359, 10605, 10617		\@totalrightmargin .		. . 5742, 5745, 5749	
\@thesidefnmark 12036		\@vslnumleft 5651, 5654	
. 10777, 10797,		\@trivlist 5175		\@vslnumright 5650, 5654	
10829, 10834,		\@trplargomm 4230		\@vsp@t 5805,	
10842, 10847,		\@trplargoom 4235, 4266		5807, 5812, 5814	
10854, 10859,		\@twocol@footgroupv@r			
10868, 10872, 10874	 10086			

- `\vspat` 5803,
 5808, 5810,
 5814, 5816, 5819
`\vssptitle` . 5841, 5847
`\vssptitle` . 5841, 5856
`\vsstringtoarray` .
 5735, 5738
`\vstypelinenum`
 5650, 5664
`\vstypeptitle`
 . . 5853, 5859, 5862
`\vsxcentercr`
 . . 5667, 5668, 5669
`\whilesw`
 6077, 10474, 10483
`\width` 2482,
 2487, 6158,
 6168, 6195,
 6766, 9103,
 9105, 9374,
 11821, 11919, 11947
`\wrglom@m`
 . . 9214, 9216, 9219
`\wrindex@m` 8949, 8953
`\wrspindex` . 8999, 9001
`\x@sf` 9639,
 9642, 9662,
 9668, 9965,
 9971, 10818, 10824
`\xaddvskip` 7510
`\xargarraycr` 6191, 6193
`\xarraycr` . 6187, 6188
`\xcmidrule` . 6679, 6680
`\xcvipt` . 730, 1251,
 1261, 1271, 1281
`\xfloat` 3093
`\xfootnote` . . 329, 9852
`\xfootnotemark`
 329, 9855
`\xfootnotenext`
 6422, 9858
`\xfootnotetext` . . 330
`\xhline` 6302
`\xifmtarg` 12351–12353
`\xlviipt` . . . 730,
 844, 982, 1195,
 1211, 1227,
 1237, 1247, 1257
`\xobeysp` . . . 11293,
 11295, 11300,
 11533, 11545, 11549
`\xobeytab` . . 11298,
 11306, 11313, 11314
`\xssect` 4302
`\xsidefootnote`
 10828, 10831
`\xsidefootnotemark`
 10840, 10844
`\xsidefootnotetext`
 10853, 10856
`\xtabularcr` 6233
`\xtrplargomm` 4230
`\xtrplargoom` 4235
`\xverbfootnote` . . 9711
`\xxtrplarg` . 4230, 4237
`\xxxpt` 730,
 832, 962, 1107,
 1164, 1178,
 1193, 1209,
 1225, 1235, 1278
`\xxxvipt` . 730, 838,
 972, 1117, 1179,
 1194, 1210,
 1226, 1236, 1246
`\yargarraycr` 6192, 6193
`\[` . . . 235, 11411, 11414
`\{` 233, 11414
`\}` 233, 11414
`\]` . . . 235, 11411, 11414
`\^` 11305, 11306, 11394,
 11415, 11558, 12148
`\‘` 11979
`\|` 31, 11785, 11786, 11963
`\~` 6318,
 11415, 11588, 11607
`\sq` 2769, 2781,
 2785, 2786,
 2803, 2813,
 2854, 2903,
 3937, 4096,
 4144, 4968,
 6504, 10989,
 11190, 11191,
 11292, 11293,
 11295, 11549, 13131

A

`A` (environment) . . 12043
`\abnormalparskip` . 3116
`\abovecaptionskip` .
 . . . 7138, 7155,
 7273, 7293,
 7318, 7331, 13372
`\abovecolumnspenalty`
 . . 6819, 6882, 6972
`\abovedisplayshortskip`
 786, 792,
 798, 804, 810,
 816, 822, 828,
 834, 840, 846,
 852, 862, 874,
 886, 898, 910,
 922, 934, 944,
 954, 964, 974,
 984, 997, 1009,
 1021, 1033,
 1045, 1057,
 1069, 1079,
 1089, 1099,
 1109, 1119, 3087
`\abovedisplayskip` .
 785, 791,
 797, 803, 809,
 815, 821, 827,
 833, 839, 845,
 851, 855, 861,
 873, 885, 897,
 909, 921, 933,
 943, 953, 963,
 973, 983, 991,
 996, 1008, 1020,
 1032, 1044,
 1056, 1068,
 1078, 1088,
 1098, 1108,
 1118, 1126, 3085
`\aboverulesep` . 6548,
 6592, 6598, 6677
`\abovetopsep` 6548, 6586
`\abs@leftindent` 5482,
 5497, 5501, 5519
`\abscolnamefont`
 5477, 5503
`\abscoltextfont`
 5477, 5504
`\abslabeldelim` . . . 5491

- \absleftindent 8235, 8831, 4016, 4026,
- 5482, 5497 8887, 9164, 13395 4053, 4069,
- \absnamepos \added 11040 4090, 4111,
- . . 5480, 5505, 5539 \addlinespace 6607, 6642 4158, 4171,
- \absparindent \addperiod 4620, 4657, 4183, 4200, 4212
- . . 5482, 5510, 5517 4663, 4689, \afterchapternum . .
- \absparsep 4695, 4748, . . . 3704, 3721,
- . . 5482, 5511, 5521 4754, 4845, 3791, 3805,
- \absrightindent 4851, 4878, 4884 3817, 3829,
- . . 5482, 5502, 5520 \addtocontents 3714, 3847, 3886,
- \abstitlekip . 5480, 3715, 8194, 3894, 3913,
- 5494, 5506, 5540 8623, 8633, 3939, 3941,
- abstract (environ- 8635, 8637, 3959, 3979,
- ment) 5535 8643, 8712, 4023, 4024,
- \abstractcol 5470 8717, 8722, 4043, 4045,
- \abstractintoc . . . 5470 8727, 8732, 4057, 4079,
- \abstractname 8737, 8742, 4100, 4113,
- . . . 5494, 5529, 8747, 8752, 8757 4148, 4166,
- 5533, 5539, 13205 \addtocounter . 7017, 4189, 4206, 4218
- \abstractnamefont 5475, 7199, 7277, \afterchaptertitle .
- 5494, 5503, 5539 7875, 13145, 13152 3710,
- \abstractnum 5470 \addtodef 179, 3721, 3776,
- \abstractrunin . . . 5470 182, 358, 2757, 8851 3796, 3870,
- \abstracttextfont 5475, 369, 3309, 3883, 3898,
- . . 5475, 5504, 5544 3317, 3327, 5551 \addtoiargdef 3954, 3972,
- \active 9088, 11292, 4025, 4088, 8257
- 11293, 11305, \afterepigraphskip .
- 11306, 11394, . . . 7925, 7956, 7971
- 11415, 11558, \afterevery@verbatim
- 11588, 11594, 11285, 11379
- 11605, 11607, 12148 \aftergroup
- \add@bstotocfalse 5464 . . 424, 9736, 11266
- \add@bstotoctrue . 5471 \afterparaskip
- \add@special 4388, 4392
- 11584, 11616 \afterpartskip
- \addappheadtotoc 5354, . . . 3528, 3597, 4574
- . . 4902, 4920, 4963 6320, 6469, \afterPoemTitle . .
- \addcontentsline 6470, 6511, . . 5949, 5951, 5991
- 3473, 6579, 6583, 11282 \afterPoemTitlenum .
- 3477, 3564, \afterbookskip 5943, 5951
- 3568, 3680, . . . 3437, 3506, 4563 \afterPoemTitleskip
- 3684, 3689, \afterchapskip 3717, 5957, 5962
- 4286, 4296, 3729, 3735, \afterpoemttitleskip
- 4921, 4964, 3795, 3796, 5865, 5871
- 5003, 5529, 3811, 3823, \aftersecskip 4315, 4319
- 5849, 5929, 3871, 3884, \aftersubparaskip .
- 5933, 7301, 3944, 3955, 4410, 4414
- 7304, 7542, 3973, 3976, \aftersubsecskip . .
- 4338, 4342

- \aftersubsubsecskip 4361, 4365
 - \afterZtitle 8256
 - \aliaspagestyle 2603,
2837–2839,
3234, 3274,
3522, 3523,
3613, 3927, 8894
 - \allowbreak . . . 9610,
9621, 10111,
10172, 10214,
10273, 10793, 10809
 - \alsoname 9073
 - \altindentfalse 5614,
5711, 5831, 5836
 - \altindenttrue . . . 5709
 - altverse (environ-
ment) 5708
 - \amname
13103, 13120, 13124
 - \anappendixfalse . .
. 4886, 4946
 - \anappendixtrue . . .
. 4886, 4945
 - \and 3296, 3321
 - \andnext 3278, 3297, 3322
 - \anyptfilebase 634, 747
 - \anyptsize 634, 747, 748
 - appendices (environ-
ment) 4944
 - \appendix 4886
 - \appendixname
. . . 4891, 4939,
4966, 4974, 13205
 - \appendixpage 4895
 - \appendixpage* . . . 4895
 - \appendixpagename .
. . . 4903, 4904,
4908, 4912,
4913, 4917, 13205
 - \appendixrefname . .
. 8139, 13224
 - \appendixtocname . .
. . 4921, 4964, 13205
 - \Aref 8139
 - \array 6209
 - \arraybackslash . . 6490
 - \arraycolsep 5993, 6209
 - \arrayrulewidth . . .
. . . 5995, 6289,
6296, 6304, 6807
 - \arraystretch . 6166,
6167, 6764, 6765
 - \arraytostring
. . . 5779, 5805, 5812
 - \artoptfalse 665
 - \artopttrue 666
 - \AtBeginClass 192
 - \AtBeginDocument 210,
2287, 3203,
5510, 6545,
7074, 7591,
8592, 8849,
8980, 9055,
9079, 9237,
10934, 13157, 13281
 - \AtBeginDvi 2285
 - \AtBeginFile
. 177, 185, 193
 - \AtBeginPackage 184,
12028, 13365, 13407
 - \atcentercr 5025
 - \AtEndClass 192
 - \AtEndDocument
. 8268, 13157, 13203
 - \AtEndFile 177,
187, 190, 195, 198
 - \AtEndOfClass 681
 - \AtEndPackage 184
 - \atendtheglossaryhook
. 9169, 9171
 - \author 3285, 3317
 - \autocols 6834
 - \autorows 6922
- B**
- \b@vdocount 11658, 11688
 - \b@vdoinside
. 11668, 11692
 - \b@vdooutside
. 11668, 11691
 - \b@vtop 11638, 11731
 - \backmatter 3415
 - \backup@length 6284,
6287–6289,
6291, 6294–6297
 - \barlength
. . . 4121–4123, 4125
 - \baselineskip . 2005,
2007, 2071,
2073, 2078,
2079, 2081,
2087, 2088,
2090, 2993,
3036, 3037,
3040, 3041,
3044, 3047,
3049, 3053,
3055, 3056,
3061, 3064,
3124, 3125,
3956, 4069,
4073, 4081,
4089, 4484,
5055, 6180,
6779, 7926,
7928, 8613,
9101, 10330,
10383, 11532,
11705, 11801,
11809, 11810,
11836, 13060, 13081
 - \baselinestretch . .
. 1291, 2538,
2984, 3037,
3038, 3043,
3096, 9646,
9796, 9916,
9933, 10092,
10116, 10145,
10159, 10196,
10219, 10247,
10260, 10336,
10355, 10391,
10406, 10601,
10611, 10789,
13304, 13306, 13314
 - \beforebookskip . . .
. . . 3437, 3454, 4561
 - \beforechapskip 3717,
3722, 3733,
3783, 3784,
3809, 3810,
3821, 3822,
3908, 3931,
3956, 3961,
3974, 3988,
4007, 4015,

- 4052, 4070, `\belowdisplayskip` . `\booknamenum`
 4072, 4081, 855, 991, 1126, 3086 . . . 3460, 3485, 4568
 4124, 4131, `\belowrulesep` . 6548, `\booknumberline` . . .
 4133, 4140, 6587, 6593, 6684 3474, 3380
 4156, 4157, `\bf` 8070 `\booknumberlinehook`
 4170, 4182, `\bfdefault` 13310 3380
 4193, 4199, 4211 `\bibindent` 683, 684, 8795 `\booknumfont`
`\beforeepigraphskip` . . 7925, 7952, 7963 `\bibintoc` 8840 . . . 3456, 3462,
`\beforeparaskip` 4387, 4392 `bibitemlist` (environ- 4565, 4569, 4625
`\beforepartskip` 3528, 3545, 4572 `\bibitemsep` . 8797, 8799 `\bookpageend` 3506
`\beforePoemTitleskip` 5952, 5962 `\biblistextra` 8799, 8811 `\bookpagemark` 3336, 3480
`\beforepoemtitleskip` 5863, 5871 `\bibmark` 2776, 2810, `\bookrefname` 8140, 13224
`\beforesecskip` 4314, 4319 2831, 2862, `\booktitlefont`
`\beforesubparaskip` 4409, 4414 2909, 3345, 8828 . . . 3456, 3463,
`\beforesubsecskip` 4337, 4342 `\bibname` 2776, 4566, 4570, 4626
`\beforesubsubsecskip` 4360, 4365 `\bottomfraction` . . 2395
`\begin` 30 `\bottomrule` . 6585, 6639
`\begintheglossaryhook` 9168, 9171 `\bottomsectionskip` . . . 4227, 4247, 4249
`\belowbottomsep` 6548, 6599 `\bouderrorfalse` 5714, 5759
`\belowcaptionskip` 7138, 7196, `\bouderrortrue` 5764, 5770, 5776
 7274, 7294, `\bicaption` 7310 `\boxedverbatim` . . 11709
 7319, 7332, 13380 `\bicontcaption` . . . 7328 `\boxedverbatim*` . 11709
`\belowdisplayshortskip` 787, 793, `\bigskipamount` 1324, `\boxedverbatim@input`
 799, 805, 811, 9630, 9844, 11741
 817, 823, 829, 10044, 10060, 10442 `\boxedverbatiminput`
 835, 841, 847, `\binding` 1924, 1931, 11741
 853, 863, 875, 1936, 1937, `\boxmaxdepth`
 887, 899, 911, 1940, 1945, 1946 . . . 9772, 10516,
 923, 935, 945, `\bionenumcaption` . 7285 10529, 11862, 11864
 955, 965, 975, `\bitwonumcaption` . 7265 `\boxverb@botpage`
 985, 998, 1010, `\bktabrule` . . . 6538, 11723, 11730
 1022, 1034, 6625, 6626, `\boxverb@split`
 1046, 1058, 6689, 6691, 11640, 11685, 11727
 1070, 1080, 6695, 6697, 6811 `\boxverb@toprule`
 1090, 1100, `blockdescription` (en- 11707, 11719, 11731
 1110, 1120, 3088 `vironment)` . 5418 `\boxverbflag`
`\blockdescriptionlabel` 5420, 5422 11631, 11703, 11727
`\book` 3434 `\break` . . . 4503, 5076,
`\book*` 3434 12455, 12464, 13050
`\bookblankpage` . . . 3501 `\Bref` 8139
`\bookname` 3460, 4567, 13205 `\brokenpenalty` . . . 1376
`\booknamefont` 3456, 3460, `\bs` 213
 4564, 4567, 4624 11699, 11705, 11739

\bvcountinsidefalse	\c@bvlinectr	\c@lrightskip
..... 11666 11642, 11646	... 6827, 6846,
\bvcountinsidettrue	\c@chapter	6851, 6891,
.... 11664, 11665	... 3350,	6897, 6934,
\bvcountlinesfalse	3360, 3866,	6939, 6982,
..... 5573, 5579	3904, 3950,	6986, 6994, 6997
\bvcountlinestrue	3968, 4176,	\c@m@morig@ctr
..... 5573, 5582	4892, 4940, 6011	.. 12298
\bvendofpage	\c@chrsinstr	\c@memfbvline
.... 11722 5717, 5731,	11642, 11651, 11652
\bvendrulehook	5734, 5739,	\c@memfvline
11735, 11751,	5740, 5744,	... 5561, 5594, 5595
11759, 11766,	5781, 5785, 5787	\c@memmarkcntra
11773, 11780, 12224	\c@cksm@mctr	... 72
\bvleftsidehook	. 12523,	\c@modulo@vs
... 11689, 11720,	12583, 12586, 5561
11735, 11757,	12652, 12937,	\c@mpfootnote
11764, 11771, 11778	12946, 12953, 12960 9870
\bvnumbersinside	\c@cmsm@mctr	\c@msm@mctr
11663	. 12523,	.. 12523,
\bvnumbersoutside	12604, 12607,	12590, 12593,
11663	12628, 12899,	12644, 12913, 12930
\bvnumlength	12908, 12915, 12922	\c@newflo@tctr
11670, 11676, 13247	\c@cp@cntr 7018, 7019
\bvperpagefalse 12313	\c@page
11762, 11769, 11776	\c@csm@mctr	717, 2841, 2974,
\bvperpagetrue	. 12518, 12562,	2976, 2980,
.... 11636, 11755	12565, 12676, 12975	2981, 9118,
\bvrightsidehook	\c@dbltopnumber	11164, 11244,
... 11699, 11720,	.. 2399	12322, 12325,
11735, 11758,	\c@enumi	12432, 12436,
11765, 11772, 11779 5337	12440, 12444, 13154
\bvslides	\c@enumii	\c@pagenote
.... 11775 5338 10900
\bvtopandtail	\c@enumiii	\c@paragraph
... 11768 5339	3350, 3367
\bvtopmidhook	\c@enumiv	\c@part
.... 11720, 11735 5340, 8803, 8810 3350, 3359
\bvtopofpage	\c@equation	\c@poem
.... 11638 6011 5881
\bvtoprulehook	\c@figure	\c@poemline
11719, 11735, 11756, 13263	. 5561, 5588
11763, 11770, 11777	\c@footnote	\c@secnumdepth
	... 5553,
	9631, 9846, 9847	... 2706, 2720,
	\c@ism@mctr	2739, 2746,
	2767, 2780,
	. 12518, 12548,	2801, 2852,
	12551, 12707,	2902, 3349,
	12721, 12987, 12999	3471, 3484,
	\c@ksm@mctr	3562, 3575,
	.. 12518,	3656, 4270,
	12569, 12572,	4287, 4297, 4975
	12668, 12951, 12968	\c@section
\C	\c@lastpage	. 3350, 3361
11416 13139	\c@sheetsequence
\c@@memmarkcntra	\c@lastsheet	..
... 2703, 2706, 13139 13134, 13147
2717, 2720,	\c@lleftskip	\c@sidefootnote
2736, 2739, 2746 10768, 10846
\c@book	... 6827, 6845,	\c@storedpagenu
.... 3350, 3358	6848, 6891,	number
\c@bottomnumber	6897, 6933,	.. 2978, 2980, 2981
.. 2394	6936, 6982,	
\c@bsm@mctr	6986, 6994, 6997	
....		
. 12523, 12611,		
12614, 12621, 12893		

\c@subparagraph . . .	\captionstyle . <u>7101</u> ,	\cftbeforebookskip .
. <u>3350</u> , 3369	<u>7104</u> , 7717, 7719 8381, <u>8415</u>
\c@subsection <u>3350</u> , 3363	\captiontitlefinal <u>7135</u>	\cftbeforechapterskip
\c@subsubsection . .	\captiontitlefont <u>7098</u> 8509, <u>8549</u>
. <u>3350</u> , 3365	\captionwidth <u>7114</u>	\cftbeforepartskip .
\c@table 13273	\cardinal <u>12691</u> 8443, <u>8480</u>
\c@tocdepth	\catcode 6513, 6514,	\cftbeforeXskip . . <u>8332</u>
. . 8383, 8445, 8507	9088, 11290,	\cftbookafterpnum <u>8415</u>
\c@topnumber <u>2392</u>	11292, 11293,	\cftbookaftersnum .
\c@totalnumber . . . <u>2396</u>	11305, 11306, 8398, <u>8415</u>
\c@vslineno	11394, 11411,	\cftbookaftersnumb .
. . <u>5561</u> , 5673, 5824	11415, 11558, 8399, <u>8415</u>
\c@workm@mctr	11586, 11588,	\cftbookbreak <u>8380</u>
. . <u>12528</u> , 12543,	11594, 11603,	\cftbookdotsep . . . <u>8415</u>
12545, 12547–	11605, 11607,	\cftbookfillnum . . .
12552, 12554–	11785, 11786, 8403, <u>8415</u>
12559, 12561–	11963, 12148, 12350	\cftbookfont
12566, 12568–	\centerfloat <u>5083</u>	. . 8395, 8402, <u>8415</u>
12573, 12575–	\centering 2553, 2563,	\cftbookformatpnum <u>8415</u>
12580, 12582–	2573, 2583,	\cftbookformatpnumhook
12587, 12589–	3481, 3495, <u>8415</u>
12594, 12596–	3572, 3586,	\cftbookindent 8386,
12601, 12603–	3865, 3878,	8387, 8392, <u>8415</u>
12608, 12610–12614	3882, 3909,	\cftbookleader . . . <u>8415</u>
\c@X <u>7020</u> , <u>8297</u>	3917, 3924,	\cftbookname
\c@xksm@mctr . <u>12518</u> ,	3925, 3936,	. . <u>8378</u> , 8395, 8413
12576, 12579,	3940, 3943,	\cftbooknumwidth . .
12660, 12944, 12964	3948, 3965, 8396, <u>8415</u>
\c@xmsm@mctr . <u>12523</u> ,	4107, 4144,	\cftbookpagefont . <u>8415</u>
12597, 12600,	4147, 4165,	\cftbookpresnum . . .
12636, 12906, 12926	4167, 4175, 8397, <u>8415</u>
\c@xsm@mctr . . <u>12518</u> ,	4177, 4457,	\cftchapterafterpnum
12555, 12558,	4465, 4703, <u>8549</u>
12684, 12706,	4708, 4713,	\cftchapteraftersnum
12720, 12983, 12995	4718, 4733, 8521, <u>8549</u>
\c@Zdepth <u>7030</u>	4738, 4743,	\cftchapteraftersnumb
\cal <u>8102</u>	4905, 4914, 8522, <u>8549</u>
\calccentering . . <u>12418</u>	4994, 5006,	\cftchapterbreak . .
\cancelhanksrule . <u>3195</u>	5868, 5960, 8508, <u>8572</u>
\cancelthanksrule 3196	5961, 6842,	\cftchapterdotsep <u>8549</u>
\caption <u>7143</u> , 7254,	7162, 7164, 7166	\cftchapterfillnum .
7256, 7258,	\centerlastline . . <u>5068</u> 8529, <u>8549</u>
7268, 7269,	\centerline . . . 4096,	\cftchapterfont . . .
7279, 7280,	4101, 11913,	. . 8524, 8528, <u>8549</u>
7288, 7289,	11925, 11926, 11935	\cftchapterformatpnum
7313, 7314, <u>13364</u>	\cftaddnumtitleline <u>8549</u>
\captiondelim <u>7092</u> , 7233 <u>8636</u>	\cftchapterformatpnumhook
\captionnamefont . .	\cftaddtitleline . <u>8634</u> <u>8549</u>
. <u>7095</u> , 7715	\cftappendixname . .	\cftchapterindent .
\captionsize <u>1288</u> <u>8537</u> , 8540 8510,

- 8511, 8516,
8549, 8625, 8626
\cftchapterleader 8549
\cftchaptername ...
..... 8504, 8533
\cftchapternumwidth
..... 8525,
8549, 8627, 8628
\cftchapterpagefont
..... 8549
\cftchapterpresnum .
..... 8520, 8549
\cftdot 8281
\cftdotfill
... 8281, 8358,
8426, 8491, 8560
\cftdotsep . 8284, 8357
\cftinsert 8640
\cftinsertcode ... 8640
\cftinserthook ... 8640
\cftlocalchange .. 8632
\cftnodots ... 8284,
8427, 8492, 8561
\cftpagenumbersoff 8594
\cftpagenumberon 8598
\cftparfillskip ...
..... 8286, 8596
\cftparskip
.. 8216, 8238, 8247
\cftpartafterpnum 8480
\cftpartaftersnum .
..... 8460, 8480
\cftpartaftersnumb .
..... 8461, 8480
\cftpartbreak 8442
\cftpartdotsep ... 8480
\cftpartfillnum ...
..... 8465, 8480
\cftpartfont
.. 8457, 8464, 8480
\cftpartformatpnum 8480
\cftpartformatpnumhook
..... 8480
\cftpartindent 8448,
8449, 8454, 8480
\cftpartleader ... 8480
\cftpartname
.. 8440, 8457, 8478
\cftpartnumwidth ..
..... 8458, 8480
\cftpartpagefont . 8480
\cftpartpresnum ...
..... 8459, 8480
\cftsetindents 8374,
8575, 8577,
8579, 8581,
8583, 8587,
8589, 13268, 13278
\cftwhatismyname . 8309
\cftXafterpnum ... 8352
\cftXaftersnum ... 8352
\cftXaftersnumb .. 8352
\cftXdotsep 8352
\cftXfillnum 8362
\cftXfont 8352
\cftXformatpnum .. 8362
\cftXformatpnumhook
..... 8362
\cftXindent 8334
\cftXleader 8352
\cftXname 8352
\cftXnumwidth 8334
\cftXpagefont 8352
\cftXpresnum 8352
\cftZafterlisthook 8225
\cftZbeforelisthook
..... 8225
\ch@ngetext
12356, 12372, 12380
\ch@pt@c 3624,
3644, 3648, 3650
\changepcaptionwidth
..... 7114
\changed 11052
\changeglossactual 9240
\changeglossnum .. 9240
\changeglossnumformat
..... 9240
\changeglossref .. 9240
\changemarks 11029
\changemarksfalse .
.... 11028, 11030
\changemarkstrue 11029
\changepage 12374
\changes 2269
\changetext 12366
\changetocdepth ...
... 8712, 8717,
8722, 8727,
8732, 8737,
8742, 8747,
8752, 8757, 8778
\chapindent ... 3842,
3846, 3849,
3986, 3989, 3996
\chapnamefont
... 3723, 3730,
3785, 3786,
3853, 3854,
3875, 3878,
3909, 4018,
4022, 4024,
4030, 4038,
4042, 4054,
4114, 4118,
4173, 4174, 4194
\chapnumfont
... 3725, 3730,
3788, 3789,
3803, 3804,
3815, 3816,
3827, 3828,
3841, 3842,
3846, 3855,
3866, 3876,
3879, 3886,
3903, 3910,
3912, 3924,
3934–3936,
3941, 3949,
3958, 3966,
3967, 3978,
3984, 3986,
3998, 4055,
4063, 4076,
4082, 4096,
4098, 4103,
4115, 4131,
4162, 4163,
4165, 4174,
4175, 4187,
4188, 4195,
4204, 4205,
4207, 4208,
4216, 4217, 4219
\chapter 3615,
5533, 8827,
8876, 9153, 10996
\chapterheadstart .
..... 3699,

- 3721, 3770, 4698, 4728, \chs@crosshead ... 4155
 3784, 3810, 4762, 4794, \chs@culver 3919
 3822, 3890, 4822, 4855, 13288 \chs@dash 3929
 3987, 4007, \chapttitlefont 3728, \chs@demo 3863
 4073, 4094, 3730, 3793, \chs@demo2 3946
 4112, 4157, 8256 3794, 3814, \chs@demo3 3963
 \chaptermark 3815, 3826, \chs@dowding 4169
 ... 2765, 2799, 3827, 3841, \chs@ell 3982
 2826, 2850, 3860, 3867, \chs@ger 4005
 2900, 3336, 3869, 3880, \chs@hangnum 3839
 3667, 3669, 3744 3882, 3915, \chs@komalike 4181
 \chaptername 3917, 3942, \chs@lyhne 4013
 3370, 13205, 13293 3943, 3951, \chs@madsen 4028
 \chapternamenum ... 3953, 3969, \chs@ntglike 4192
 ... 3703, 3721, 3971, 3985, \chs@pedersen 4048
 3787, 3802, 4003, 4010, \chs@reparticle .. 3819
 3813, 3825, 4019, 4032, \chs@section 3799
 3844, 3892, 4056, 4064, \chs@southall 4066
 3933, 3994, 4074–4076, \chs@tandh 4198
 4020, 4034, 4087, 4105, \chs@thatcher 4092
 4042, 4060, 4107, 4116, \chs@veelo 4109
 4078, 4097, 4163, 4177, \chs@verville 4138
 4119, 4161, 4190, 4196, \chs@wilsondob ... 4210
 4186, 4203, 4215 4207, 4208, 4219 \citeindexfile ... 9078
 \chapternumberline . \char 213, \cl@ckpt 6408
 .. 3681, 3685, 8543 233, 235, 237, \ClassError ... 90, 6235
 \chapternumberlinehook 238, 9090, 11530 \ClassInfo . 270, 11613
 8543 \chardef .. 11785, 12516 \ClassWarning 91, 11000
 \chapterprecis ... 8604 \checkandfixthelayout \ClassWarningNoLine
 \chapterprecishere 2269 246, 293,
 8605, 8607 \checkarrayindex .. 314, 6492, 13366
 \chapterprecistoc . .. 5752, 5755, 5758 \cleardoublepage ..
 8606, 8622 \CheckCommand 6302 2841,
 \chapterrefname ... \checkifinteger ... 3272, 3379,
 8142, 13224 5791, 5810 3398, 3403,
 \chapterstyle . 3779, \checkoddpages . 7815, 3420, 3443, 3534
 3797, 3800, 9294, 9322, \clearforchapter ..
 3808, 3820, 9485, 12315, 12403 . 672, 674, 676,
 3832, 3840, \checkstre@eof ... 3617, 3926, 8868
 3852, 3864, 12163, 12168 \clearmark 2758
 3874, 3889, \checkstre@mnoteof . \clearpage ... 674,
 3902, 3907, 12162 712, 716, 2841,
 3920, 3921, \checkthelayout ... 3272, 3400,
 3930, 3947, 2093, 2273 3417, 3422,
 3964, 3983, \chs@article 3807 3445, 3536,
 4006, 4014, \chs@bianchi 3873 10487, 10493,
 4029, 4051, \chs@bringhurst .. 3888 12431, 12432,
 4068, 4093, \chs@brotherton .. 3901 12439, 12440,
 4110, 4139, \chs@chappell 3906 13157, 13181, 13187
 4634, 4667, \chs@companion ... 3849 \clearplainmark .. 2758

- \cleartoevenpage [12430](#)
- \cleartooddpage . [12438](#)
- \cleartorecto . [672](#), [712](#)
- \cleartoverso . [676](#), [716](#)
- \cline [6808](#)
- \closein . . [11504](#), [12142](#)
- \closeinputstream [12139](#)
- \closeout . . . [11006](#),
 [11018](#), [11564](#), [12124](#)
- \closeoutputstream .
 [12121](#)
- \clubpenalty
 . . [1370](#), [8813](#), [8814](#)
- \cmd [239](#)
- \cmdprint [239](#)
- \cmidrule
 . . [6640](#), [6664](#), [6680](#)
- \cmidrulekern
 . . [6548](#), [6652](#), [6657](#)
- \cmidrulesep [6548](#), [6682](#)
- \cmidrulewidth
 [6548](#), [6665](#)
- \cmrkern@l [6647](#), [6657](#),
 [6658](#), [6689](#),
 [6690](#), [6695](#), [6696](#)
- \cmrkern@r [6648](#), [6652](#),
 [6653](#), [6692](#), [6698](#)
- \cmrsideswitch
 . . [6653](#), [6658](#), [6660](#)
- \cntrmodfalse . . . [13171](#)
- \cntrmodtrue [13168](#)
- \CodelineIndex [13](#)
- \col@number . [3218](#), [13056](#)
- \col@sep [6071](#),
 [6209](#), [6220](#), [6753](#)
- \color@begingroup .
 [9653](#),
 [9730](#), [9738](#),
 [9755](#), [9800](#),
 [9922](#), [9938](#),
 [10097](#), [10121](#),
 [10150](#), [10164](#),
 [10201](#), [10224](#),
 [10252](#), [10265](#),
 [10342](#), [10361](#),
 [10397](#), [10412](#),
 [10521](#), [10532](#),
 [10606](#), [10618](#), [10798](#)
- \color@endgroup [9657](#),
 [9740](#), [9743](#),
 [9805](#), [9834](#),
 [9926](#), [9942](#),
 [10099](#), [10123](#),
 [10152](#), [10166](#),
 [10203](#), [10226](#),
 [10254](#), [10267](#),
 [10344](#), [10363](#),
 [10399](#), [10414](#),
 [10525](#), [10536](#),
 [10608](#), [10622](#), [10800](#)
- \colorbox
 [12014](#), [12024](#), [12047](#)
- \colorchapnum [4048](#)
- \colorchaptitle . . [4048](#)
- \columnsep [1974](#), [2251](#),
 [2316](#), [8880](#),
 [9157](#), [12360](#),
 [12371](#), [12379](#), [13234](#)
- \columnseprule [1975](#),
 [2251](#), [2316](#),
 [8879](#), [9156](#), [13235](#)
- \columnwidth
 . . . [4488](#), [9110](#),
 [9366](#), [9368](#),
 [9451](#), [9584](#),
 [9628](#), [9650](#),
 [9728](#), [9752](#),
 [9767](#), [9797](#),
 [9919](#), [9935](#),
 [10118](#), [10161](#),
 [10221](#), [10262](#),
 [10357](#), [10408](#),
 [10615](#), [10688](#),
 [11817](#), [11989](#),
 [11993](#), [12006](#),
 [12037](#), [12050](#),
 [12358](#), [12360](#), [12363](#)
- \commentsoff [11402](#)
- \commentson [11405](#)
- \contcaption
 . . . [7198](#), [7298](#),
 [7323](#), [7330](#), [7336](#)
- \contentsline [8635](#), [8638](#)
- \contentsname . [2773](#),
 [2787](#), [2807](#),
 [2814](#), [2859](#),
 [2870](#), [2906](#),
 [2915](#), [8278](#), [13205](#)
- \continuousmarks . [3155](#)
- \continuousnotenums
 [10904](#)
- \contsubbottom . . . [7453](#)
- \contsubcaption . . [7427](#)
- \contsubtop [7467](#)
- \copy [6290](#), [6299](#)
- \copypagestyle
 [2418](#), [2607](#)
- \count [110](#), [111](#), [9516](#),
 [10057](#), [10061](#),
 [10084](#), [10140](#),
 [10188](#), [10242](#),
 [10323](#), [10381](#),
 [10639](#), [13070](#),
 [13096](#)–[13100](#),
 [13112](#), [13117](#)–
 [13119](#), [13127](#), [13131](#)
- \count@ [6049](#),
 [6049](#), [6050](#),
 [6052](#), [6053](#),
 [6058](#), [6078](#),
 [6109](#), [6141](#),
 [6146](#), [6149](#),
 [6274](#), [6276](#),
 [6277](#), [6320](#),
 [6322](#), [6344](#),
 [6345](#), [6350](#), [6470](#)
- \counterwithin
 . [3390](#), [3391](#), [12284](#)
- \counterwithin* . [12284](#)
- \counterwithout [3375](#),
 [3376](#), [3410](#),
 [3411](#), [10907](#),
 [12291](#), [13289](#)–[13292](#)
- \counterwithout* [12291](#)
- \cplabel
 [12313](#), [12318](#), [12319](#)
- \crrc [6223](#), [6788](#)
- \createmark . . . [2681](#),
 [2785](#), [2786](#),
 [2813](#), [2868](#),
 [2869](#), [2913](#), [2914](#)
- \createplainmark . .
 . . [2661](#), [2787](#)–
 [2792](#), [2814](#)–
 [2819](#), [2870](#)–
 [2875](#), [2915](#)–[2920](#)
- \Cref [8139](#)
- \crtok [6819](#),
 [6872](#), [6908](#),

- 6959, 7009, 7010
`\cs` 238, 2269, 2270, 4558
`\CT@arc@` 6545,
6626, 6691, 6697
`\ctableftskip` . 6721,
6727, 6730,
6739, 6742,
6758, 6900,
6924, 6927, 7001
`\ctabrightskip`
. . . 6721, 6728,
6733, 6740,
6745, 6774,
6905, 6925,
6930, 7005, 7006
`\ctabsetlines` . 6783,
6815, 6835, 6923
`\ctabular` 6738
`\ctabular*` 6724
`\ctt` 27
`\currentgroup` 6708
`\currenttitle` 8169
- D**
- `\D` 6355
`\d@llarbegin` . . 6113,
6116, 6117,
6146, 6207,
6210, 6220, 6754
`\d@llarend` 6115–6117,
6146, 6207,
6210, 6221, 6755
`\date` 3286, 3327
`\dblfloatpagefraction`
. 2401
`\dblfloatsep` 1537
`\dbltextfloatsep` . .
1537, 13058, 13072
`\dbltopfraction` . . 2400
`\DC@` 6317, 6355
`\DC@centre` . 6323, 6328
`\DC@end` 6329, 6347, 6355
`\DC@endcentre` 6329, 6333
`\DC@endright` 6347, 6354
`\DC@right` . . 6325, 6339
`\DC@rl` . 6342, 6344, 6353
`\DC@x` 6320, 6321
`\DeclareOldFontCommand`
. . . 8048, 8056,
8064, 8072,
8080, 8088, 8096
`\DeclareRobustCommand`
. 217, 232, 234,
236, 238, 3160,
7638, 8106,
8118, 8126,
8175, 8177,
8594, 8598,
8624, 8778,
12315, 12356,
12366, 12374,
12476, 12478,
12481, 12483,
12487, 13009,
13015, 13021,
13027, 13033, 13039
`\DeclareTextFontCommand`
. 8134
`\defaultaddspace` . .
. 6548, 6608
`\defaultlists` 5207, 5305
`\defaultsecnum` . . . 4433
`\deleted` 11046
`\DeleteShortVerb` . .
. 31, 11598
`\DescribeMacro` 10
description (environ-
ment) 5412
`\descriptionlabel` .
. 5414, 5416
`\dimen` 2261,
9085, 9521–
9523, 9547,
9785, 10056,
10062, 10075–
10077, 10083,
10139, 10187,
10241, 10322,
10330–10333,
10380, 10383–
10386, 10645–
10648, 10673, 10705
`\dimen@` . 6216, 6217,
6348, 6350–
6352, 6370–
6374, 6435–
6437, 6443–
6445, 6451,
6736, 6737,
10507, 10509,
11835–11837,
11847–11849,
11851, 11852,
11856, 11866,
11867, 11875,
11877, 11883–
11886, 12446–
12448, 12452,
12454, 12458, 12460
`\dimen@ii` 6344, 6349,
6350, 11879–
11881, 11886,
11888, 11891,
11892, 12453,
12454, 12459–12461
`\discretionary`
5045, 11533, 11545
`\DisemulatePackage` . 157
`\displaywidowpenalty`
. 1372
`\dlfm@m@endpart` . . .
. . . 4999, 5011, 5013
`\dlfm@mappage`
. . . 4982, 4985, 4990
`\dlfm@msappage` . . .
. . . 4981, 4984, 4990
`\do` 6082, 6246,
6649, 6858,
6906, 6947,
7007, 7540,
11376, 11394,
11557, 11619,
11623, 11624, 12147
`\doccoltocetc` 8195
`\DocInput` 34
`\documentclass` 2
`\dol@stpage` 13143
`\dol@stsheets` 13143
`\donemaincaptionfalse`
. . . 7056, 7059, 7063
`\donemaincaptiontrue`
. 7144
`\dospecials`
. 11376, 11394,
11557, 11618,
11619, 11625, 12147
`\doublerulessep` 5996,
6136, 6303,
6308, 6565, 6616

- DoubleSpace (environment) 3075
- \DoubleSpacing 3015, 3077
- \dp 6167, 6194, 6288, 6295, 6370, 6765, 7080, 9356, 9363, 9505, 9609, 9611, 9617, 9622, 9727, 9790, 10296, 10507, 10517, 10707, 10783, 10792, 10794, 10805, 10810, 11728
- \draftdocfalse 644, 647, 656
- \draftdoctrue 650
- \dropchapter 7972
- \droptitle 3208, 3248
- E**
- \easypagecheck 12307
- \edef 222, 333, 339, 346, 363, 374, 428, 439, 457, 2453, 2456, 2460, 2464, 2472, 4491, 5351, 5377, 5384, 5404, 6031, 6244, 6344, 6390, 6408, 6498, 6509, 7018, 7025, 7275, 7295, 7320, 7333, 8163, 8302, 9639, 9662, 9729, 9754, 9965, 10818, 10928, 11480, 11796, 11862, 11961, 12332, 13199
- \eject 11838, 11898, 11920
- \em 28, 8126
- \emergencystretch 12470
- \eminnershape 8126
- \emph 8126, 9073, 9075
- \emptythanks 3276
- \EmulatedPackage 134, 153, 264, 287, 299, 330, 13323–13362
- \EmulatedPackageWithOptions 134, 154
- \EnableCrossrefs 12
- \end 37
- \end@dblfloat 7040, 7067
- \end@float 7039, 7067
- \endarray 6223, 6226
- \endboxedverbatim 11709
- \endboxedverbatim* 11709
- \endboxverbatim 11717
- \endctabular 6787
- \endctabular* 6787
- \endfboxverbatim 11575
- \endframed 11782, 12029
- \enditemize 5410
- \endleftbar 12031
- \endlinechar 11501, 11503, 12199, 12201
- \endMakeFramed 11819, 12011, 12016, 12021, 12026, 12041, 12054
- \endminipage 3107, 9822
- \endshaded 11782, 12030
- \endsidecaption 7753
- \endsidecontcaption 7873
- \endsidelegend 7915
- \endsidenamedlegend 7899
- \endsnugshade 12032
- \endtabular 6226, 6476
- \endtabular* 6226
- \endtabularx 6395, 6467
- \endthebibliography 8849
- \endverbatim 11390, 11576
- \endverbatim* 11390
- \endverbatimoutput 11563
- \endwriteverbatim 12152
- \ensuremath 218, 228, 12482, 12490
- \ensureonecol 8195, 8228
- \enumerate 5374
- environments:
 - @bstr@ctlist 5513
 - A 12043
 - abstract 5535
 - adjustwidth 12386
 - adjustwidth* 12386
 - altverse 5708
 - appendices 4944
 - bibitemlist 8801
 - blockdescription 5418
 - description 5412
 - DoubleSpace 3075
 - epigraphs 7962
 - fboxverbatim 11566
 - figure 13261
 - figure* 13261
 - flexlabelled 5429
 - framed 12009
 - hangparas 5121
 - itemize 5398
 - labelled 5424
 - leftbar 12009
 - midsloppypar 12473
 - onecolabstract 5547
 - OnehalfSpace 3071
 - patverse 5830
 - patverse* 5835
 - qframe 12034
 - quotation 5443
 - quote 5450
 - shaded 12009
 - SingleSpace 3046
 - SingleSpace* 3052
 - snugshade 12009
 - Spacing 3066
 - subappendices 4961
 - subfloat 7435
 - symbols 5454
 - table 13271
 - table* 13271
 - thebibliography 8837
 - theglossary 9145
 - theindex 8867
 - titlingpage 3259
 - verbatimoutput 11553

- verse 5681
- vminipage 3100
- vplace 12424
- writeverbatim 12145
- X 7039
- X* 7039
- \epigraph 7952
- \epigraphflush 7934,
7938, 7953,
7955, 7963,
7970, 7980, 7984
- \epigraphfontsize 7937
- \epigraphforheader 8014
- \epigraphhead 7990
- \epigraphpicture . 8019
- \epigraphposition 7937
- \epigraphrule
.. 7925, 7942, 7945
- epigraphs (environ-
ment) 7962
- \epigraphsize . 7933,
7937, 7953, 7964
- \epigraphsourceposition
..... 7937
- \epigraphtextposition
..... 7937
- \epigraphwidth
.. 7925, 7942,
7944, 7948,
7953, 7958,
7965, 7991, 8021
- \escapechar 12247
- \etexfalse 267
- \etextrue 267
- \eTeXversion
.. 275, 277, 279, 281
- \evensidemargin ...
.. 2010, 2223–
2225, 2228,
2310, 12369, 12377
- \every@verbatim ...
.... 11285, 11359
- \everydisplay 3089
- \everylistparindent
..... 5159,
5217, 5237,
5248, 5261, 5275
- \ext@subX 7047
- \ext@X 7030
- \ext@Z 8220
- \extracolsep 6145, 6148
- \extrafeetendmini .
..... 9814, 9822
- \extrafeetendminihook
..... 9814, 9883
- \extrafeetins 9769
- \extrafeetinshook .
..... 9769, 9877
- \extrafeetminihook .
..... 9809, 9887
- \extrafeetreinshook
..... 9778, 9880
- \extrafontsizesfalse 88
- \extrafontsizestrue 691
- \extrarowheight ...
.. 6164, 6183, 6762
- \extratabsurround .
.. 6282, 6290, 6299
- F**
- \f@@t 9734
- \f@rbdy 3643, 3754, 10940
- \f@rbody 3640
- \f@rhdr 3640,
3740, 3742,
3748, 3750, 3752
- \f@rtoc 3640, 10939
- \f@t 9734
- \familydefault .. 13307
- \fancybreak 4454
- \fb@adjheight
.. 11814, 11838,
11899, 11921, 11955
- \fb@afterframe 11833,
11854, 11907, 11939
- \fb@frh 11790,
11845, 11846,
11849, 11859,
11891, 11893, 11943
- \fb@frw ... 11790, 11943
- \fb@put@frame
.... 11826, 11829
- \fb@putboxa .. 11832,
11853, 11906, 11928
- \fb@resto@set
11862, 11898, 11902
- \fb@sizeofframe ...
..... 11789,
11840, 11857, 11943
- \fboxrule 6006,
11966, 11968, 12035
- \fboxsep 6006,
11966, 11969,
12014, 12035, 12044
- \fboxverbatim ... 11566
- fboxverbatim (environ-
ment) 11566
- \fcardinal 12696
- \feetabovefloat . 10583
- \feetbelowfloat . 10583
- figure (environment)
..... 13261
- figure* (environment)
..... 13261
- \figurename 13218, 13261
- \figurerefname
..... 8136, 13218
- \figureversion ... 4082
- \file 22
- \fill 6721, 6722, 6727,
6728, 6739,
6740, 6924, 6925
- \finalhyphendemerits
..... 5133
- \firmlist 5281
- \firmslists 5226
- \firmslists* 5226
- \FirstFrameCommand .
.... 11826, 11965
- \firstthline 6285
- \fixdvipslayout .. 2277
- \fixheaderwidths ..
..... 2229, 2961
- \fixpdflayout 2277
- \fixthelayout 2212, 2274
- \flagverse 5628
- \flegtocX 7034
- \flegX 7034
- flexlabelled (environ-
ment) 5429
- \floatingpenalty ..
.. 9727, 9788, 10781
- \floatpagefraction 2398
- \floatsep 1477, 1562,
1566, 1570,
1574, 1578, 1582
- \flushbottom 9127, 13256
- \flushleft 5127
- \flushleftright .. 5029

- \flushright 9699, 10011, 2206, 2249,
- .. 4114, 4116, 5128 10012, 10015, 2312, 2950, 12384
- \fnsymbol .. 3154, 5558 10017, 10020, 10031
- \fnum@X 7034
- \fnumbersep .. 12506, \footnote 328,
- 12623, 12642, 3237, 3242,
- 12646, 12666, 3310, 3318,
- 12670, 12693, 12703 3328, 7157,
- \fo@t .. 9732, 9734, 9758 7160, 9851,
- \font 223, 224, 8128, 11356 10963, 10965, 10966
- \fontdimen . 6373, 8128
- \footfootmark . 9677, \footnotemark
- 9696, 9863, 329, 3160,
- 10111, 10214, 10350 7158, 7161, 9854
- \footfudgefactor .. 3196, 3204, \footnoterule
- 10297, 10333, 10386 3205, 3260, 328, 3195,
- \footfudgefiddle .. 9626, 9764, \footpagenote . 10961
- .. 10317, 10332 9817, 9829, \fordinal 12716
- \footins 328, 9886, 10047, \foremargin . . . 1928,
- 1465, 3043, 10447, 10523, 10534 1944, 2101,
- 6004, 9630, \footnotesatfoot 10593 2102, 2155,
- 9645, 9723, \footnotesep 2162, 2224, 2243
- 9785, 9843, . . . 1453, 9655, \form@tnumber
- 9845, 10452, 9726, 9731, . 12617, 12694,
- 10520, 10524, 9756, 9789, 12698, 12704, 12718
- 10531, 10535, 9803, 9924, \fps@X 7034
- 10547, 10562, 9940, 10620, 10782 \FrameCommand
- 10575, 10588, \footnotesinmargin . 11789, 11826,
- 10610, 10726, 11965, 12009,
- 10742, 10753, 10761 \footnotetext 12014, 12019,
- \footinsdim 993, 3261, 12024, 12035, 12044
- .. 9783, 10056, 7380, 9083, \framed . . . 11782, 12029
- 10062, 10083, 9724, 9749, framed (environment)
- 10139, 10187, 9783, 10679, 10771 12009
- 10241, 10322, 10380
- \footinsv@r . . . 9774, \footnotetext
- 9779, 10059, 330, 5553,
- 10083, 10084, 6498, 6502, 9857
- 10091, 10187, \footparindent 9671,
- 10188, 10195, 9692, 9701, 10029
- 10322, 10323, \footref . . . 9705, 9862
- 10335, 10453, 10762
- \footmarksep .. 9671, \footruleheight . . .
- 9693, 9700, 10030 . . 2475, 2599, 2968
- \footmarkstyle \footruleskip
- .. 9674, 9698, 9865 . . 2475, 2599, 2968
- \footmarkwidth 9671, \footscript . . . 9674,
- 9678, 9679, 9680, 9684, 9686
- 9681, 9683, \footskip 1377, 1971,
- 9686, 9694, 2044, 2116,
- 2117, 2203, \fref 8136
- 2206, 2249,
- 2312, 2950, 12384
- \foottextfont
- .. 9648, 9655,
- 9783, 9797,
- 9803, 9850,
- 10081, 10093,
- 10111, 10117,
- 10185, 10197,
- 10214, 10220,
- 10303, 10320,
- 10329, 10337,
- 10350, 10356,
- 10603, 10613, 10620
- \footpagenote . 10961
- \fordinal 12716
- \foremargin . . . 1928,
- 1944, 2101,
- 2102, 2155,
- 2162, 2224, 2243
- \form@tnumber
- . 12617, 12694,
- 12698, 12704, 12718
- \fps@X 7034
- \FrameCommand
- . 11789, 11826,
- 11965, 12009,
- 12014, 12019,
- 12024, 12035, 12044
- \framed . . . 11782, 12029
- framed (environment)
- 12009
- \FrameHeightAdjust .
- 11881,
- 11956, 11958, 11973
- \framepichead
- .. 2935, 2956, 2957
- \framepicttextfoot .
- .. 2943, 2958, 2959
- \FrameRestore
- . 11785, 11963,
- 11975, 12010,
- 12015, 12020, 12025
- \FrameRule 11965, 12035
- \FrameSep . . . 11965,
- 12014, 12035,
- 12039, 12044,
- 12046, 12049, 12052
- \fref 8136

- `\frenchspacing` 11384,
 11523, 11706,
 11716, 11742,
 11744, 12188, 12216
`\frontmatter` 3372
`\frontmatter*` 3372
`\Ftrimpicbl` 11099, 11149
`\ftype@X` 7016
`\futurelet` 5358, 6255,
 6580, 6610,
 9732, 9758, 11427
`\futurenonspacel` .
 6578,
 6623, 6627,
 6634, 6679, 6813
- G**
- `\g@addto@macro` 6889,
 6896, 6981,
 6984, 6993,
 6996, 9877,
 9880, 9883,
 9887, 9893, 13137
`\get@vsindent`
 . . 5674, 5802, 5827
`\getarrayelement` . .
 5754, 5803
`\getstar@vsindent` .
 5675, 5820
`\getthelinenumber` .
 5598,
 5657, 5661, 11655
`\glossary` . . 8160, 9199
`\glossarycolsep` . . .
 9157, 9180
`\glossaryentry` . . . 9233
`\glossaryintoc` . . . 9175
`\glossarymark` . 2833,
 2864, 3347, 9161
`\glossaryname`
 . . . 2792, 2819,
 2864, 2875,
 2920, 9153,
 9158, 9164, 13205
`\glossaryrule` 9156, 9180
`\glossaryspace` . . . 9182
`\glossitem` 9249
`\gobm` 5793, 5798
- H**
- `\hangafter` . 5120, 7183
`\hangcaption` 7121
`\hangfrom` 5116
`\hangindent`
 . . . 5118, 5120,
 7176, 7182,
 8906–8908, 9098
`\hangpara` . . 5120, 5122
`hangparas` (environ-
 ment) 5121
`\hangsecnum` 4433
`\hangsubcaption` . . 7342
`\hb@xt@` . 2548, 2558,
 2568, 2578,
 3172, 3179,
 4488, 6890,
 6897, 6985,
 6997, 7553,
 8292, 8365,
 8368, 8412,
 8432, 8438,
 8477, 8498,
 8546, 8567,
 9366, 9679,
 9686, 10012,
 10020, 10867,
 10874, 11163,
 11175, 11177,
 11179, 11243,
 11255, 11257,
 11259, 11680, 13378
`\hbadness`
 . 6471, 6473, 12469
`\hds@bringhurst` . . 4665
`\hds@crosshead` . . . 4697
`\hds@default` 4554
`\hds@dowding` 4726
`\hds@komalike` 4756
`\hds@memman` 4622
`\hds@ontglike` 4788
`\hds@tandh` 4820
`\hds@wilsondob` . . . 4853
`\headdrop` 1962
`\headheight` . . . 1377,
 1964, 1965,
 1970, 2041,
 2054, 2112,
 2113, 2192,
 2196, 2216,
 2247, 2314, 12382
`\headindent` 4070
`\headnameref` 8145
`\headnamereffalse` 8147
`\headnamereftrue` . 8146
`\headsep` 1377,
 1967, 2042,
 2055, 2114,
 2115, 2193,
 2197, 2215,
 2247, 2314, 12383
`\headstyles` 4554
`\headwidth`
 . . 2890, 2963–2967
`\heavyrulewidth` . . .
 . . 6548, 6589, 6601
`\height` 11790, 11802,
 11809, 11810, 11816
`\hfilneg` 10071
`\hfuzz` . . 6457, 6472,
 6474, 12471, 12472
`\hideindexmarks` . . 8929
`\hline` 6291,
 6293, 6302,
 6307, 6618, 6816
`\hmpunct`
 13103, 13112, 13131
`\hrulefill`
 . . 3990, 4090, 4101
`\hsip` 11250
`\ht` 5047, 6158, 6163,
 6287, 6294,
 6371, 6761,
 9347, 9358,
 9362, 9506,
 9523, 9597,
 9608, 10075,
 10176, 10181,
 10277, 10282,
 10297, 10647,
 10703, 10706,
 10710, 10791,
 11846, 11851,
 11865, 11872,
 11873, 11884,
 11888, 11910,
 11950, 11952,
 13067, 13068, 13071

\HUGE	<u>1128</u> , 3966, 3984, 4055, 4115, 4116	6839, 6847, 6850, 6926, 6929, 6935, 6938, 8232, 11420, 11440, 11446, 11448, 11458, 11466, 11468, 11490	\if@nobreak 206, 4246, 4259, 10931, 11793, 11976, 12335 \if@noskipsec 4252, 11977 \if@numptitle <u>5876</u> , 5916 \if@nyptsizeropt <u>634</u> , 746 \if@openleft ... <u>78</u> , 720 \if@openright <u>75</u> , 723, 3419, 3442, 3510, 3533, 3601, 5015 \if@restonecol <u>74</u> , 3271, 8202, 8210, 8892, 9169 \if@reversemargin 9341, 9409, 9415 \if@shortsubcap <u>7342</u> , 7558 \if@tempswa 3517, 3608, 5021, 6077, 8697, 8760, 11368 \if@tightsubcap 7050, <u>7355</u> , 7482 \if@twocolumn <u>642</u> , 714, 718, 1307, 1729, 1988, 2843, 3217, 3262, 3448, 3539, 3671, 3761, 5135, 5144, 5498, 7076, 7799, 8197, 8205, 8869, 9114, 9146, 9286, 9325, 10475, 12359, 13253 \if@twoside <u>640</u> , 712, 716, 2010, 2762, 2841, 3272, 3397, 3509, 3600, 5014, 7814, 9117, 9293, 9392, 9430, 9566, 11167, 11247, 13249 \ifadd@bstotoc <u>5461</u> , 5526
\Huge	<u>1128</u> , 3458, 3549, 3709, 3732, 3793, 3803, 3855, 3867, 3880, 3951, 3969, 4033, 4144, 4147, 4216, 4566, 4577, 4626, 4631	\if@bsonecol <u>5461</u> , 5499 \if@bsrunin <u>5461</u> , 5537, 5544 \if@contbotsub 7214, <u>7391</u> , 7493, 7513 \if@contcw <u>7088</u> , 7163, 7192 \if@conthang <u>7088</u> , 7173 \if@contindent <u>7088</u> , 7179 \if@epicenter <u>7976</u> , 7999, 8028 \if@epirhs <u>7976</u> , 7994, 8023 \if@fcolmade 2631, 10474, 10483 \if@filesw 8267, 8914, 8982, 9192, 12115, 13144, 13151 \if@firstamp 6062, 6199 \if@firstcolumn 7803, 9115, 9287, 9326, 10476, 13085 \if@hangsubcap <u>7342</u> , 7567 \if@mainmatter . <u>81</u> , 2707, 2721, 2740, 2747, 2768, 2802, 2853, 3657, 5917 \if@mem@nofoot <u>10449</u> , 10461 \if@memoldfont . <u>84</u> , 8046, 8054, 8062, 8070, 8078, 8086, 8094, 8102, 8114 \if@minipage .. 7206, 7229, 7247, 7483, 11361, 13399 \if@mparswitch <u>640</u> , 9338, 9408	
\huge	<u>1128</u> , 3456, 3457, 3547, 3548, 3730, 3731, 3785, 3788, 3876, 3985, 4056, 4074, 4075, 4177, 4204, 4564, 4565, 4575, 4576, 4624, 4625, 4629, 4630, 4758–4760	\if@contbotsub 7214, <u>7391</u> , 7493, 7513 \if@contcw <u>7088</u> , 7163, 7192 \if@conthang <u>7088</u> , 7173 \if@contindent <u>7088</u> , 7179 \if@epicenter <u>7976</u> , 7999, 8028 \if@epirhs <u>7976</u> , 7994, 8023 \if@fcolmade 2631, 10474, 10483 \if@filesw 8267, 8914, 8982, 9192, 12115, 13144, 13151 \if@firstamp 6062, 6199 \if@firstcolumn 7803, 9115, 9287, 9326, 10476, 13085 \if@hangsubcap <u>7342</u> , 7567 \if@mainmatter . <u>81</u> , 2707, 2721, 2740, 2747, 2768, 2802, 2853, 3657, 5917 \if@mem@nofoot <u>10449</u> , 10461 \if@memoldfont . <u>84</u> , 8046, 8054, 8062, 8070, 8078, 8086, 8094, 8102, 8114 \if@minipage .. 7206, 7229, 7247, 7483, 11361, 13399 \if@mparswitch <u>640</u> , 9338, 9408	
\Hy@temp@A	... 8960, 8961, 9010, 9011, 9040, 9041		
\HyInd@ParenLeft 8961, 9011, 9041		
\hyperlink	. <u>8971</u> , 9024		
\hyperpage <u>8971</u>		
\hyperspindexpage	<u>9024</u>		
\hyphenchar 223, 224, 11356		
I			
\ialign 6171		
\idtextinnotes <u>10981</u> , 10985		
\if	.. 149, 388, 395, 406, 408, 415, 452, 2407, 2414, 2694, 6033– 6035, 6037– 6041, 6044– 6046, 6176, 6246, 6367, 6516, 6520, 6524, 6729, 6732, 6741, 6744, 6836,		

- \ifaltindent [5613](#), [5673](#)
- \ifanappendix . [3679](#),
3690, 3747,
3904, 3950,
3968, 4175, [4886](#)
- \ifartopt [664](#),
3396, 3416,
3427, 3616,
3664, 3758,
8610, 9874, 13287
- \ifbounderror [5713](#), [5804](#)
- \ifbvcountinside ..
[11663](#), [11669](#), [11675](#)
- \ifbvcountlines [5573](#),
[11658](#), [11669](#), [11675](#)
- \ifbvperpage
.... [11635](#), [11684](#)
- \ifcat [447](#),
[5793](#), [9734](#), [11350](#)
- \ifchangemarks
.. [11028](#), [11034](#),
[11042](#), [11048](#), [11054](#)
- \ifcntrmod [13159](#), [13179](#)
- \ifdonemaincaption .
.. [7055](#), [7397](#), [7444](#)
- \ifdraftdoc . [644](#), [11035](#)
- \ifeof . [11496](#), [11510](#),
[12164](#), [12176](#), [12206](#)
- \ifetex [267](#), [6706](#)
- \ifextrafont sizes [87](#),
[760](#), [1129](#), [1163](#),
[1177](#), [1192](#), [1208](#)
- \iffalse
6186, [11976](#), [11977](#)
- \IfFileExists . [162](#),
[245](#), [269](#), [292](#),
[313](#), [707](#), [748](#),
[10969](#), [11005](#),
[11017](#), [11525](#),
[11748](#), [12128](#), [13409](#)
- \ifhbox ... [10310](#), [10315](#)
- \ifheadnameref
... [3696](#), [3752](#),
[4269](#), [5936](#), [8145](#)
- \ifinner
.. [4517](#), [4537](#), [11831](#)
- \ifinteger . [5715](#), [5811](#)
- \iflowernumtoname .
..... [12494](#),
[12731](#)–[12735](#),
- 12737–[12763](#),
[12809](#)–[12811](#),
[12813](#), [12816](#)–
[12818](#), [12878](#)–[12885](#)
- \ifluatex [311](#)
- \ifm@mAnd [105](#), [3661](#),
[3678](#), [3702](#),
[5921](#), [5928](#), [5941](#)
- \ifm@mOr [105](#)
- \ifm@mXor [105](#)
- \ifm@mfnmargin
..... [10593](#), [10599](#)
- \ifm@mifetex [267](#), [13338](#)
- \ifm@mifluatex
..... [311](#), [13339](#)
- \ifm@mifpdf . [243](#), [13340](#)
- \ifm@mifxetex
.. [290](#), [301](#), [13341](#)
- \ifm@mno booknewpage
..... [3501](#), [3507](#)
- \ifm@mno partnewpage
..... [3592](#), [3598](#)
- \ifm@mnozpskip
... [3115](#), [5221](#),
[5262](#), [5276](#), [6002](#)
- \ifm@mpn@new@chap .
..... [3635](#), [10945](#)
- \ifm@mpn@new@schap .
..... [3635](#), [10949](#)
- \ifm@mpn@contopt . [10896](#)
- \ifm@mpn@pageopt ...
[10896](#), [10935](#), [10987](#)
- \ifm@msetmp . [9328](#), [9377](#)
- \ifm@msetsp . [9432](#), [9486](#)
- \ifm@mxindy [9029](#)
- \ifmakeordinal
..... [12496](#), [12982](#)
- \ifmem@em@starred@listof
..... [8233](#), [8262](#)
- \ifmem@noetex . [693](#), [706](#)
- \ifmemhyperindex ..
..... [9026](#), [9057](#)
- \ifmemlandscape [614](#), [700](#)
- \ifmempagenotes ...
..... [10911](#),
[10968](#), [11003](#), [11015](#)
- \ifmemtortm
... [9281](#), [9332](#),
[9482](#), [9493](#),
- 9501, 9561,
9562, 9590, [10696](#)
- \ifminusnumber
[12496](#), [12619](#), [12890](#)
- \ifmmode
.. [219](#), [12477](#), [12479](#)
- \ifmsdoc ... [652](#), [13302](#)
- \ifnamesubappendix .
..... [4956](#), [4965](#)
- \ifnobibintoc [8829](#), [8840](#)
- \ifnoglossaryintoc .
..... [9162](#), [9175](#)
- \ifnoindexintoc ...
..... [8885](#), [8896](#)
- \ifnotcntrmod
.... [13159](#), [13186](#)
- \ifnotnumtonameallcaps
..... [12496](#),
[12791](#), [12804](#),
[12846](#), [12860](#),
[12874](#), [12895](#),
[12902](#), [12918](#),
[12940](#), [12956](#), [12978](#)
- \ifnumber@bs
.. [5461](#), [5527](#), [5538](#)
- \ifodd [407](#), [712](#),
[717](#), [2841](#), [5673](#),
[9118](#), [11164](#),
[11244](#), [12320](#),
[12322](#), [12432](#),
[12436](#), [12440](#), [12444](#)
- \ifoddpagel 7824, [7829](#),
[9300](#), [9306](#),
[9339](#), [12307](#), [12404](#)
- \ifonecolglossary .
..... [9140](#), [9151](#)
- \ifonecolindex
..... [8861](#), [8874](#)
- \ifonlyfloats
... [2631](#), [2636](#),
[2638](#), [2640](#), [2642](#)
- \ifpattern [5615](#), [5674](#),
[5761](#), [5767](#), [5773](#)
- \ifpdf [243](#), [2291](#)
- \ifpriornum .. [12496](#),
[12625](#), [12633](#),
[12641](#), [12649](#),
[12657](#), [12665](#),
[12673](#), [12681](#),
[12900](#), [12907](#),

12914,	12923,	\ifscapmargleft . . .	6057,	6206,
12927,	12931,	. . . <u>7703</u> , 7716, 7767	6507,	7210,
12938,	12945,	\ifshowheadfootloc .	7234,	7251,
12952,	12961,	. . . <u>2926</u> , 2936, 2946	7302,	7305,
12965,	12969,	\ifshowindexmark . .	7422,	7549,
12976,	12984,	. . . <u>8929</u> , 8951,	7572,	7575,
12988,	12991,	8955, 9003,	8639,	9655,
12996, 13000, 13003		9033, 9059, 9065	9731,	9756,
\ifraggedbottomsection		\ifshowtextblockloc	9803,	9924,
. <u>4222</u> , 4246	 <u>2926</u> , 2949	9940,	10607,
\ifreportnoidxfile .		\ifshowtrims	10620, 13396, 13403	
. . . <u>8929</u> , 8940, 8989		. . . <u>660</u> , 11271, 11281	\iirdstring	
\ifreversesidepar .		\ifsidebaroneside <u>9565</u>	<u>12506</u> , 12710, 12724	
. . . <u>9425</u> , 9468, 9474		\ifsideparswitch . .	\iindstring	
\ifsamenname <u>93</u> , 1768,	 <u>9425</u> , 9467	<u>12506</u> , 12709, 12723	
1770,	1778,	\ifstarpattern	\immediate	203,
1788,	1790, <u>5617</u> , 5675	210, 8271, 8920,	
1805,	1807,	\ifstre@mnoteof . . .	8983,	9193,
1809,	1818, <u>12162</u> , 12168	9233,	10916,
1834,	1836,	\IfStreamOpen	10928,	11006,
1859,	1861, <u>12086</u> , 12091,	11008,	11018,
1863,	1891,	12097, 12103,	11556,	11560,
2119,	2123,	12109, 12191, 12219	11564,	12116,
2127,	2131,	\ifstrictpagecheck .	12124,	12132,
2492,	2496, <u>12307</u> , 12317	12142,	12150,
2503,	2507,	\ift@bs 11324,	12158, 13146, 13153	
2514,	2518,	11381, 11386,	\immediate@protected@write	
2525,	2529,	11519, 11693, 12184 <u>10924</u> , 10933	
2663,	2667,	\ifTX@ 6388, 6419	\incr@vslines <u>5643</u> , 5666	
2671,	2684,	\ifvbox	\indentcaption <u>7121</u>	
2688,	2700,	10502, 10590, 10755	\indentpattern <u>5800</u>	
2714,	2728,	\ifvoid 9774,	\index 241,	
5431,	5433,	9779, 9815,	8160, <u>8911</u> , 9080	
5435,	5437,	9826, 9878,	\indexcolsep 8880, <u>8901</u>	
5439,	8648,	9881, 9884,	\indexentry 8983	
8653,	8658,	9894, 10452–	\indexintoc <u>8896</u>	
8663,	8668,	10454, 10539,	\indexmark 2777, 2811,	
8673,	8678,	10547, 10562,	2832, 2863,	
8683,	8688,	10575, 10588,	2910, <u>3346</u> , 8884	
8693,	8711,	10591, 10718,	\indexmarkstyle . . .	
8716,	8721,	10726, 10742, <u>9082</u> , 9096	
8726,	8731,	10753, 10756,	\indexname 2777,	
8736,	8741,	10757, 10761–	2791, 2811,	
8746,	8751,	10764, 11266,	2818, 2863,	
8756,	9530,	11268, 11903, 11929	2874, 2910,	
9533,	9536,	\ifxetex <u>290</u> , 2280, 2288	2919, 8876,	
9539,	9542,	\ignorenoidxfile . <u>8929</u>	8881, 8887, <u>13205</u>	
9545,	10656,	\ignorespaces . 5182,	\indexrule . 8879, <u>8901</u>	
10659,	10662,	5629, 5630,	\indexspace <u>8909</u>	
10665, 10668, 10671		5641, 5671,	\InputIfFileExists . <u>161</u>	

<code>\insert</code> . 9094, 9603, 9645, 9723, 9779, 9882, 9915, 10091, 10144, 10195, 10246, 10335, 10390, 10588, 10590, 10591, 10610, 10753, 10755–10757, 10786	<code>\itemindent</code> 684, 685, 868, 880, 892, 904, 916, 928, 1003, 1015, 1027, 1039, 1051, 1063, <u>5151</u> , 5169, 5413, 5419, 5425, 5435, 5445, 5456, 5518, 5687, 5688, 7967	4878, 4884, 8079, 8081, 8083, 8131, 8615, 8621, 9872
<code>\insert@column</code> 6055, 6114, 6116, 6117, 6119–6121	<code>\itemize</code> 5398 <code>itemize</code> (environment) <u>5398</u>	J <code>\jot</code> <u>6012</u> <code>\justlastaggedleft</code> <u>5099</u>
<code>\insertchapterspace</code> <u>3670</u> , <u>3713</u>	<code>\itemsep</code> . . 867, 879, 891, 903, 915, 927, 939, 949, 959, 969, 979, 989, 1002, 1014, 1026, 1038, 1050, 1062, 1074, 1084, 1094, 1104, 1114, 1124, 5282, 5284, 5303, 5313, 5320, 5322, 5686, 8798, 8799	K <code>\KeepFromToc</code> <u>8262</u> <code>\keepthetitle</code> <u>3293</u> , <u>3304</u> <code>\kill@lastcounter</code> <u>109</u> , <u>13262</u> , <u>13266</u> , <u>13269</u> , <u>13272</u> , <u>13276</u> , <u>13279</u> <code>\killm@matf</code> <u>166</u> , <u>173</u> , <u>6707</u> <code>\killtitle</code> <u>3303</u>
<code>\instre@mandclosed</code> <u>12094</u> , <u>12129</u>	<code>\itemsepi</code> <u>5184</u> , 5199, 5211, 5232, 5243, 5256, 5270, 5303	L <code>\l@</code> <u>8330</u> <code>\l@appendix</code> <u>8539</u> <code>\l@book</code> <u>8380</u> <code>\l@chapapp</code> <u>8506</u> , <u>8533</u> , <u>8540</u> <code>\l@chapter</code> <u>8532</u> <code>\l@figure</code> <u>13267</u> <code>\l@index</code> <u>8905</u> <code>\l@ngrelx</code> . . . 364, 375 <code>\l@nohyphenation</code> . . <u>214</u> <code>\l@paragraph</code> <u>8574</u> <code>\l@part</code> <u>8442</u> <code>\l@section</code> <u>8574</u> <code>\l@subfigure</code> <u>8586</u> <code>\l@subparagraph</code> . . <u>8574</u> <code>\l@subsection</code> <u>8574</u> <code>\l@subsubsection</code> . . <u>8574</u> <code>\l@subtable</code> <u>8586</u> <code>\l@table</code> <u>13277</u> <code>\l@X</code> <u>8306</u> <code>\label</code> . . 7271, 7291, 7316, 7396, 7417, 7429, 7443, 7459, 7625, 7739, 7869, <u>8151</u> , 8159
<code>\instre@mandopen</code> <u>12088</u> , 12140, 12167, 12175	<code>\itemsepii</code> <u>5191</u> , 5202, 5218 <code>\itemsepiii</code> . . . <u>5191</u> , 5205, 5219, 5322	<code>\labelenumi</code> <u>5341</u> <code>\labelenumii</code> <u>5341</u>
<code>\integerfalse</code> 5796 <code>\integertrue</code> 5794 <code>\interfootnotelinepenalty</code> . 9725, 9787, 10780 <code>\interlinepenalty</code> <u>1375</u> , 3482, 3496, 3573, 3587, 3708, 3774, 4283, 4906, 4915, 4995, 5007, 5947, 5989, 8248, 8317, 8393, 8455, 8518, 9725, 9787, 10780, 11369, 11372, 11703	<code>\itshape</code> . 230, 3875, 3915, 3966, 4054–4056, 4177, 4216, 4651, 4652, 4657, 4663, 4678, 4679, 4689, 4695, 4718, 4743, 4748, 4754, 4833, 4834, 4845, 4851, 4866, 4867,	
<code>\intertextsep</code> <u>1477</u> <code>\iscntrmod</code> 13159, 13178, 13185 <code>\isopage</code> <u>2331</u> <code>\iststring</code> 12506, 12708, 12722 <code>\it</code> <u>8078</u> <code>\itdefault</code> 13311 <code>\item</code> 5448, 5452, 5460, 5522, 5706, 7958, 8891, 11360, 12396, 12416		

- `\labelenumiii` [5341](#)
- `\labelenumiv` [5341](#)
- `\labelitemi` [5394](#)
- `\labelitemii` [5394](#)
- `\labelitemiii` [5394](#)
- `\labelitemiv` [5397](#)
- `labelled` (environment) [5424](#)
- `\labelsep` [5152](#), [5310](#),
[5317](#), [5327](#),
[5331](#), [5335](#),
[5389](#), [5416](#),
[5419](#), [5422](#),
[5433](#), [5457](#),
[5997](#), [7967](#), [8806](#)
- `\labelwidth` . . . [5152](#),
[5309](#), [5310](#),
[5316](#), [5317](#),
[5326](#), [5327](#),
[5330](#), [5331](#),
[5334](#), [5335](#),
[5413](#), [5419](#),
[5425](#), [5431](#),
[5457](#), [7967](#),
[8804](#), [8805](#), [11991](#)
- `\langle` [218](#)
- `\language` [225](#)
- `\LARGE` [1128](#),
[3137](#), [3853](#),
[4054](#), [4114](#),
[4162](#), [4187](#), [4190](#)
- `\Large` [1128](#),
[3814](#), [3826](#),
[3875](#), [3896](#),
[3897](#), [3915](#),
[3942](#), [4031](#),
[4194–4196](#),
[4330](#), [4331](#),
[4588](#), [4589](#),
[4766](#), [4767](#),
[4790–4792](#),
[4833](#), [4834](#),
[4866](#), [4867](#), [5477](#)
- `\large` . . [1128](#), [3140](#),
[3143](#), [3833](#),
[3834](#), [3909](#),
[3910](#), [3934](#),
[4105](#), [4353](#),
[4354](#), [4595](#),
[4596](#), [4771](#),
[4772](#), [4799](#),
[5868](#), [5960](#),
[5961](#), [8421](#),
[8426](#), [8428](#),
[8486](#), [8491](#),
[8493](#), [13304](#), [13306](#)
- `\lastbox` [10287](#),
[10309](#), [10314](#), [11577](#)
- `\LastFrameCommand` [11922](#), [11965](#)
- `\lasthline` [6293](#)
- `\lastkern` . . [9638](#), [11895](#)
- `\lastlineparrule` . [5059](#)
- `\lastlinerulefill` [5059](#)
- `\lastskip` [4513](#),
[4533](#), [7486](#),
[7509](#), [8851](#), [11792](#)
- `\lccode` [11414–](#)
[11416](#), [11588](#), [11607](#)
- `\lcmminusname` [12501](#)
- `\Lcount` [23](#)
- `\leaders` [5065](#),
[6689](#), [6691](#),
[6695](#), [6697](#),
[6803](#), [6811](#), [8283](#)
- `\leadpagetoclevel` [4979](#), [5003](#)
- `\leavespergathering` [13194](#)
- `\left` . . [6366](#), [6367](#), [6374](#)
- `\leftbar` [12031](#)
- `leftbar` (environment) [12009](#)
- `\leftcenterright` . [5075](#)
- `\leftmargin` . . . [683](#),
[864](#), [876](#), [888](#),
[900](#), [912](#), [924](#),
[936](#), [946](#), [956](#),
[966](#), [976](#), [986](#),
[999](#), [1011](#), [1023](#),
[1035](#), [1047](#),
[1059](#), [1071](#),
[1081](#), [1091](#),
[1101](#), [1111](#),
[1121](#), [5135](#),
[5179](#), [5180](#),
[5300](#), [5308](#),
[5315](#), [5325](#),
[5329](#), [5333](#),
[5413](#), [5425](#),
[5437](#), [5446](#),
[5451](#), [5456](#),
[5483](#), [5485](#),
[5487](#), [5519](#),
[5689](#), [5693–](#)
[5696](#), [5700](#),
[5702](#), [5703](#),
[7968](#), [8805](#),
[8806](#), [12391](#),
[12392](#), [12405](#),
[12406](#), [12410](#), [12411](#)
- `\leftmargini` . . [864](#),
[876](#), [888](#), [900](#),
[912](#), [924](#), [936](#),
[946](#), [956](#), [966](#),
[976](#), [986](#), [999](#),
[1011](#), [1023](#),
[1035](#), [1047](#),
[1059](#), [1071](#),
[1081](#), [1091](#),
[1101](#), [1111](#),
[1121](#), [5135](#),
[5153](#), [5300](#), [5639](#)
- `\leftmarginii`
. [5135](#), [5308](#), [5309](#)
- `\leftmarginiii`
. [5135](#), [5315](#), [5316](#)
- `\leftmarginiv`
. [5135](#), [5325](#), [5326](#)
- `\leftmarginv`
. [5135](#), [5329](#), [5330](#)
- `\leftmarginvi`
. [5135](#), [5333](#), [5334](#)
- `\leftmark` [2794](#), [2834](#),
[2877](#), [2887](#), [2922](#)
- `\Leftrightarrow` . [11055](#)
- `\leftskip`
[468](#), [470](#), [3186](#),
[3188](#), [3190](#),
[5031](#), [5032](#),
[5039](#), [5040](#),
[5069](#), [5078](#),
[5079](#), [5085](#),
[5086](#), [5095](#),
[5100](#), [5102](#),
[5104](#), [5109](#),
[8309](#), [8325](#),
[8386](#), [8400](#),
[8448](#), [8462](#),
[8510](#), [8526](#),

- 8625, 8627, 5698, 7116, 2181, 2186,
 9693, 9694, 7162, 7168, 2202, 2207, 2245
 10030–10032, 7705, 11680, `\lowernumtonamefalse`
 10045, 10443, 11689, 11690, . 12495, 13018,
 10881, 10882, 11725, 11735, 13024, 13036, 13042
 11363, 11541, 11702, 11736, 11987, `\lowernumtonametrue`
`\leftspringright` . 5124 11993, 11994, . 12791, 12804,
`\legend` 7222, 7920 11997, 11998, 12846, 12860,
`\Lenv` 25 12006, 12007, 12874, 12895,
`\letcountercounter` . 12037, 12040, 12902, 12918,
. 12298 12048, 12053, 12364 12940, 12956,
`\lightrulewidth` 5159 12978, 13012, 13030
. 6548, 6595 `\list` 5159
`\line` 2939, `\listfigurename` 2774, `\Lpack` 26
2947, 11066, 2788, 2808, `\LT@cols` 6630
11067, 11074, 2815, 2860, `\LT@hline` 6618
11075, 11081, 2871, 2907, `\Ltrimpicbl` 11070, 11144
11082, 11088, 2916, 13205, 13265 `\Ltrimpicbr` 11070, 11145
11089, 11095, `\listoffigures` . . 13265 `\Ltrimpictl` 11070, 11142
11096, 11115, `\listoffigures*` . 13265 `\Ltrimpictr` 11070, 11143
11121, 11127, `\listoftables` . . . 13275 `\luatexfalse` 311
11133, 11188, `\listoftables*` . . 13275 `\luatextrue` 311
11189, 11196, `\listparindent` `\luatexversion`
11197, 11203, 685, 5168, 320, 322, 324
11204, 11209, 5177, 5217, `\lxvchars` . 69, 1705,
11210, 11216, 5237, 5248, 1753–1756, 1985
11217, 11222, 5444, 5445,
11223, 11229, 5517, 5518,
11230, 11235, 11236 5688, 12389, 12401
`\linemodnum` `\listtablename` 2775,
. . . 5576, 5590, 2789, 2809,
5592, 5593, 2816, 2861,
5599, 5601, 2872, 2908,
5606, 5607, 2917, 13205, 13275
11647, 11649, 11650 `\lofmark` 2774, 2808,
`\linenottooshort` . 5036 2829, 2860, 2907
`\linenumberfont` `\longtable` 6617
. 5570, 13246 `\loop` 6275, 6417, 6888,
`\linenumberfrequency` 6979, 10307,
. 5566, 11299, 12168, 13184
5567, 5576, 13245 `\loosesubcaptions` 7355
`\lineskip` 1289, `\Lopt` 21
3140, 6179, `\lotmark` 2775, 2809,
6778, 9100, 11705 2830, 2861, 2908
`\linespercol` `\lower` 6299, 6374, 12485
. . 6866, 6871, 6910 `\lowercase`
`\linewidth` 4457, 4465, 11417, 11589, 11608
5178, 5179, `\lowermargin`
5181, 5681, . . . 1948, 1960,
5691, 5693, 2110, 2111, 10070, 10071, 10076
2181, 2186, 10069, 10070,
2202, 2207, 2245 10073, 10074, 10076

M

- `\M@currentTitle` 8154,
 8163, 8179, 8181
`\M@getttitle`
 . . . 3469, 3492,
 3560, 3583,
 3696, 3752,
 4269, 4306,
 5850, 5857,
 5936, 5983,
 7150, 7224,
 7241, 8185, 8186
`\m@m@addamp` 6059
`\m@m@Andfalse`
 . . 105, 3655, 5915
`\m@m@Andtrue`
 . . 105, 3658, 5918
`\m@m@BTnormal` 6809, 6817
`\m@m@cline` 6794
`\m@m@empty` 3624
`\m@m@h` 10069,
 10070, 10071, 10076
`\m@m@k` . 10069, 10070,
 10073, 10074, 10076

- \m@m@makecolfloats 10497,
10553, 10561,
10573, 10732, 10741
- \m@m@makecolintro 10513,
10546, 10559,
10571, 10725, 10739
- \m@m@makecoltext 10497,
10555, 10567,
10580, 10735, 10749
- \m@m@Ofalse 105
- \m@m@Ortrue 105
- \m@m@pagenote 10958, 10959
- \m@m@pnwrite 10933, 10953, 10969
- \m@m@singlespace 2988,
2992, 3048,
3054, 3096,
9646, 9796,
9916, 9933,
10092, 10116,
10145, 10159,
10196, 10219,
10247, 10260,
10336, 10355,
10391, 10406,
10601, 10611, 10789
- \m@m@tempdima 7654, 7781,
7783, 7788, 7790
- \m@m@wrpnote 10920, 10942
- \m@m@xfloat 3093, 3095
- \m@m@Xorfalse 105
- \m@m@Xortrue 105
- \m@mabparskip 3116, 6002
- \m@make@footgroup 9868, 10036, 10051
- \m@make@footnotemark 9856, 9961
- \m@make@footnotetext 9853, 9913, 10050
- \m@make@footstart 9867,
10042, 10052,
10063, 10132, 10234
- \m@make@makefnmark 9861, 9998
- \m@make@mpfn 9859, 9992
- \m@make@mpfootnotetext 9873, 9929, 10053
- \m@make@mppara@footgroup 10327, 10371, 10432
- \m@make@mpparafootnotetext 10368, 10402
- \m@make@mpthreecol@footgroup 10039, 10054
- \m@make@mpthreecolfootnotetext 9860, 9995
- \m@make@mptwocol@footgroup 10088, 10131, 10179
- \m@make@mptwocolfootnotetext 10128, 10155
- \m@make@para@footgroup 10324, 10370, 10424
- \m@make@para@footstart 10372, 10440
- \m@make@parafootfmt 10369, 10417
- \m@make@parafootnotetext 10367, 10388
- \m@make@threecol@footgroup 10189, 10232, 10275
- \m@make@threecolfootfmt 10231, 10269
- \m@make@threecolfootnotetext 10229, 10244
- \m@make@twocol@footgroup 10085, 10130, 10174
- \m@make@twocolfootfmt 10129, 10168
- \m@make@twocolfootnotetext 10127, 10142
- \m@make@xfootnote 9852, 9905
- \m@make@xfootnotemark 9855, 9953
- \m@make@xfootnotenext 9858, 9984
- \m@makefootfootmark 9863, 10009
- \m@makefootmarkstyle 9865, 9974
- \m@makefootnote 9851, 9898
- \m@makefootnotemark 9854, 9945
- \m@makefootnotetext 9857, 9978
- \m@makefootref 9862, 10002
- \m@makemakefootmark 9864, 10026
- \m@makemp@footgroup 10039, 10054
- \m@makethempfn 9860, 9995
- \m@malabel 5426, 5440
- \m@matbeginf 163, 168
- \m@matendf 165, 168
- \m@mcalthm 13095, 13111, 13115
- \m@mcalscscapraise 7765, 7778
- \m@mclassicht 2069, 2120
- \m@mdodoreinextrafeet 9778, 10589, 10754
- \m@mdoextrafeet 9769,
10552, 10565,
10579, 10731, 10746
- \m@mdoextrafeetendmini 9822
- \m@mdoextrafeetmini 6001, 9809
- \m@mDOSplits 10070
- \m@mDOWNSf 10692,
10705–10710, 10712
- \m@mfirmfmlists 5226
- \m@mfnmarginfalse 10594
- \m@mfnmargintrue 10595
- \m@mGgf 9227, 9233, 9235
- \m@mGobbleopt 12241, 12243
- \m@mGobbleoptandtwo 12232
- \m@mGobbleoptsandtwo 12231, 12232
- \m@mhe@dreset 2538, 2548,
2558, 2568, 2578
- \m@mhfstyle 2446
- \m@mHline 6807, 6816
- \m@mifetexfalse 268
- \m@mifetextrue 272
- \m@mifluatexfalse 312

- \m@mifluatextrue .. 317
- \m@mifpdffalse 244
- \m@mifpdftrue 249
- \m@mifxetexfalse .. 291
- \m@mifxetextrue ... 298
- \m@minterparanoteglua
..... 10301
- \m@mipn@skip
.... 10292, 10301
- \m@mmlinesht . 2076, 2124
- \m@mnm@argin
... 9258, 9382,
9387, 9438,
9443, 9570,
9575, 10628, 10633
- \m@mmakehboxofhboxes
10306, 10427, 10435
- \m@mnmf@check .. 9637,
9663, 9966, 10819
- \m@mnmf@prepare 9633,
9657, 9667,
9744, 9927,
9943, 9970,
10100, 10124,
10153, 10166,
10203, 10226,
10254, 10267,
10345, 10364,
10400, 10415,
10624, 10813, 10823
- \m@mmpar@margin ...
... 9331, 9379,
9397, 9410,
9412, 9416, 9418
- \m@mnearestht 2084, 2128
- \m@mnewlog 12253
- \m@mnewlogs 12253
- \m@mnoobooknewpagefalse
..... 3501
- \m@mnoobooknewpagetrue
..... 3501
- \m@mnoobooknewpagefalse
..... 3592
- \m@mnoobooknewpagetrue
..... 3592
- \m@mnmzpskipfalse ..
..... 3115, 3118
- \m@mnmzpskiptrue ...
..... 3115, 3122
- \m@mold@addamp ... 6059
- \m@mold@footnotetext
..... 9645, 9839
- \m@mold@mpfootnotetext
..... 9793, 9840
- \m@mopfootnote 10515,
10550, 10577, 10729
- \m@mopfootnotebf ..
10527, 10563, 10744
- \m@mopsidebar 10538,
10554, 10566,
10574, 10733, 10747
- \m@mopsidefoot
10717, 10734, 10748
- \m@mopthfwidth ... 2446
- \m@mpn@new@chapfalse
... 3635, 3756, 10946
- \m@mpn@new@chaptrue
..... 3635, 3641
- \m@mpn@new@schapfalse
... 3635, 3642, 10950
- \m@mpn@new@schaptrue
..... 3635, 3755
- \m@mpn@contoptfalse .
..... 10899
- \m@mpn@pageoptfalse .
..... 10897
- \m@mpn@pageopttrue 10904
- \m@mprovenv 12227, 12228
- \m@mprovlog 12259
- \m@mprovlogs 12259
- \m@mprtime 13109, 13110
- \m@mremovehboxes ..
10314, 10428, 10436
- \m@mrestore@spacing
..... 3058,
3069, 3073, 3077
- \m@mrigidbalance ..
... 10070, 10177,
10182, 10278, 10283
- \m@msavepartopsep 5286
- \m@msavetopsep ... 5286
- \m@mscap@capbox ...
... 7643, 7757,
7769, 7772,
7781, 7789,
7877, 7901, 7917
- \m@mscap@fbox
... 7643, 7747,
7766, 7782, 7788
- \m@mscap@forcap 7729,
7760, 7865,
7880, 7895,
7904, 7912, 7920
- \m@mscap@fortoc ...
... 7729, 7760,
7891, 7893, 7904
- \m@mscap@checkregside
... 7801, 7810, 7813
- \m@mscap@checkside .
..... 7766, 7798
- \m@mscapend@fbox ..
... 7743, 7754,
7874, 7900, 7916
- \m@mscapi 7268, 7269,
7282, 7288,
7289, 7307,
7313, 7314, 7325
- \m@mscapii ... 7279,
7280, 7282,
7300, 7303, 7307
- \m@mscaplabel . 7729,
7756, 7867,
7869, 7876, 7896
- \m@mscaplkern 7708, 7768
- \m@mscapmainwidth .
..... 7705,
7712, 7745, 7748
- \m@mscapmarg .. 7677,
7816, 7819,
7822, 7837, 7838
- \m@mscapopboxes ...
... 7763, 7764,
7883, 7907, 7923
- \m@mscappos
... 7659, 7780, 7787
- \m@mscapraise . 7654,
7769, 7772,
7782-7784,
7789, 7790,
7792, 7793, 7796
- \m@mscapstart@fbox .
... 7741, 7743,
7871, 7897, 7913
- \m@mscapthisside ..
... 7774, 7812, 7844
- \m@mscnc@me . 4245, 4307
- \m@msetm@argin
... 9258, 9381,
9437, 9569, 10627

- \m@msetmpcodes ... 9407
- \m@msetmpfalse
 - 9377, 9378
- \m@msetmptrue 9377, 9380
- \m@msetspfalse ... 9432
- \m@msetsptrue 9432, 9436
- \m@mSFirmlists ... 5226
- \m@mSidebar@margin .
 - .. 9568, 9589, 9606
- \m@mSidefoot@margin
 - 10626, 10695
- \m@mSidepar@left ..
 - 9446, 9496
- \m@mSidepar@right .
 - 9446, 9494
- \m@mSpar@margin 9435,
 - 9456, 9469,
 - 9471, 9475,
 - 9477, 9488, 9492
- \m@mSplitoff 10070
- \m@mSPrtime 13109, 13110
- \m@mStightlists .. 5250
- \m@mTten 12516,
 - 12549, 12550,
 - 12552, 12556,
 - 12557, 12559,
 - 12563, 12564,
 - 12566, 12570,
 - 12571, 12573,
 - 12577, 12578,
 - 12580, 12584,
 - 12585, 12587,
 - 12591, 12592,
 - 12594, 12598,
 - 12599, 12601,
 - 12605, 12606,
 - 12608, 12612, 12613
- \m@mTightlists ... 5250
- \m@mungebox
 - . 10294, 10345,
 - 10364, 10400, 10415
- \m@mUnvxh . 10285, 10295
- \m@mVerbfont 11352
- \m@mWhich@margin ..
 - 9284,
 - 9331, 9492,
 - 9589, 9606, 10695
- \m@mXindyfalse ... 9029
- \m@mXindytrue 9029
- \M@needsP@ 12450, 12451
- \M@sect 4266, 4268
- \m@sideb@left 9579, 9593
- \m@sideb@right
 - 9579, 9591
- \m@sideft@left
 - 10683, 10699
- \m@sideft@right ...
 - 10683, 10697
- \M@sneesp@ 12450, 12451
- \M@TitleReference .
 - 8153,
 - 8166, 8171, 8183
- \mainmatter 3387
- \mainmatter* 3387
- \makeatletter . 15, 8265
- \makeatother 17
- \makebox 3846, 4021,
 - 4036, 4062,
 - 4129, 7996,
 - 8001, 8005,
 - 8025, 8030,
 - 8034, 11670, 11676
- \makechapterstyle .
 - ... 3779, 3782,
 - 3799, 3807,
 - 3819, 3839,
 - 3851, 3863,
 - 3873, 3888,
 - 3901, 3906,
 - 3919, 3929,
 - 3946, 3963,
 - 3982, 4005,
 - 4013, 4028,
 - 4050, 4067,
 - 4092, 4109,
 - 4138, 4155,
 - 4169, 4181,
 - 4192, 4198, 4210
- \makeevenfoot
 - ... 2427, 2594,
 - 2613, 2650,
 - 2846, 2883, 2958
- \makeevenhead . 2421,
 - 2592, 2609,
 - 2654, 2794,
 - 2834, 2877,
 - 2887, 2921, 2956
- \makefootmark 9690, 9864
- \makefootmarkhook .
 - . 9675, 9695, 10033
- \makefootrule
 - .. 2475, 2599, 2968
- \MakeFramed
 - . 11788, 12010,
 - 12015, 12020,
 - 12025, 12038, 12051
- \makeglossary 9185
- \makeheadfootvposition
 - .. 2540, 2600, 2622
- \makeheadposition .
 - 2489,
 - 2597, 2882, 2898
- \makeheadrule . 2475,
 - 2598, 2848,
 - 2885, 2897, 2967
- \makeheadstyles ...
 - ... 4554, 4622,
 - 4665, 4697,
 - 4726, 4756,
 - 4788, 4820, 4853
- \makeindex 8911
- \makelabel 5172, 5392,
 - 5408, 5414,
 - 5420, 5426,
 - 5440, 5458, 7969
- \MakeLowercase
 - ... 3896, 3897,
 - 4098, 4672,
 - 4673, 4684,
 - 4713, 4724,
 - 4738, 4872, 4873
- \makememglossaryhook
 - 9187, 9197
- \makememindexhook 8911
- \makeoddfont .. 2427,
 - 2595, 2615,
 - 2651, 2847,
 - 2884, 2959, 4136
- \makeoddhead
 - ... 2427, 2593,
 - 2611, 2655,
 - 2795, 2821,
 - 2835, 2878,
 - 2888, 2923, 2957
- \makeordinalfalse .
 - 13010, 13016, 13022
- \makeordinaltrue ..
 - 13028, 13034, 13040
- \makepagenote ... 10914

<code>\makepagestyle</code> 2544 , 2608, 2647, 2649, 2653, 2763, 2797, 2824, 2845, 2880, 2892, 2955	<code>\marginparpush</code> 1729 , 1980, 9357	<code>\maxsecnumdepth</code> 8782 , 13243
<code>\makeshook</code> 2536	<code>\marginparsep</code> . 1729 , 1978, 2016, 2024, 2103, 2253, 2318, 2894, 2964, 3856, 7647, 9098, 9109, 9110, 9368, 9370, 9448, 9452, 9552, 10677	<code>\maxtocdepth</code> 8780
<code>\makesmarks</code> . . 2536 , 2601, 2629, 2764, 2784, 2798, 2812, 2825, 2876, 2886, 2899, 2911	<code>\marginparwidth</code> 1979, 2010, 2104, 2253, 2318, 2895, 2965, 3857, 7649, 9097, 9109, 9370, 9447, 9499, 9553, 10677	<code>\mbox</code> . . . 4008, 4011, 7233, 12482, 12490, 13183, 13189
<code>\makerunningfootwidth</code> 2446	<code>\markboth</code> 2664, 2677, 2704, 2737, 2766, 2773– 2777, 2851, 2859–2864, 2900, 2906– 2910, 8161, 8223	<code>\meaning</code> 340, 382, 6269, 6510
<code>\makerunningheadwidth</code> 2446	<code>\markright</code> 2668, 2718, 2779, 2800, 2807–2811, 2858, 2901, 8162	<code>\medievalpage</code> 2323
<code>\makerunningwidth</code> 2446 , 2596, 2617, 2881, 2896, 2966	<code>\mathbf</code> 8073	<code>\medmuskip</code> 12477
<code>\MakeShortVerb</code> . . 11581	<code>\mathcal</code> 8105, 8107, 8111	<code>\medskipamount</code> . . . 1324
<code>\makesidefootmark</code> 10878 , 10886	<code>\mathcode</code> . . 6332, 6353	<code>\medspace</code> 12475
<code>\makesidefootmarkhook</code> 10863 , 10883	<code>\mathit</code> 8081	<code>\mem@currentlabelname</code> ... 7588, 7599, 7605, 7615, 7621
<code>\makethanksmark</code> 3183 , 3216	<code>\mathnormal</code> 8117, 8119, 8123	<code>\mem@doclearpage</code> 10458
<code>\makethanksmarkhook</code> 3168 , 3192	<code>\mathop</code> 12255, 12257, 12261, 12263	<code>\mem@em@starred@listoftrue</code> 8263
<code>\maketitle</code> 3211 , 3238, 3282	<code>\mathrm</code> 8049	<code>\mem@makecol</code> 10545 , 10583, 10724
<code>\maketitlehooka</code> 3146 , 3249	<code>\mathsf</code> 8057	<code>\mem@makecolbf</code> 10558 , 10584, 10724
<code>\maketitlehookb</code> 3146 , 3251, 13294	<code>\mathtt</code> 8065	<code>\mem@makecoldblf</code> 10570
<code>\maketitlehookc</code> 3146 , 3253, 13296	<code>\maxdepth</code> 1451 , 4515, 4535, 10556, 10568, 10581, 10736, 10750, 13064	<code>\mem@noetexfalse</code> . . 693
<code>\maketitlehookd</code> 3146 , 3255	<code>\maxdimen</code> 6410, 6472, 9085, 10341, 10360, 10396, 10411, 11830	<code>\mem@noetextrue</code> . . . 693
<code>\MakeUppercase</code> 18, 2658, 4107, 4118, 4703, 4733, 4790, 4792, 4827, 4828, 4860, 4861, 4974		<code>\mem@oldshipout</code> 11264 , 11269 , 11272 , 11278
<code>\marg</code> 232		<code>\mem@ps@find@real</code> 2402
<code>\marginpar</code> 10600, 11036		<code>\mem@ps@safe@change</code> 2402 , 2422 , 2428 , 2434 , 2440 , 2447 , 2465 , 2470 , 2480 , 2484 , 2490 , 2536 , 2541 , 2756
<code>\marginparmargin</code> . 9379		<code>\mem@set@ps@extra@info</code> .. 2402 , 2545, 2604

- \membicaptioninfo 7261, 7324
- \membionenumcaptioninfo 7261, 7306
- \membitwonumcaptioninfo 7261, 7281
- \membookinfo 3465, 3475, 3478
- \membookstarinfo 3465, 3494
- \memcaptioninfo 7146, 7151, 7267, 7287, 7312
- \memchapinfo 3630, 3686, 3693
- \memchapstarinfo 3630, 3750
- \memcline 6794, 6807
- \memdskip 3084, 3091
- \memdskipstretch 3079, 3085–3088
- \memfblinexa 11925, 11936
- \memfblinexb 11913, 11925
- \memfblistfixparams 11995, 12004
- \memfontenc 742, 756, 758, 761, 763
- \memfontfamily 742, 756, 761
- \memfontpack 742, 758, 763
- \memglodesc 9234, 9251
- \memglofile 9210, 9221, 9226, 9233
- \memglonum 9251
- \memgloref 9235, 9251
- \memgloterm 9234, 9251
- \memgobble 128
- \memhline 6807
- \memhyperindexfalse 9026, 9056
- \memhyperindextrue 9026
- \memjustarg 128
- \memlandscapefalse 614, 617
- \memlandscapetrue 614, 616
- \memleadpageinfo 4987, 5005
- \memleadpagestarinfo 4987, 4993
- \memlegendinfo 7222
- \memletcmdtxt 122
- \memletttxcmd 122
- \memletttxtxt 122
- \memlistsubcaptions 7532
- \memnamedlegendinfo 7237, 7242
- \memoirpostopthook 709
- \memold@docleapage 10458
- \memorigdbs 5025, 5030
- \memorigpar 5025, 5052, 5066
- \mempagenotesfalse 10912
- \mempagenotestruetrue 10917
- \mempartinfo 3556, 3566, 3569
- \mempartstarinfo 3556, 3585
- \memPD 3099, 3106, 3109, 3110
- \mempnofilewarn 10969, 10999, 11010, 11021
- \mempoeminfo 5844, 5851
- \mempoemstarinfo 5844, 5858
- \memPoemTitleinfo 5899, 5931, 5934
- \memPoemTitlestarinfo 5899, 5978
- \memRTLleftskip 468, 3187, 3189, 3191, 5070, 5096, 5101, 5103, 5110, 8311, 8326, 8387, 8401, 8449, 8463, 8511, 8527, 8626, 8628, 9562, 11364, 11542
- \memRTLraggedleft 468
- \memRTLraggedright 468, 3701, 3772, 3834, 3836, 3897, 4075, 4208, 4331, 4354, 4381, 4589, 4596, 4603, 4640, 4646, 4652, 4673, 4679, 4767, 4772, 4777, 4828, 4834, 4840, 4861, 4867, 4873
- \memRTLrightskip 468, 5072, 5094, 5103, 5112, 8313, 8315, 8389, 8391, 8451, 8453, 8513, 8515, 8630, 9560, 9562, 11364, 11540
- \memRTLvleftskip 468, 5630
- \memRTLvrightskip 468, 5656, 5660
- \memsavefootnote 10961
- \memsavepagenote 10961
- \memsecinfo 4242, 4272, 4276
- \memsecstarinfo 4242, 4307
- \memsetcounter 209, 13147, 13154
- \memsub@label 7396, 7429, 7443, 7459, 7581
- \memsubfig@caption 7556, 7559, 7566
- \memsubfig@captionpar 7571, 7574, 7578
- \memtortmfalse 9281, 9288, 9296, 9303, 9307, 9314
- \memtortmtrue 9281, 9285, 9290, 9298, 9301, 9309, 9316

\memUchead	<u>2657</u> , 2664,	4125–4127,	N
	2668, 2677,	4133, 4149, 4179	\n 13417–13419
	2705, 2719,	\MidFrameCommand . .	\n@me@number
	2738, 2745, 11922, <u>11965</u>	. 12887, 13013,
	2766, 2773–	\midpartskip . . 3528,	13019, 13025,
	2777, 2779,	3577, 4573, 4632	13031, 13037, 13043
	2800, 2807–2811	\midPoemTitleskip .	\namedlegend <u>7237</u>
\memwritetoglo 5954, <u>5962</u>	\namedsubappendices
. 9229, <u>9231</u>		\midrule . . . 6585, 6638 4956
\mergepagefloatstyle		\midsloppy <u>12467</u> , 12473	\namenumberand
. 2633		midsloppy (environ-	. 12501, 12908,
\message 11812,		ment) 12473	12915, 12946,
11837, 11841,		\miniscule <u>1128</u>	12953, 12984,
11846, 11852,		\minusname <u>12501</u> , 12890	12988, 12996, 13000
11856, 11865,		\minusnumberfalse 12542	\namenumbercomma . .
11870, 11872,		\minusnumbertrue 12544 <u>12501</u> ,
11878, 11885,		\mit <u>8114</u>	12900, 12908,
11896, 11904,		\mkern 8283	12915, 12938,
11909, 11915,		\morecmidrules	12946, 12953, 12976
11933, 11949,	 6682, <u>6700</u>	\namerefoff <u>8185</u>
11980, 11987, 11998		\movetoevenpage . 12434	\namerefon <u>8185</u>
\MessageBreak		\movetooddpage . . <u>12442</u>	\namesubappendixfalse
. . . . 247, 294,		\mp@footgroupv@r 4956
306, 315, 749,		. . 9817, <u>10066</u> ,	\namesubappendixtrue
766, 1772, 1811,		10089, 10193, 10328 4956
1899, 2133,		\msdocfalse 648, 651, 653	\nametest . 93, 1767,
2134, 2674,		\msdoctrue 655	1769, 1777,
2690, 2731,		\msdoublespacing . .	1787, 1789,
3238, 6446,	 <u>13300</u> , 13303	1804, 1806,
7670, 8044,		\mskip 12477, 12479	1808, 1817,
8105, 8117,		\mssinglespacing . .	1833, 1835,
8703, 9383,	 <u>13300</u> , 13305	1858, 1860,
9439, 9571,		\multfootsep <u>9632</u> , 9641	1862, 1890,
10629, 13367, 13368		\multicolumn . . <u>6197</u> ,	2118, 2122,
\meta . 214, 233, 235, 237		6286, 6293, 6298	2126, 2130,
\meta@font@select . 214		\multiplefootnotemarker	2491, 2495,
\meta@hyphen@restore	 <u>9633</u> , 9638	2502, 2506,
. 222, 227		\multiply 5593,	2513, 2517,
\midbicaption <u>7128</u>		5607, 6914,	2524, 2528,
\midbookskip . . 3437,		7551, 10331,	2662, 2666,
3486, 4562, 4627		10332, 10384,	2670, 2683,
\midchapskip . . 3717,		10385, 11650,	2687, 2699,
3726, 3734,		12550, 12557,	2713, 2727,
3790, 3791,		12564, 12571,	5430, 5432,
3935, 3937,		12578, 12585,	5434, 5436,
3938, 3975,		12592, 12599,	5438, 8647,
4009, 4017,		12606, 12613,	8652, 8657,
4046, 4059,		13098, 13165, 13198	8662, 8667,
4063, 4071,		\multispan . . . 6198,	8672, 8677,
4072, 4086,		6688, 6694, 6811	8682, 8687,

- 8692, 8710, \newenvironment ... 28
- 8715, 8720, \newfixedcaption . 7254
- 8725, 8730, \newflo@tctr 7012
- 8735, 8740, \newfloat 7015,
- 8745, 8750, 13261, 13271, 13407
- 8755, 9530, \newfootnoteseries 9842
- 9533, 9536, \newif 74, 75, 78, 81,
- 9539, 9542, 84, 87, 93, 105–
- 9545, 10656, 107, 243, 250,
- 10659, 10662, 267, 273, 290,
- 10665, 10668, 10671 296, 311, 318,
- \NAT@idxtxt 9080 614, 634, 644,
- \NAT@index 9080 652, 660, 664,
- \NC@ 6250, 6254 693, 2926, 2930,
- \NC@char 6244– 3115, 3501,
- 6247, 6249–6251 3592, 3635,
- \NC@do . 6248, 6259, 6266 3637, 4222,
- \NC@ecs 6145, 6148 4886, 4956,
- \NC@find 6253, 6262, 5461, 5463,
- 6265, 6280, 6416 5465, 5467,
- \NC@ifend . . 6255, 6256 5573, 5613,
- \NC@list 6077, 5615, 5617,
- 6248, 6266, 6271 5713, 5715,
- \NC@rewrite . 6258, 6260 5876, 6059,
- \NC@rewrite@* 6272 6388, 7055,
- \NC@rewrite@X 6415 7088–7090,
- \NC@show . . . 6266, 6267 7342, 7343,
- \NC@strip . . 6268, 6270 7355, 7391,
- \NC@tabular 7703, 7976,
- . . . 6737, 6748, 6750 7977, 8145,
- \Needspace 12450 8262, 8840,
- \needspace 12446 8861, 8896,
- \Needspace* 12450 8929, 8933,
- \negthinspace . . . 12479 9026, 9029,
- \neq 11049 9140, 9175,
- \new@environment 12229 9282, 9377,
- \newarray . . . 5719, 5729 9425, 9428,
- \newblock . . . 688, 8821 9433, 9565,
- \newcol@ 6251, 6252, 6460 10449, 10593,
- \newcolumnntype 6243, 10896, 10898,
- 6272, 6355, 6458 10911, 11028,
- \newcommand . . 21–27, 29 11324, 11635,
- \newcomment 11663, 12162,
- 11399, 11405, 11407 12307, 12308,
- \newcount . 72, 6020– 12494, 12496–
- 6022, 6387, 12499, 13159, 13160
- 6569, 6570, \newinputstream . 12071
- 6573, 6574, \newinsert 9084, 9514,
- 6819, 6821– 9843, 9869,
- 6823, 10069, 11297 10059, 10066, 10637
- \newlabel 7596, 7602, 7612, 7618, 7630
- \newlanguage 215
- \newleadpage 4979
- \newline 5641
- \newlistentry
- . . . 7046, 8296,
- 8574, 8576,
- 8578, 8580,
- 8582, 8586,
- 8588, 13267, 13277
- \newlistof . . . 8219,
- 8278, 13265, 13275
- \newloglike 12253
- \newoutputstream 12056
- \newpmemlabel
- 12339, 12341
- \newread . . 11492, 12073
- \newsavebox . 7643, 7644
- \newskip 3119,
- 4322, 4325,
- 4345, 4348,
- 4369, 4373,
- 4395, 4398,
- 4418, 4422,
- 5286, 5287,
- 6721, 6722,
- 7349–7353, 10301
- \newsubfloat 7045
- \newtoks 5348,
- 6271, 6496,
- 6824, 9082,
- 11285, 11287, 11296
- \newwrite 8270,
- 8919, 9192,
- 10915, 11553, 12058
- \next 6246, 6255, 6257,
- 6581–6583,
- 9732, 9734–
- 9736, 9758,
- 11421–11423,
- 11427, 11429,
- 11432, 11436,
- 11437, 11439,
- 11445, 11450,
- 11454, 11456,
- 11465, 11467,
- 11473, 11476,
- 11480, 11487,

11489, 11509,	\Namexvi	\nointerlineskip . .
11512, 12205, 12208	12752, 12784, 12839	. . . 9365, 9373,
\nexttoken . 6579, 6582	\Namexvii	11274, 11940, 11957
\nfss@text 219	12752, 12785, 12840	\nolimits . 12257, 12263
\Namec 12731,	\Namexviii	\nonzeroparskip . . 3116
12902, 12940, 12978	12752, 12786, 12841	\nopagebreak
\Namei . . . 12737, 12767	\Namexx 4552, 8625, 8626
\Nameii . . 12737, 12768	12752, 12795, 12851	\nopartblankpage . 3592
\Nameiii . . 12737, 12769	\Namexxx	\nopfbreakOutput . .
\Nameiv	12757, 12796, 12852	. . 4491, 4494, 4505
12737, 12770, 12812	\noalign 6196, 6585,	\noprelistbreak . . 4552
\Nameix . . 12742, 12775	6591, 6597,	\normalbaselineskip
\NameI	6603, 6607,	. . . 2478, 3044,
12757, 12798, 12854	6633, 6664,	9101, 9103, 9105
\NameIx	6679, 6700,	\normalbottomsection
12757, 12799, 12855	6711, 6805, 4222
\NameIxx	6810, 6812, 10073	\normalcaption . . . 7121
12757, 12800, 12856	\nobibintoc 8840	\normalcaptionwidth
\NameIxxx	\nobibintocfalse . 8841 7114
12762, 12801, 12857	\nobibintoctrue . . 8842	\normalcolor . . 9817,
\NameM 12731,	\nobookblankpage . 3501	9828, 9886,
12962, 12966, 12970	\nobvbox 11761	10522, 10533, 13053
\NameMm . . . 12731,	\nochangemarks . . 11029	\normalfont . . . 1751,
12924, 12928, 12932	\noDisplayskipStretch	1758, 2921–
\NameMmm . . 12731, 12895 3079	2924, 3173,
\NameO	\noexpand 340,	3177, 3179,
12731, 12766, 12808	347, 364, 375,	3215, 3456–
\NameV . . . 12737, 12771	429, 430, 440,	3458, 3483,
\Namevi	447, 458–461,	3497, 3547–
12742, 12772, 12814	5351, 6171,	3549, 3574,
\Namevii	6246, 6498,	3588, 3700,
12742, 12773, 12815	6528, 7026,	3701, 3730–
\Nameviii 12742, 12774	7422, 8163,	3732, 3771,
\NameX	8303, 9734,	3772, 3785,
12742, 12778, 12833	11420, 11440,	3788, 3793,
\NameXc	11446, 11448,	3803, 3814,
12762, 12802, 12858	11458, 11466,	3826, 3833–
\NameXi	11468, 11480,	3836, 3853,
12747, 12779, 12834	11481, 11490,	3855, 3867,
\NameXii . . 12747, 12780	11624, 11628,	3875, 3876,
\NameXiii	12270, 13082, 13384	3880, 3934,
12747, 12781, 12836	\nofiles 202	3942, 3951,
\NameXiv	\noglossaryintoc . 9175	3966, 3969,
12747, 12782, 12837	\noglossaryintocfalse	3984, 3985,
\NameXix 9176	4018, 4019,
12752, 12787, 12842	\noglossaryintoctrue	4031, 4033,
\NameXl 9177	4054–4056,
12757, 12797, 12853	\noindexintoc . . . 8896	4098, 4105,
\NameXv	\noindexintocfalse 8897	4114–4116,
12747, 12783, 12838	\noindexintoctrue 8898	4162, 4173,

4177,	4187,	8615,	8621,	\notcntrmodtrue .	13172
4190,	4194–	9083,	9481,	\notedivision	
4196,	4204,	9528,	9556,	10995,	11004, 11016
4216,	4316,	9596,	9632,	\noteentry	10954, <u>10976</u>
4339,	4362,	9680,	9684,	\noteidinnotes	
4389,	4411,	9686,	9704,	10978, <u>10983</u>
4564–4566,		10000,	10013,	\noteinnotes	
4575–4577,		10018,	10021,	10978, <u>10990</u>
4624–4626,		10679,	10702,	\notenuminnotes . . .	
4629–4631,		10771,	10777,	<u>10972</u> , 10985
4639,	4640,	10790,	10868,	\notenumintext	
4645,	4646,	10872,	10874,	10944, <u>10972</u>
4651,	4652,	10975,	11353, 11661	\notepageref	<u>10904</u>
4657,	4663,	\normallineskip . . .		\notesname	<u>10995</u>
4672,	4673,	<u>1289</u> , 9100	\notnumtonameallcapsfalse
4678,	4679,	\normalrulethickness		13023, 13041
4684,	4689,	<u>2475</u> ,	\notnumtonameallcapstrue	
4695,	4703,	2848,	2885,	13011,
4708,	4713,	2897,	2967, 9767	13017, 13029, 13035	
4718,	4724,	\normalsize <u>781</u> ,	1288,	\nouppercaseheads .	
4733,	4738,	2538,	3097,	. .	<u>2657</u> , 2867, 2912
4743,	4748,	4018,	4019,	\nthNamei .	12809, 12822
4754,	4766,	4380,	4381,	\nthNameii	12810, 12823
4767,	4771,	4403,	4429,	\nthNameiii	12811, 12824
4772,	4776,	4602,	4603,	\nthNameiv	12812, 12825
4777,	4781,	4609,	4615,	\nthNameix	12817, 12830
4786,	4799,	4651,	4652,	\nthNameI .	12868, 12881
4804,	4809,	4657,	4663,	\nthNameIx	12869, 12882
4813,	4818,	4684,	4689,	\nthNameLxx	12870, 12883
4827,	4828,	4695,	4713,	\nthNameLxxx	
4833,	4834,	4718,	4724,	12871, 12884
4839,	4840,	4743,	4748,	\nthNameo .	12808, 12821
4845,	4851,	4754,	4776,	\nthNamev .	12813, 12826
4860,	4861,	4777,	4781,	\nthNamevi	12814, 12827
4866,	4867,	4786,	4804,	\nthNamevii	12815, 12828
4872,	4873,	4809,	4813,	\nthNameviii	
4878,	4884,	4818,	4839,	12816, 12829
4907,	4916,	4840,	4845,	\nthNamexc	12872, 12885
4996,	5008,	4851,	4872,	\nthNamexii	12818, 12835
5395,	5417,	4873,	4878,	\nthNamexl	12867, 12880
5423,	5475–	4884,	7209,	\nthNamexx	12865, 12878
5478,	5868,	7232,	7250,	\nthNamexxx	12866, 12879
5940,	5960,	9481,	9528,	\nthstring	
5961,	5987,	9556,	9596,	<u>12506</u> , 12705,
7381,	7382,	10702,	10790,	12719,	12808,
8049,	8057,	13304, 13306, 13402		12812–12818,	
8065,	8073,	\normalsubcaption <u>7342</u>		12833,	12834,
8081,	8089,	\notabverbatim@processline		12836–12842, 12991	
8097,	8352,	<u>11344</u> , 11349	\num@bs	<u>5533</u> , 5538
8358,	8359,	\notcntrmodfalse	13169		

<code>\number@bsfalse</code> . . .	<code>\onelineskip</code> <u>69</u> , <u>1292</u> ,	4849, 4857–
. 5466, 5473	3438, 3529,	4859, 4863–
<code>\number@bstrue</code> . . . 5472	3869, 3871,	4865, 4869–
<code>\numberheight</code>	3882, 3883,	4871, 4875,
. . 4059, 4120, 4124	3899, 3914,	4876, 4881,
<code>\numberline</code> <u>8289</u>	3917, 3931,	4882, 5208–
<code>\numberlinehook</code> . . <u>8289</u>	3939, 3944,	5210, 5212–
<code>\NumberPoemTitle</code> . <u>5876</u>	3953, 3955,	5214, 5231,
<code>\numdigits</code>	3961, 3971,	5234, 5240–
<u>12530</u> , 12618, 12889	3973, 3975,	5242, 5245,
<code>\NumToName</code> <u>13021</u>	3976, 4000,	5268, 5626,
<code>\numtoName</code> . . . 3866,	4002, 4143,	5963, 5967,
3904, 3950,	4146, 4152,	7139, 7141,
3968, 4176, <u>13015</u>	4156, 4158,	9183, 9554, 10678
<code>\numtoname</code> <u>13009</u>	4159, 4170–	<code>\openany</code> <u>671</u> , 726
	4172, 4182–	<code>\openin</code> . . . 11495, 12132
	4184, 4199–	<code>\openinputfile</code> . . <u>12127</u>
	4201, 4211–	<code>\openleft</code> <u>671</u> , 721
	4213, 4441,	<code>\openout</code> 8271, 8920,
<code>\obeylines</code> 11375	4443, 4448,	9193, 10916,
<code>\oddpagetrue</code>	4450, 4460,	11008, 11556, 12116
. . . . <u>12307</u> , 12316	4468, 4627,	<code>\openoutputfile</code> . <u>12112</u>
<code>\oddpagetrue</code>	4632, 4636–	<code>\openright</code> . . . <u>671</u> , 724
<u>12307</u> , 12320, 12322	4638, 4642–	<code>\operator@font</code> 12255,
<code>\oddsidemargin</code> <u>2010</u> ,	4644, 4648–	12257, 12261, 12263
2048, 2218–	4650, 4654,	<code>\oplus</code> 11043
2222, 2227,	4655, 4660,	<code>\ordinal</code> <u>12701</u>
2310, 12370, 12378	4661, 4669–	<code>\OrdinalToName</code> . . <u>13039</u>
<code>\omit</code> 6796, 6799	4671, 4675–	<code>\ordinaltoName</code> . . <u>13033</u>
<code>oncolabstract</code> (envi-	4677, 4681,	<code>\ordinaltoname</code> . . <u>13027</u>
ronment) . . <u>5547</u>	4682, 4686,	<code>\ordscript</code>
<code>\oncolglossary</code> . . <u>9142</u>	4687, 4692,	<u>12514</u> , 12713, 12727
<code>\oncolglossaryfalse</code>	4693, 4700–	<code>\ordstring</code> . . 12705,
. <u>9140</u> , 9143	4702, 4705–	12708–12710,
<code>\oncolglossarytrue</code>	4707, 4710–	12713, 12719,
. <u>9140</u> , 9142	4712, 4715–	12722–12724, 12727
<code>\oncolindex</code> <u>8864</u>	4717, 4721,	<code>\output</code> 4491, <u>4508</u> , 11685
<code>\oncolindexfalse</code> .	4722, 4730–	<code>\outputpenalty</code>
. 8862, 8865	4732, 4735–	. 4493, 4498, 11727
<code>\oncolindextrue</code> . 8864	4737, 4740–	<code>\outstre@mandclosed</code>
<code>\oncoltocetc</code> <u>8195</u>	4742, 4745, <u>12106</u> , 12113
<code>\oncolumn</code> 3263, 3449,	4746, 4751,	<code>\outstre@mandopen</code> .
3540, 8198,	4752, 4796–	<u>12100</u> , 12122, 12156
8210, 8875,	4798, 4801–	<code>\oval</code> 11198,
8892, 9152,	4803, 4806–	11211, 11224, 11237
9169, 13258, 13316	4808, 4824–	<code>\overfullrule</code>
<code>OnehalfSpace</code> (environ-	4826, 4830–	. 645, 646, 649, 657
ment) <u>3071</u>	4832, 4836–	<code>\overridescapmargin</code>
<code>\OnehalfSpacing</code> . . .	4838, 4842, <u>7844</u>
. <u>2997</u> , 3073	4843, 4848,	

P		
\p@enumii	5345	\pagenote 560–566, 1917,
\p@enumiii	5345	. 10918, 10958, 1931, 1932,
\p@enumiv	5345, 8809	10962, 10963, 10966 1936, 1940,
\p@footnote	9848	\pagenotesubhead 1941, 1945,
\p@subX	7047	10947, 10951, 11025 1986, 2011,
\PackageError	416, 458	\pagenumbering 2971, 2019, 2034,
\page	11794	3385, 3407, 13237 2097, 2145,
\pageai	552	\pagenumbering* 2971 2148, 2154,
\pageaai	552	\pageold 524 2163, 2220,
\pageaaii	552	\pagepostvo 533 2239, 2306,
\pageaaiii	552	\pagepottvo 533 2325, 2333,
\pageaiv	552	\pageref 8138 2342, 2351,
\pageao	552	\pagerefname 2356, 2360,
\pageav	552	8138, 10989, 13223 2364, 2383,
\pageavi	552	\pageroyalvo 540 2389, 4121,
\pagebi	560	\pageshrink 4125, 11103,
\pagebii	560	. 11844, 11848, 11165, 11170,
\pagebiii	560	11859, 11883, 11893 11175, 11177,
\pagebiv	560	\pagesmalldemyvo 533 11179, 11245,
\pagebo	560	\pagesmallroyalvo 540 11250, 11255,
\pagebroadsheet	524	\pagestatement 524 11257, 11259, 12419
\pagebv	560	\pagestretch 11843, 11859, 11893 \paragraph 4383
\pagebvi	560	\pagestyle 19, 13182, \paragraphfootnotes 10319
\pagecrownvo	533	13188, 13236, 13320 \paragraphfootstyle 10366
\pagedbill	524	\pagesuperroyalvo 540 \paragraphmark 3336
\pagedemyvo	540	\pagetofootnote 10961 \paraheadstyle
\pagedepth	4518, 4538	\pagetotal 3224, 4389, 4392
\pageexecutive	524	3243, 4474, \parahook 4384, 4390
\pagefillstretch	11795	4477, 11811, \paraindent 4386, 4392
\pagefilstretch	11795	11812, 11835, \parbox 2550, 2552,
\pagefoolscapvo	533	11841, 12453, 12459 2554, 2560,
\pagegoal	4474, 4477, 11813, 11830, 11835, 11842, 11877, 11879, 12453, 12459	\pagrefename 13218 2562, 2564,
\pageimperialvo	540	\paperheight 524–531, 533–545, 547– 2570, 2572,
\pageinnotes 10978, 10986	550, 552–558, 2574, 2580,
\pagelargecrownvo	533	560–566, 1916, 2582, 2584,
\pagelargepostvo	533	1951, 1957, 4101, 4104,
\pageledger	524	2003, 2039, 4457, 4465,
\pagelegal	524	2106, 2169, 7165, 7194, 7579
\pageletter	524	2172, 2178, \parg 232
\pagemcrownvo	547	2187, 2239, \parindent 1307, 3185,
\pagedemyvo	547	2306, 2335, 3700, 3701,
\pagemediumvo	540	2350, 2354, 3771, 3772,
\pagemlargecrownvo	547	2373, 2378, 4417, 4612,
\pagesmallroyalvo	547	2382, 2387, 4659, 4691,
\pagename	13218	11103, 11173, 11253 4720, 4750,
		\paperwidth 524–531, 4783, 4815,
		533–545, 547– 4847, 4880,
		550, 552–558, 5046, 5048,

- 5049, 5077, 5270, 5273, 5301 5293, 5298,
 5084, 5090, \parsepii 5184, 5201, 5321, 5323,
 5097, 5105, 5214, 5216, 5516, 13295, 13297
 5108, 5121, 5218, 5234, \partopsepii 5191
 5177, 5261, 5236, 5245, \partopsepiiii
 5275, 5510, 5247, 5258, . . . 5194, 5220,
 5940, 5987, 5260, 5272, 5264, 5278, 5323
 8246, 8316, 5274, 5312, 5313 \partpageend 3597
 8392, 8454, \parsepiiii 5204 \partrefname 8141, 13224
 8516, 8889, \parshape 5181, \parttitlefont
 9098, 9166, 11989, 11994, 12007 . . . 3547, 3554,
 9491, 9692, \parskip 1292, 4577, 4581,
 10029, 10104, 3038, 3060, 4631, 4760, 4792
 10207, 10348, 3063, 3117, \PassOptionsToPackage
 10419, 10445, 3121, 5176, 141
 10880, 11365, 5198, 5201, \patchcmd 380
 11543, 12389, 12401 5204, 5282, \patchcmd@a . . . 382, 384
 \parnopar 9422 5284, 5511, \patchcmd@b . . . 391, 394
 \parsep 686, 866, 867, 5669, 6002, \patchcmd@c 402, 405, 425
 878, 879, 890, 8238, 8247, \patchcmd@d
 891, 902, 903, 8890, 9167, . 408, 409, 411, 414
 914, 915, 926, 9422, 9612, \patchcmd@e 388, 389, 426
 927, 938, 939, 10795, 11361, \patchcmdError 385, 444
 948, 949, 958, 11365, 12390, 12402 \patchcommand . 381, 397
 959, 968, 969, \part 3525 \patternfalse . 5616,
 978, 979, 988, \partblankpage . . . 3592 5709, 5833, 5836
 989, 1001, 1002, \partmark 3336, \patterntrue 5831
 1013, 1014, 3571, 4903, patverse (environ-
 1025, 1026, 4912, 4992, 5004 ment) 5830
 1037, 1038, \partname patverse* (environ-
 1049, 1050, . 3551, 4578, 13205 ment) 5835
 1061, 1062, \partnamefont \pdffalse 243
 1073, 1074, . . . 3547, 3551, \pdfhorigin 2282
 1083, 1084, 4575, 4578, \pdfoutput . . . 252,
 1093, 1094, 4629, 4758, 4790 254, 256, 258, 2292
 1103, 1104, \partnamenum \pdfpageheight . . . 2278
 1113, 1114, . . 3551, 3576, 4579 \pdfpagewidth 2279
 1123, 1124, \partnumberline . . . \pdftrue 243
 5176, 5301, 3565, 8474 \pdfvorigin 2281
 5312, 5319, \partnumberlinehook \pen@ltyabovetfpbreak
 5447, 5521, 8474 4480
 5690, 12390, 12402 \partnumfont
 \parsepi 5184, . . . 3547, 3553, \pen@ltyabovetfpbreak
 5198, 5210, 4576, 4580, 4480,
 5211, 5215, 4630, 4759, 4791 4493, 4524, 4544
 5231–5233, \partopsep . . . 5196, \pen@ltybelowtfpbreak
 5235, 5242– 5208, 5230, 4480
 5244, 5246, 5240, 5254, \pen@ltybelowtfpbreak
 5255, 5256, 5264, 5268, 4481,
 5259, 5269, 5278, 5290, 4498, 4526, 4546

- \pfbre@kdispl@y 4487, 4495, 4502
- \pfbreak 4510
- \pfbreakdisplay 4485, 4488
- \pfbreakOutput 4492, 4508
- \pfbreakskip .. 4483, 4487, 4497, 4501, 4521, 4522, 4541, 4542
- \phantom 4042, 5636, 6331
- \phantomsection 11, 200, 3470, 3493, 3561, 3584, 4921, 4963, 5002, 5528, 5848, 5903, 8234, 8830, 8886, 9163
- \plainbreak 4438
- \plainfancybreak . 4472
- \plainfootnotes .. 9838
- \plainfootstyle 9895, 10049
- \PlainPoemTitle .. 5876
- \pmemlabel 12318, 12337
- \pmemlabelref 12319, 12341
- \pmemprotected@write 12327, 12338
- \pmname ... 13103, 13116
- \pnchap ... 10939, 10947
- \pnschap .. 10939, 10951
- \poemf@rhdr 5902, 5972, 5974, 5976, 5978
- \poemf@rtoc 5902
- \poemlines 5566
- \poemt@c 5894, 5904, 5908, 5910
- \PoemTitle 5889
- \poemtitle 5840
- \PoemTitlefont 5956, 5960
- \poemtitlefont 5864, 5868
- \PoemTitleheadstart .. 5939, 5951, 5986
- \poemtitlemark 5852, 5868, 5926
- \PoemTitlenumfont 5953, 5960
- \poemtitlestyle 5885, 5927
- \poemtitlestarmark 5885, 5976
- \poemtitlestarpstyle 5885, 5977
- \poemtoc 5842, 5849, 5929, 5933
- \postauthor . 3130, 3142
- \postautotab 6831, 7009
- \postbibhook 8839, 8845, 8851
- \postcaption 7128
- \postchapterprecis 8608, 8609
- \postdate .. 3130, 3144
- \postdisplaypenalty 1372
- \postnoteinnotes 10979, 10992
- \posttitle . 3130, 3138
- \preauthor . 3130, 3139
- \preautotab . 6831, 7005
- \prebibhook 8833, 8834, 8845
- \precaption 7128
- \prechapterprecis 8608, 8609
- \prechapterprecisshift 8609
- \precisfont 8609
- \precistocfont 8621, 8631
- \precistocetext ... 8622
- \predate ... 3130, 3143
- \predisplaypenalty 1372, 11362
- \Pref 8139
- \pref 8136
- \preglossaryhook 9154, 9159, 9173
- \preindexhook 8877, 8882, 8904
- \prenoteinnotes 10977, 10992
- \prepnext@tok . 6049, 6080, 6110, 6122, 6142, 6147
- \pretitle .. 3130, 3137
- \prevdepth ... 3106, 4515, 4516, 4535, 4536, 5053, 7155, 9611, 9617, 9618, 9620, 10794, 10805, 10806, 10808, 13051
- \printbookname 3460, 3485, 4567
- \printbooknum 3460, 3485, 4569
- \printbooktitle 3460, 3488, 3498, 4570
- \PrintChanges 36
- \printchaptername 3703, 3721, 3786, 3801, 3812, 3824, 3843, 3854, 3865, 3877, 3891, 3912, 3932, 3948, 3965, 3993, 4042, 4058, 4077, 4095, 4117, 4141, 4160, 4179, 4185, 4202, 4214
- \printchapternonum 3706, 3721, 3773, 3792, 3885, 3911, 3925, 3940, 3957, 3977, 4024, 4041, 4102, 4145, 4167, 4178, 4220, 8258
- \printchapternum 3703, 3721, 3789, 3804, 3816, 3828, 3845, 3866, 3879, 3893, 3903, 3923,

3936,	3949,	12669,	12677,	\providelength ..	<u>12245</u>
3967,	3995,	12685,	12896,	\provideloglike .	<u>12259</u>
4021,	4035,	12903,	12911,	\ProvidesFile	
4042,	4061,	12919,	12941,	. 42, 44, 46, 48,	
4080,	4099,	12949,	12957, 12979	50, 52, 54, 56,	
4128,	4142,	\protected@edef ...		58, 60, 62, 64, 11500	
4164,	4175, 126, 4275,		\ps@afterbook	<u>3522</u>
4188, 4205, 4217		5728, 5756,		\ps@afterpart	<u>3613</u>
\printchaptertitle .		5786, 5792,		\ps@book	<u>3522</u>
... 3709, <u>3721</u> ,		7588, 8153,		\ps@chapter	<u>2837</u>
3775, 3794,		9651, 9798,		\ps@cleared	<u>2837</u>
3858, 3868,		9920, 9936,		\ps@companion	<u>2890</u>
3881, 3895,		10095, 10119,		\ps@empty	<u>2647</u>
3916, 3943,		10148, 10162,		\ps@epigraph	<u>7992</u>
3952, 3970,		10199, 10222,		\ps@headings	<u>2762</u>
4001, 4010,		10250, 10263,		\ps@indextitlepagestyle	
4064, 4085,		10339, 10358,		<u>8894</u>
4106, 4150, 8259		10394, 10409,		\ps@myheadings ...	<u>2824</u>
\printglossary ...	<u>9256</u>	10604, 10616, 10796		\ps@part	<u>2837</u>
\PrintIndex	35	\protected@write ..		\ps@plain	<u>2649</u>
\printindex	8927 205, 7595,		\ps@Ruled	<u>2880</u>
\printnotes	<u>10998</u>	7601, 7611,		\ps@ruled	<u>2845</u>
\printpageinnotes	<u>10986</u>	7617, 7629,		\ps@showlocs	<u>2955</u>
\printpagenotes .	10998	8957, 8962,		\ps@simple	<u>2653</u>
\printpagenotes*	<u>10998</u>	8965, 9005,		\pstyle	24
\printpartname		9012, 9017,		\put 2939, 2947, 2950,	
.. <u>3551</u> , 3576, 4578		9035, 9042,		7996, 8001,	
\printpartnum		9047, 9060,		8005, 8025,	
.. <u>3551</u> , 3576, 4580		9066, 9220, 10936		8030, 8034,	
\printparttitle <u>3551</u> ,		\protected@xdef ...		11066, 11067,	
3579, 3589,	 115, 3315,		11074, 11075,	
4581, 4908,		3325, 3333,		11081, 11082,	
4917, 4997, 5009		5552, 9712,		11088, 11089,	
\printPoemTitlenum		10829, 10842, 10854		11095, 11096,	
.. 5945, <u>5951</u> , 5988		\providecommand 11,		11103, 11115,	
\printPoemTitlenum .		134, 139, 211,		11121, 11127,	
..... 5942, <u>5951</u>		636, 637, 709,		11133, 11188–	
\printPoemTitletitle		742–744, 5372,		11190, 11196–	
.. 5948, <u>5951</u> , 5990		6546, 7259,		11198, 11203,	
\printtime		8133, 8281,		11204, 11209–	
11191, 13103, <u>13108</u>		8282, 8284,		11211, 11216,	
\printtime*	<u>13108</u>	8912, 9185,		11217, 11222–	
\printZnonum	<u>8258</u>	10995, 11965,		11224, 11229,	
\printZtitle	<u>8259</u>	11970–11973,		11230, 11235–11237	
\priornumfalse		12261, 12263, 12265		\put@bsintoc <u>5525</u> , 5543	
.... 12620, 12891		\providecounter .	<u>12237</u>		
\priornumtrue		\provideenvironment			
. 12622, 12629,		<u>12227</u>	Q	
12637, 12645,		\providexcaption		\Q	12350
12653, 12661,		<u>7254</u>	qframe (environment)	
				<u>12034</u>

- \qitem 7957
- \qitemlabel . 7957, 7969
- \quad 3816, 3828, 4165,
4205, 4217,
4434, 4436,
4485, 4969, 4976
- \quarkmarks 11183
- quotation (environ-
ment) 5443
- quote (environment) 5450
- R**
- \raggedbottom . . 32,
9127, 13251, 13319
- \raggedbottomsection
. 4222
- \raggedbottomsectionfalse
. 4222
- \raggedbottomsectiontrue
. 4222
- \raggedleft 473, 2555,
2565, 2575,
2585, 3854,
3860, 3869,
3953, 3971,
4003, 4018,
4019, 4031,
4033, 4217,
4220, 6840,
7717, 9482, 9561
- \raggedright . . . 33,
472, 2551, 2561,
2571, 2581,
3700, 3771,
3833, 3835,
3896, 4074,
4207, 4330,
4353, 4380,
4588, 4595,
4602, 4639,
4645, 4651,
4672, 4678,
4766, 4771,
4776, 4827,
4833, 4839,
4860, 4866,
4872, 6837,
7719, 7958,
9102, 9482,
9561, 10106, 10209
- \raggedrightthenleft
. 5107
- \raggedwrap 11537
- \raggedyright 5089
- \ragrparindent
. 5089, 11543
- \raise 6290, 12484
- \raisebox 3998,
4062, 7769, 7772
- \rangle 228
- \read 11509,
12169, 12179, 12205
- \readaline 12174
- \readboxedverbatim .
. 12214
- \readstream 12166
- \readverbatim . . . 12183
- \RecordChanges 14
- \ref 7640, 7641, 8136,
8137, 8139–
8143, 8171,
8172, 9707, 10005
- \registrationColour
. 11183
- \rem@special
11602, 11617, 11622
- \renewcommand 18
- \renewfixedcaption 7254
- \renewleadpage . . . 4979
- \reparticle 3831
- \repeat 6279,
6421, 6894,
6990, 10312,
11302, 12171, 13190
- \reportnoidxfile . 8929
- \reportnoidxfilefalse
. 8931
- \reportnoidxfiletrue
. 8930
- \RequireAtEndClass . 192
- \RequireAtEndPackage
. 184
- \RequirePackage . . .
. 245, 269,
292, 313, 707, 13410
- \RequireXeTeX 290
- \reserved@a . . . 6302,
6307, 10928,
10929, 12332, 12333
- \resetbvlinenumber .
. 11655
- \resizebox . . . 4022,
4037, 4063, 4131
- \restoreapp . 4922, 4941
- \restorefromonecol .
. 8195, 8243
- \restoregenumber . 2980
- \restorepagenumber 2981
- \restoretrivseps . 5286
- \reversesideparfalse
. . . 9400, 9402,
9426, 9459, 9461
- \reversesidepartrue
. . . 9398, 9404,
9427, 9457, 9463
- \right . 6366, 6367, 6374
- \rightblock 4071
- \rightmargin . . 5167,
5178, 5439,
5446, 5451,
5520, 5655,
5656, 5692,
5699, 5702,
7968, 11982,
11985, 12393,
12394, 12407,
12408, 12412, 12413
- \rightmark . . . 2795,
2821, 2835,
2878, 2888, 2923
- \rm 8046
- \rmdefault 13308
- \Roman 3922, 5363
- \romannumeral
. . . 5170, 5377,
5388, 5402, 5404
- \rule . . . 3937, 3938,
3991, 4008,
4011, 4133,
7942, 9655,
9731, 9756,
9803, 9924,
9940, 10620, 11739
- \russianpar 5043
- S**
- \S 13227
- \samenamefalse 95
- \samename true 100

\save@decl ... <u>6052</u> ,	4747, 4780,	4801, 4830, 4863
6085, 6142, 6144	4812, 4844, 4877	\setbeforesubsubsecskip
\savepagenumber .. <u>2980</u>	\setafterseccskip <u>4365</u> ,
\savetrivseps <u>5286</u> <u>4319</u> ,	4600, 4648,
\saythanks <u>5557</u>	4587, 4638,	4681, 4710,
\sbox 7159,	4671, 4702,	4740, 4774,
7174, 7568, 13373	4732, 4765,	4806, 4836, 4869
\sc <u>8094</u>	4798, 4826, 4859	\setbiblabel <u>8823</u>
\sc@ref 7639, <u>7640</u>	\setaftersubparaskip	\setbinding <u>1924</u>
\scapmargleftfalse <u>4414</u> ,	\setbvlینums .. <u>11645</u>
... <u>7703</u> , 7806,	4614, 4662,	\setcolsepandrulē <u>1973</u>
7820, 7825,	4694, 4723,	\setcounter 20
7828, 7839, 7851	4753, 4785,	\setDisplayskipStretch
\scapmarglefttrue .	4817, 4850, 4883 <u>3079</u>
... <u>7703</u> , 7804,	\setaftersubseccskip	\setfillsize .. <u>1803</u> ,
7817, 7823, <u>4342</u> ,	1932, 1941,
7830, 7836, 7847	4594, 4644,	1951, 1957, 1964
\scdefault 13313	4677, 4707,	\setfloatlocations <u>7043</u>
\scriptsize <u>1128</u>	4737, 4770,	\setfootnoterule . <u>9763</u>
\scshape 2877, 2887,	4803, 4832, 4865	\sethangfrom
3853, 3896,	\setaftersubsubseccskip	.. <u>4431</u> , 4678, 4679
3897, 4018, <u>4365</u> ,	\setheaderspaces . <u>1962</u>
4031, 4098,	4601, 4650,	\setheadfoot <u>1969</u>
4639, 4640,	4683, 4712,	\setlrmargins <u>1928</u> , 2050
4672, 4673,	4742, 4775,	\setlrmarginsandblock
4684, 4713,	4808, 4838, 4871 <u>1939</u> ,
4724, 4738,	\setarrayelement . <u>5751</u>	2328, 2337, 2345
4872, 4873,	\setbeforeparaskip .	\setlrvchars <u>1751</u>
8095, 8097, 8099 <u>4392</u> ,	\setm@mscaplkern ..
\secdef <u>104</u>	4607, 4654, <u>7708</u> , 7746
\secheadstyle 4316, <u>4319</u>	4686, 4715,	\setmarginnotes .. <u>1977</u>
\sechook ... 4311, <u>4317</u>	4745, 4779,	\setmpbools . 9389, <u>9391</u>
\secindent . 4313, <u>4319</u>	4811, 4842, 4875	\setnzzplist ... <u>5195</u> ,
\section	\setbeforeseccskip .	5222, 5263, 5277
. <u>4310</u> , 8834, 11026 <u>4319</u> ,	\setpagebl <u>2348</u>
\sectionbib 8857	4586, 4636,	\setpagebm <u>2367</u>
\sectionmark	4669, 4700,	\setpagebr <u>2367</u>
... 2778, 2827,	4730, 4764,	\setpagecc <u>2367</u>
2858, 2901, 3336	4796, 4824, 4857	\setpageml <u>2348</u>
\sectionname 4966	\setbeforesubparaskip	\setpagemr <u>2367</u>
\sectionrefname <u>4414</u> ,	\setpagetl <u>2348</u>
..... 8143, <u>13224</u>	4613, 4660,	\setpagetm <u>2348</u>
\see <u>9073</u>	4692, 4721,	\setpagetr <u>2367</u>
\seealso <u>9073</u>	4751, 4784,	\setparaheadstyle .
\seename <u>9073</u>	4816, 4848, 4881 <u>4392</u> ,
\semiisopage <u>2340</u>	\setbeforesubseccskip	4609, 4657,
\setafterparaskip <u>4342</u> ,	4689, 4718,
..... <u>4392</u> ,	4593, 4642,	4748, 4781,
4608, 4656,	4675, 4705,	4813, 4845, 4878
4688, 4717,	4735, 4769,	\setparahook <u>4390</u> , 4605

<code>\setparaindent</code>	<code>\setsubparahook</code>	<code>\setulmarginsandblock</code>
. <u>4392</u> , 4606 <u>4412</u> , 4611 <u>1956</u> ,
<code>\setpnumwidth</code> <u>8279</u> , 8633	<code>\setsubparaindent</code>	2329, 2338, 2346
<code>\setrectanglesize</code> <u>4414</u> ,	<code>\setup@bstrack</code>
. <u>1766</u> , 1915, 1920	4612, 4659, <u>5496</u> , 5536
<code>\setrmarg</code> <u>8279</u> , 8633	4691, 4720,	<code>\setupbox@verb</code> <u>11701</u> ,
<code>\setseheadstyle</code>	4750, 4783,	11713, 11749, 12223
. 3833,	4815, 4847, 4880	<code>\setupboxverb@line</code>
3834, <u>4319</u> ,	<code>\setsubseheadstyle</code> <u>11682</u> ,
4588, 4589, 3835,	11711, 11747, 12220
4639, 4640,	3836, <u>4342</u> ,	<code>\setupcomment</code>
4672, 4673,	4595, 4596, <u>11393</u> , 11400
4703, 4733,	4645, 4646,	<code>\setverbatimbreak</code> <u>11528</u>
4766, 4767,	4678, 4679,	<code>\setverbatimfont</code> <u>11352</u>
4799, 4827,	4708, 4738,	<code>\setverselinenums</code> <u>5587</u>
4828, 4860, 4861	4771, 4772,	<code>\setxlvchars</code> <u>1751</u>
<code>\setsechook</code> <u>4317</u> , 4584	4804, 4833,	<code>\sf</code> <u>8054</u>
<code>\setsecindent</code> <u>4319</u> , 4585	4834, 4866, 4867	<code>\sf@memsub@label</code>
<code>\setsecnumdepth</code> <u>8786</u>	<code>\setsubsechook</code> 7589, <u>7591</u>
<code>\setsecnumformat</code> <u>4432</u> <u>4340</u> , 4591	<code>\sf@memsub@label</code>
<code>\setsidebarheight</code>	<code>\setsubsecindent</code> 7583, 7584, <u>7587</u>
. <u>9520</u> , 9546 <u>4342</u> , 4592	<code>\sf@size</code> 12482, 12490
<code>\setsidebars</code> <u>9529</u>	<code>\setsubsubseheadstyle</code>	<code>\sfdefault</code> 13309
<code>\setsidecappos</code> <u>7659</u> <u>4365</u> , 4602,	<code>\sffamily</code> 3867,
<code>\setsidecaps</code> <u>7650</u>	4603, 4651,	3875, 3880,
<code>\setsidefeet</code> <u>10655</u>	4652, 4684,	3951, 3969,
<code>\setsidefootheight</code>	4713, 4743,	3984, 3985,
. <u>10641</u> , 10672	4776, 4777,	4019, 4022,
<code>\setSingleSpace</code> <u>2987</u>	4809, 4839,	4033, 4038,
<code>\setSpacing</code>	4840, 4872, 4873	4187, 4190,
. <u>2983</u> , 2992,	<code>\setsubsubsechook</code>	4624–4626,
2998, 3000, <u>4363</u> , 4598	4629–4631,
3002, 3004,	<code>\setsubsubsecindent</code>	4758–4760,
3006, 3008, <u>4365</u> , 4599	4766, 4767,
3010, 3012,	<code>\setthesection</code>	4771, 4772,
3016, 3018, <u>4948</u> , 4953	4776, 4777,
3020, 3022,	<code>\settocdepth</code> <u>8708</u>	4781, 4786,
3024, 3026,	<code>\settodepth</code> <u>7788</u> , <u>7789</u>	8055, 8057, 8059
3028, 3030,	<code>\settoheight</code>	<code>\shaded</code> <u>11782</u> , 12030
3048, 3054, 3069 3935, 7781, 7782	<code>shaded (environment)</code>
<code>\setspbools</code> <u>9455</u> , 9487	<code>\settrimmedsize</code> <u>12009</u>
<code>\setspcode</code> <u>9466</u> , 9489 <u>1908</u> , 1982,	<code>\shipout</code> <u>11264</u> , <u>11281</u>
<code>\setstocksize</code> <u>1908</u>	2349, 2353,	<code>\shortsubcaption</code> <u>7342</u>
<code>\setsubparaheadstyle</code>	2358, 2362,	<code>\showbox</code> 11824, 11934,
. <u>4414</u> ,	2368, 2372,	11951, 13063, 13066
4615, 4663,	2377, 2381, 2386	<code>\showcols</code> <u>6266</u>
4695, 4724,	<code>\settrims</code> <u>1908</u> , 1983	<code>\showheadfootlocfalse</code>
4754, 4786,	<code>\settypeblocksize</code> <u>1919</u> <u>2926</u>
4818, 4851, 4884	<code>\setulmargins</code>	<code>\showheadfootlocoff</code>
 <u>1948</u> , 2056, 2141 <u>2926</u>

- `\showheadfootlocon` [2926](#)
- `\showheadfootloctrue`
 - [2926](#)
- `\showindexmarkfalse`
 - [8935](#)
- `\showindexmarks` .. [8929](#)
- `\showindexmarktrue` [8934](#)
- `\showlists` [11940](#)
- `\showoutput` .. [11824](#), [11934](#), [11940](#), [11951](#)
- `\showtextblockloclfalse`
 - [2926](#)
- `\showtextblocklocoff`
 - [2926](#)
- `\showtextblocklocon`
 - [2926](#)
- `\showtextblockloctrue`
 - [2926](#)
- `\showtrimsfalse` ...
 - [661](#), [11059](#)
- `\showtrimsoff` ... [11059](#)
- `\showtrimson` [11059](#)
- `\showtrimstrue`
 - [662](#), [11060](#)
- `\sidebar` ... [4062](#), [9602](#)
- `\sidebarfont` .. [9528](#), [9543](#), [9596](#), [9607](#)
- `\sidebarform` [9559](#), [9607](#)
- `\sidebarheight`
 - [9548](#), [9549](#)
- `\sidebarhsep`
 - ... [2257](#), [9525](#), [9531](#), [9581](#), [9585](#)
- `\sidebarmargin` ... [9568](#)
- `\sidebaronesidefalse`
 - [9566](#)
- `\sidebaronesidettrue`
 - [9566](#)
- `\sidebartopsep` [2259](#), [9518](#), [9540](#), [9598](#)
- `\sidebarvsep` .. [2259](#), [9525](#), [9537](#), [9624](#)
- `\sidebarwidth`
 - ... [2257](#), [9525](#), [9534](#), [9580](#), [9604](#)
- `\sidecapfloatwidth` .
 - [7705](#), [7745](#)
- `\sidecapmargin` ... [7677](#)
- `\sidecapraise` [7656](#), [7796](#)
- `\sidecapsep`
 - ... [2255](#), [7646](#), [7651](#), [7711](#), [7771](#)
- `\sidecapstyle`
 - ... [7714](#), [7759](#), [7879](#), [7903](#), [7919](#)
- `\sidecaption` [7722](#)
- `\sidecapwidth` . [2255](#), [7646](#), [7652](#), [7710](#), [7758](#), [7878](#), [7902](#), [7918](#)
- `\sidecontcaption` . [7860](#)
- `\sidecontents`
 - [9588](#), [10541](#)
- `\sidefootadjust` ...
 - [10641](#), [10666](#), [10704](#)
- `\sidefootcontents` .
 - [10694](#), [10720](#)
- `\sidefootfootmark` .
 - [10863](#), [10884](#)
- `\sidefootform`
 - [10681](#), [10790](#)
- `\sidefootheight` ...
 - [2265](#), [10641](#), [10674](#)
- `\sidefoothsep`
 - ... [2263](#), [10651](#), [10657](#), [10685](#), [10689](#)
- `\sidefootins`
 - . [10637](#), [10645](#)–[10648](#), [10673](#), [10705](#)–[10707](#), [10713](#), [10718](#), [10757](#), [10764](#), [10786](#)
- `\sidefootmargin` . [10626](#)
- `\sidefootmarksep` ..
 - [10881](#), [10888](#)
- `\sidefootmarkstyle` .
 - [10863](#), [10894](#)
- `\sidefootmarkwidth` .
 - . [10866](#), [10867](#), [10869](#), [10871](#), [10874](#), [10882](#), [10888](#)
- `\sidefootnote` ... [10827](#)
- `\sidefootnotemark` [10838](#)
- `\sidefootnotetext` [10851](#)
- `\sidefootparindent` .
 - [10880](#), [10888](#)
- `\sidefootscript` . [10863](#)
- `\sidefoottextfont` .
 - . [10669](#), [10702](#), [10771](#), [10790](#), [10799](#)
- `\sidefootvsep` . [2265](#), [10651](#), [10663](#), [10708](#), [10709](#), [10812](#)
- `\sidefootwidth`
 - .. [2263](#), [10651](#), [10660](#), [10684](#), [10787](#)
- `\sideins` [2261](#), [9514](#), [9521](#)–[9523](#), [9547](#), [9599](#), [9603](#), [10454](#), [10539](#), [10591](#), [10756](#), [10763](#)
- `\sidelegend` [7909](#)
- `\sidenamedlegend` . [7885](#)
- `\sidepar` [9483](#)
- `\sideparfont` [9481](#), [9500](#)
- `\sideparform` [9481](#), [9500](#)
- `\sideparmargin` [9432](#), [9435](#), [9486](#), [9488](#)
- `\sideparswitchfalse`
 - ... [9398](#), [9400](#), [9429](#), [9457](#), [9459](#)
- `\sideparswitchtrue` .
 - ... [9402](#), [9404](#), [9430](#), [9461](#), [9463](#)
- `\sideparvshift`
 - .. [9483](#), [9510](#)–[9512](#)
- `SingleSpace` (environ-
ment) [3046](#)
- `SingleSpace*` (environ-
ment) [3052](#)
- `\SingleSpacing` ... [2991](#)
- `\skip@` .. [4513](#), [4520](#)–[4522](#), [4525](#), [4533](#), [4540](#)–[4542](#), [4545](#), [11792](#), [11796](#), [11798](#), [11801](#)–[11805](#)
- `\sl` [8086](#)
- `\slashfrac` [12483](#)
- `\slashfracstyle` ...
 - [12481](#), [12484](#), [12485](#)
- `\sldefault` [13312](#)
- `\sloppybottom` [9135](#)
- `\small` [23](#), [858](#), [3261](#), [5475](#), [5476](#), [5571](#),

- 7933, 11043,
11049, 11055, 13246
`\smallskipamount` . 1324
`\smash` 3991
`\snugshade` 12032
`snugshade` (environ-
ment) 12009
`\sourceatright` . . . 5130
`\sourceflush` . . 7934,
7940, 7949, 7950
`\spacefactor` . . 9639,
9642, 9662,
9668, 9965,
9971, 10818, 10824
`Spacing` (environment)
. 3066
`\special` . . . 2285, 11240
`\specialindex` 8911
`\specialrule` 6585, 6641
`\spinemargin`
. . . 1928, 1943,
1946, 2048–
2050, 2099,
2100, 2157,
2160, 2221,
2243, 2325–
2328, 2333,
2334, 2337,
2342–2345,
4123, 4127, 12422
`\splitbotmark` . . . 11869
`\splitfirstmark` . 11868
`\splitmaxdepth` 9609,
9727, 9790,
10783, 10792, 13064
`\splittopskip`
. . . 9608, 9726,
9789, 10071,
10176, 10177,
10181, 10182,
10277, 10278,
10282, 10283,
10782, 10791,
11863, 11864, 13064
`\Sref` 8139
`\ssc@ref` . . . 7639, 7640
`\stanzaskip` . 5625, 5690
`\starpatternfalse` .
. 5618,
5709, 5831, 5838
`\starpatterntrue` . 5836
`\start@vsline`
. . . 5670, 5671, 5672
`\stepcounter`
. . . 5645, 9711,
9901, 9949,
10828, 10841,
11658, 12318,
13137, 13148, 13155
`\stockai` 508, 569
`\stockaii` 508, 570
`\stockaiii` 508, 571
`\stockaiv` 508, 572
`\stockao` 508, 568
`\stockav` 508, 573
`\stockavi` 508, 574
`\stockbi` 516, 576
`\stockbii` 516, 577
`\stockbiii` 516, 578
`\stockbiv` 516, 579
`\stockbo` 516, 575
`\stockbroadsheet` . .
. 480, 594
`\stockbv` 516, 580
`\stockbvi` 516, 581
`\stockcrownvo` . 489, 598
`\stockdbill` . . . 480, 587
`\stockdemyvo` . . 496, 603
`\stockexecutive` 480, 589
`\stockfoolscapvo` . .
. 489, 597
`\stockheight` . . 475,
480–487, 489–
501, 503–506,
508–514, 516–
522, 611, 701,
702, 1909, 1982,
2105, 2167,
2174, 2235,
2278, 2285,
2350, 2354,
2373, 2378,
2382, 2387, 11240
`\stockimperialvo` . .
. 496, 608
`\stocklargecrownvo` .
. 489, 600
`\stocklargepostvo` .
. 489, 601
`\stockledger` . . 480, 593
`\stocklegal` . . . 480, 592
`\stockletter` . . 480, 590
`\stockmcrownvo` . 503, 582
`\stockmdemyvo` . 503, 584
`\stockmediumvo` 496, 604
`\stockmlargecrownvo`
. 503, 583
`\stocksmallroyalvo`
. 503, 585
`\stockold` 480, 591
`\stockpostvo` . . 489, 599
`\stockpottvo` . . 489, 596
`\stockroyalvo` . 496, 606
`\stocksmalldemyvo` .
. 489, 602
`\stocksmallroyalvo` .
. 496, 605
`\stockstatement` 480, 588
`\stocksuperroyalvo` .
. 496, 607
`\stockwidth`
. . . 475, 480–487,
489–501, 503–
506, 508–514,
516–522, 612,
702, 703, 1910,
1982, 2096,
2143, 2150,
2218, 2235,
2279, 2285,
2351, 2356,
2360, 2364,
2383, 2389,
11165, 11170,
11240, 11245, 11250
`\stre@mbvin`
. 12215, 12216, 12218
`\stre@mnoteoffalse` .
. 12164
`\stre@mnoteoftrue` 12164
`\stre@mverb@input` .
. 12187, 12188, 12190
`\stretch` . . 12425, 12427
`\strictpagecheck` 12307
`\strictpagecheckfalse`
. 12307
`\strictpagechecktrue`
. 12307
`\string` 211, 239, 241,
334, 340, 347,

- 351, 402, 406,
 459, 461, 2418,
 3237, 3238,
 5567, 5568,
 6031, 6244,
 6261, 6263,
 7418, 7596,
 7602, 7612,
 7618, 7630,
 7669, 7694,
 7853, 8958,
 8963, 8966,
 8983, 9006,
 9013, 9018,
 9036, 9043,
 9048, 9061,
 9067, 9089,
 9221, 9233–
 9235, 9383,
 9439, 9571,
 10629, 10947,
 10951, 10954,
 11491, 11582,
 11586, 11591,
 11599, 11603,
 11604, 11610,
 11614, 11615,
 11841–11845,
 11949, 11998,
 12157, 12247,
 12339, 13147, 13154
 \stringtoarray 5726, 5801
 \strip@pt 2939, 2947,
 2950, 2951,
 10333, 10386, 11103
 \strut .. 2551, 2553,
 2555, 2561,
 2563, 2565,
 2571, 2573,
 2575, 2581,
 2583, 2585,
 5044, 5056,
 6891, 6897,
 6982, 6986,
 6994, 6997,
 7781, 7789,
 9484, 9742,
 10111, 10171,
 10172, 10214,
 10272, 10273,
 10350, 10421, 11691
 \strutbox 3039,
 5047, 6163,
 6167, 6761,
 6765, 9505,
 9523, 9597,
 9608, 9609,
 9611, 9615,
 9617, 9622,
 9656, 9727,
 9790, 9804,
 9925, 9941,
 10176, 10181,
 10277, 10282,
 10621, 10647,
 10703, 10710,
 10783, 10791,
 10792, 10794,
 10802, 10805, 10810
 subappendices (envi-
 ronment) .. 4961
 \subbottom 7437
 \subcaption 7394
 \subcaptionfont .. 7377
 \subcaptionlabelfont
 7377
 \subcaptionref ... 7638
 \subcaptionsize .. 7377
 \subcaptionstyle . 7339
 \subconcluded 7384
 \subfig@bottom
 .. 7495, 7498, 7527
 \subfig@top ... 7494,
 7497, 7507, 7510
 subfloat (environ-
 ment) 7435
 \subfloatbottomskip
 ... 7349, 7360,
 7370, 7495, 7497
 \subfloatcapmargin .
 7349,
 7362, 7372, 7550
 \subfloatcapskip ..
 ... 7349, 7358,
 7368, 7406,
 7412, 7516, 7522
 \subfloatcaptopadj .
 7349,
 7359, 7369, 7523
 \subfloatlabelskip .
 7050,
 7349, 7361, 7371
 \subfloattopskip ..
 ... 7349, 7357,
 7367, 7494, 7498
 \subitem 8906
 \subparagraph 4405
 \subparagraphmark 3336
 \subparaheadstyle .
 4411, 4414
 \subparahook 4406, 4412
 \subparaindent
 4408, 4414
 \subsecheadstyle ..
 4339, 4342
 \subsechook . 4334, 4340
 \subsecindent 4336, 4342
 \subsection 4333
 \subsectionmark .. 3336
 \subsubitem 8906
 \subsubsecheadstyle
 4362, 4365
 \subsubsechook
 4357, 4363
 \subsubsecindent ..
 4359, 4365
 \subsubsection ... 4356
 \subsubsectionmark 3336
 \subtop 7467
 \symboldef 5460
 \symbollabel 5454, 5458
 symbols (environment)
 5454
 \symbolthanksmark 3151
- T**
- \t@bsfalse 11335
 \t@bstrue 11329
 \tab@position 11297,
 11301, 11302,
 11317, 11319, 11341
 \tab@size ... 11319,
 11330, 11336, 11341
 \tabbingsep 5997
 \tabcolsep 5994, 6220,
 6753, 6865, 6953
 table (environment) 13271
 table* (environment)
 13271

- `\tablename` [13218](#), [13271](#)
- `\tableofcontents` . [8278](#)
- `\tableofcontents*` [8278](#)
- `\tablerefname`
- [8137](#), [13218](#)
- `\tabskip` [6149](#), [6173](#),
- [6758](#), [6771](#),
- [6774](#), [6900](#),
- [6901](#), [6903](#),
- [6905](#), [7001](#),
- [7003](#), [7005](#), [7006](#)
- `\tabsoff` . . [11324](#), [11567](#)
- `\tabson` [11324](#)
- `\tabular` . . . [6214](#), [6475](#)
- `\tabular*` [6214](#)
- `\tabularnewline` . . .
- [6182](#), [6781](#)
- `\tabularx` . . [6389](#), [6466](#)
- `\tabularxcolumn` . . .
- [6405](#), [6459](#)
- `\tabverbatim@processline`
- [11341](#),
- [11382](#), [11387](#),
- [11520](#), [11694](#), [12185](#)
- `\tamark` . [3162](#), [3173](#),
- [3177](#), [3179](#), [3215](#)
- `\teennumbername` . . .
- [12777](#), [12791](#)
- `\teenordinalname` . .
- [12832](#), [12846](#)
- `\teenstring`
- [12506](#), [12749](#)–[12755](#)
- `\temptokstre@m` [12169](#),
- [12170](#), [12179](#), [12180](#)
- `\tensnumbername` . . .
- [12789](#),
- [12910](#), [12948](#), [12997](#)
- `\tensordinalname` . .
- [12844](#), [12985](#)
- `\tensunitsep`
- [12506](#), [12805](#), [12861](#)
- `\TeX` [29](#)
- `\textasteriskcentered`
- [5396](#)
- `\textbf` [3882](#), [8071](#), [8075](#)
- `\textbullet` [5394](#)
- `\textendash` [5395](#)
- `\textfloatsep`
- . . . [1477](#), [1563](#),
- [1567](#), [1571](#),
- [1575](#), [1579](#), [1583](#)
- `\textflush` . . . [7934](#),
- [7939](#), [7944](#), [7946](#)
- `\textfont` [6373](#)
- `\textfraction` [2397](#)
- `\textheight`
- . . . [1921](#), [1951](#),
- [1952](#), [1958](#),
- [2003](#), [2043](#),
- [2070](#), [2073](#),
- [2074](#), [2077](#),
- [2081](#), [2082](#),
- [2085](#), [2090](#),
- [2091](#), [2107](#),
- [2180](#), [2185](#),
- [2241](#), [2308](#),
- [2951](#), [9557](#),
- [10479](#), [10649](#),
- [10679](#), [12357](#),
- [12358](#), [12367](#),
- [12375](#), [13067](#), [13068](#)
- `\textit` . . . [29](#), [3879](#),
- [3886](#), [8079](#), [8083](#)
- `\textperiodcentered`
- [5397](#)
- `\textrm` [8047](#), [8051](#)
- `\textsc` . . [27](#), [8095](#), [8099](#)
- `\textsf` [21](#), [26](#), [8055](#), [8059](#)
- `\textsl` [23](#), [24](#), [8087](#), [8091](#)
- `\textsubscript` . . [12487](#)
- `\textsuperscript` . .
- [3167](#),
- [9632](#), [9698](#),
- [10055](#), [10894](#), [10973](#)
- `\texttt` . [22](#), [25](#), [213](#),
- [238](#), [239](#), [8063](#), [8067](#)
- `\textwidth` . . . [1922](#),
- [1932](#), [1933](#),
- [1942](#), [1988](#),
- [2012](#), [2020](#),
- [2036](#), [2098](#),
- [2156](#), [2161](#),
- [2226](#), [2241](#),
- [2308](#), [2596](#),
- [2598](#), [2599](#),
- [2848](#), [2881](#),
- [2885](#), [2893](#),
- [2939](#), [2947](#),
- [2951](#), [2963](#),
- [2968](#), [4008](#),
- [4011](#), [4071](#),
- [4122](#), [4126](#),
- [5085](#), [5659](#),
- [5660](#), [7800](#),
- [7930](#), [11817](#),
- [12358](#), [12368](#),
- [12376](#), [12420](#), [13054](#)
- `\thanks` [3239](#),
- [3295](#), [3312](#),
- [3320](#), [3330](#), [5551](#)
- `\thanksfootextra` . [3289](#)
- `\thanksfootmark` . . .
- [3170](#), [3193](#)
- `\thanksgap` [3161](#), [3291](#),
- [3314](#), [3324](#), [3332](#)
- `\thanksheadextra` . .
- . . [3157](#), [3199](#), [3288](#)
- `\thanksmark`
- . . . [3160](#), [3290](#),
- [3313](#), [3323](#), [3331](#)
- `\thanksmarksep` [3165](#),
- [3186](#), [3187](#), [3201](#)
- `\thanksmarkseries` .
- . . [3151](#), [3198](#), [3287](#)
- `\thanksmarkstyle` . [3166](#)
- `\thanksmarkwidth` . .
- [3164](#),
- [3171](#), [3172](#),
- [3174](#), [3176](#),
- [3179](#), [3188](#),
- [3189](#), [3200](#), [3201](#)
- `\thanksrule` . [3195](#), [3203](#)
- `\thanksscript` . [3166](#),
- [3173](#), [3177](#), [3179](#)
- `\the@epigraph` [8014](#), [8021](#)
- `\the@toks` [6056](#),
- [6058](#), [6074](#), [6146](#)
- `\the@vsstring` [5728](#), [5735](#)
- `\theauthor` . [3306](#), [3309](#)
- `\theb@vlinenumber` .
- [11655](#), [11671](#), [11677](#)
- `thebibliography` (envi-
 ronment) . . [8837](#)
- `\thebook` [3358](#), [3462](#),
- [3474](#), [3475](#), [4569](#)
- `\thebsm@mctr`
- [12623](#), [12894](#)
- `\thechapter`
- . . . [2769](#), [2803](#),

- 2854, [3358](#),
 3681, 3682,
 3685, 3686,
 3725, 3789,
 3804, 3816,
 3828, 3846,
 3879, 3904,
 3922, 3924,
 3937, 3950,
 3968, 3998,
 4022, 4038,
 4063, 4083,
 4096, 4131,
 4144, 4165,
 4175, 4188,
 4205, 4217,
 4892, 4940,
 4948, 6011,
 10947, 13263, 13273
 \thechrsinstr 5786
 \thecksm@mctr
 12650, 12654, 12939
 \thecmsm@mctr
 12626, 12630, 12901
 \thecp@cntr 12318, 12319
 \thecsm@mctr
 12674, 12678, 12977
 \thedata . . . 3307, [3309](#)
 \theenumi [5337](#),
 5341, 5345, 5346
 \theenumii
 . . [5337](#), 5342, 5346
 \theenumiii
 . . [5337](#), 5343, 5347
 \theenumiv
 . . [5337](#), 5344, 8810
 \theequation [6009](#), 6013
 \thefigure 13263
 \thefootnote
 . . 3152, 5558, 9849
 theglossary (environ-
 ment) [9145](#)
 \theHbook [3431](#)
 \theHpoem [5881](#)
 \theHpoemline [5677](#)
 theindex (environ-
 ment) [8867](#)
 \theism@mctr
 . 12689, 12985,
 12989, 12991,
 12997, 13001, 13003
 \theksm@mctr . 12666,
 12670, 12948, 12955
 \thempfn [330](#),
 9712, 9718, 9860
 \themsm@mctr . 12642,
 12646, 12910, 12917
 \thepage 2650,
 2651, 2654,
 2655, 2794,
 2795, 2821,
 2834, 2835,
 2846, 2847,
 2883, 2884,
 2921, 2924,
 2956–2959,
 2974, 4136,
 7598, 7604,
 7614, 7620,
 7632, 8010,
 8958, 8963,
 8966, 9007,
 9014, 9037,
 9044, 9061,
 9190, 9350,
 10954, 11191, 13083
 \thepagenote . [10900](#),
 10908, 10944, 10954
 \theparagraph [3358](#)
 \thepart [3358](#), 3553,
 3565, 3566, 4580
 \thepmemc@page . . .
[12325](#), 12329, 12339
 \thepoem [5881](#),
 5930, 5931, 5953
 \thepoemline 5679
 \thesection
 . . . 2781, 2903,
[3358](#), 4953, 4976
 \thesheetsequence 13136
 \thesidefootnote . .
 10768,
 10774, 10842, 10847
 \thesidemfn
 . [10773](#), 10829,
 10834, 10854, 10859
 \thesubparagraph . [3358](#)
 \thesubsection . . . [3358](#)
 \thesubsubsection [3358](#)
 \thesubX [7047](#)
 \thetable 13273
 \theTeXbook 29
 \thetitle . . 3305, [3309](#)
 \theTitleReference .
 . . [8149](#), 8179, 8183
 \theverse 5679
 \thevslineno 5807, 5814
 \theX [8297](#)
 \theksm@mctr
 12658, 12662, 12948
 \thexsm@mctr
 12634, 12638, 12910
 \thexsm@mctr . 12682,
 12686, 12985, 12997
 \thicklines . . 11187,
 11195, 11202,
 11208, 11215,
 11221, 11228, 11234
 \thinlines . . 11065,
 11073, 11080,
 11087, 11094,
 11102, 11114,
 11120, 11126, 11132
 \thinmuskip [12479](#)
 \thispagestyle
 713, 717, 2842,
 3230, 3267,
 3270, 3447,
 3512, 3538,
 3603, 3618,
 4998, 5010,
 5017, 8012,
 8252, 8889, 9166
 \thr@@ . . 5399, 6028,
 6044, 6094,
 6131, 6132,
 7838, 10278, 10283
 \threecolumnfootnotes
 [10184](#)
 \threecolumnfootstyle
 [10228](#)
 \tiethstring
[12506](#), 12878–12885
 \tightlist [5281](#)
 \tightlists [5250](#)
 \tightlists* [5250](#)
 \tightsubcaptions [7355](#)
 \time 13096,
 13098, 13123, 13126

- \tiny [1128](#)
- \title [3284](#), [3309](#)
- \titleref [8169](#)
- titlingpage (environment) [3259](#)
- \markbl [11106](#),
11139, 11144,
11149, 11154,
11179, 11226, 11259
- \markbm [11111](#), 11159,
11179, 11219, 11259
- \markbr [11106](#),
11140, 11145,
11150, 11155,
11179, 11213, 11259
- \markml [11111](#), 11157,
11177, 11232, 11257
- \markmr [11111](#), 11158,
11177, 11206, 11257
- \marktl [11106](#),
11137, 11142,
11147, 11152,
11175, 11185, 11255
- \marktm [11111](#), 11156,
11175, 11193, 11255
- \marktr [11106](#),
11138, 11143,
11148, 11153,
11175, 11200, 11255
- \tocbaseline [8191](#)
- \tocentryskip [8191](#)
- \toclevel@all [8775](#)
- \toclevel@appendix [8539](#)
- \toclevel@book ... [3431](#)
- \toclevel@chapter [8535](#)
- \toclevel@none ... [8775](#)
- \toclevel@part ... [8472](#)
- \toclevel@subX ... [7047](#)
- \toclevel@X [8361](#)
- \tocmark 2773, 2807,
2828, 2859, 2906
- \tocnameref [8145](#)
- \tocskip [8191](#)
- \today 11191, [13230](#)
- \toks ... 6050, 6052,
6053, 6098,
6149, 11868–
11870, 13100,
13101, 13112, 13131
- \toks@ 362,
364, 373, 375,
388, 389, 392,
397, 400, 421,
446, 450, 453,
459, 461, 6394,
6397, 6401,
6402, 6424,
6469, 6475,
6509, 11300,
11310, 11316, 11342
- \tolerance
10105, 10208, 12468
- \topfraction [2393](#)
- \topmargin ... 2039,
2052, 2213–
2217, 2312, 12381
- \toprule ... [6585](#), 6637
- \topsep ... 865, 877,
889, 901, 913,
925, 937, 947,
957, 967, 977,
987, 1000, 1012,
1024, 1036,
1048, 1060,
1072, 1082,
1092, 1102,
1112, 1122,
5289, 5292,
5297, 5302,
5311, 5318,
5320, 5515,
6752, 6790,
11804, 11807,
11941, 12388,
12400, 13295, 13297
- \topsepi [5184](#),
5197, 5212,
5233, 5244,
5257, 5271, 5302
- \topsepii [5184](#),
5200, 5215,
5235, 5246,
5259, 5273, 5311
- \topsepiii [5184](#), 5203,
5216, 5219,
5236, 5247,
5260, 5274, 5318
- \topskip [1377](#),
1451, 2008,
2074, 2082,
2086, 2091,
3914, 3960,
3980, 7156,
9136–9138,
9522, 9597,
10646, 10703,
11731, 11875,
11880, 13061, 13064
- \tracingtabularx . [6491](#)
- \traditionalparskip
..... [3116](#)
- \tref [8136](#)
- \trimedge [475](#),
1913, 2094,
2144, 2149,
2219, 2223,
2237, 2351,
2356, 2360,
2364, 2365,
2370, 2375,
2379, 2383,
2384, 2389,
2390, 11165,
11168, 11170,
11245, 11248, 11250
- \trimFrame [11136](#)
- \trimLmarks [11136](#)
- \trimmark
..... [11062](#), 11106–
11109, 11137–11140
- \trimmarks
..... [11161](#), 11239, 11273
- \trimNone [11136](#)
- \trimtop . [475](#), 1912,
2095, 2168,
2173, 2213,
2237, 2350,
2354, 2355,
2359, 2363,
2369, 2373,
2374, 2378,
2382, 2387,
2388, 11162, 11242
- \trimXmarks [11136](#)
- \tt [8062](#)
- \ttfamily . 233, 235,
237, 6507, 8063,
8065, 8067,
9083, 11190, 11353

<code>\twocolglossary</code> .. 9142	<code>\TX@get@body</code>	11208, 11215,
<code>\twocolindex</code> 6394, 6396 , 6402	11221, 11228, 11234
<code>\twocoltocetc</code> 8195	<code>\TX@newcol</code> . 6405, 6460	<code>\unitnumbername</code> ...
<code>\twocolumn</code> 3221, 3271,	<code>\TX@old@col</code> 12765 ,
3518, 3609,	.. 6384, 6432, 6449	12805, 12894,
5022, 8202,	<code>\TX@old@table</code> . 6383,	12901, 12917,
8208, 8881,	6410, 6431, 6450	12939, 12955,
8892, 9158,	<code>\TX@target</code> ... 6385,	12977, 13001, 13003
9169, 13082, 13254	6392, 6393,	<code>\unitordinalname</code> ..
<code>\twocolumnfootnotes</code>	6411, 6423, 6436 12820 ,
..... 10080	<code>\TX@trial</code>	12861, 12989, 12991
<code>\twocolumnfootstyle</code>	.. 6415, 6420, 6461	<code>\unkern</code> 9640
..... 10126	<code>\TX@trial@ftn</code> 6464, 6503	<code>\unletcounter</code> ... 12298
<code>\TX@</code> 6390, 6400, 6403 , 6428	<code>\TX@true</code> 6452	<code>\unnamedsubappendices</code>
<code>\TX@align</code>	<code>\TX@typeout</code> 4956
.. 6479, 6481, 6484	.. 6393, 6492, 6494	<code>\unpenalty</code> .. 10291,
<code>\TX@arith</code> .. 6418, 6429	<code>\TX@typeout@</code> .. 6413,	10308, 11378, 11894
<code>\TX@ckpt</code> ... 6408, 6477	6433, 6438,	<code>\unrestored@protected@xdef</code>
<code>\TX@col@width</code>	6478, 6493, 6495	117, 9707, 9718,
... 6382, 6405,	<code>\TX@v</code> 6510, 6511	10834, 10847, 10859
6411, 6416,	<code>\TX@v@</code> . 6517, 6519, 6523	<code>\unskip</code> . 3321, 3322,
6432, 6444,	<code>\TX@v@hash</code> . 6524, 6525	5043, 5064,
6447, 6449,	<code>\TX@vb</code> 6508, 6509	5131, 5663,
6451, 6482,	<code>\TX@verb</code> ... 6406, 6504	6057, 6193,
6877, 6878,	<code>\TX@vfirst</code> . 6511, 6515	6689, 6690,
6880, 6890,	<code>\TX@vwarn</code>	6695, 6696,
6897, 6964,	.. 6465, 6506, 6527	6907, 7008,
6966, 6968,	<code>\TX@xftntext</code> 6422, 6497	9615, 9825,
6970, 6985, 6997	<code>\typeoutlayout</code>	10289, 10290,
<code>\TX@cols</code> 2231 , 2275	10802, 11577, 12026
6412, 6416,	<code>\typeoutstandardlayout</code>	<code>\unvbox</code> . 4499, 6374,
6440, 6441, 2302	9599, 9750,
6443, 6483,		9773, 9779,
6855, 6870,		9795, 9830,
6871, 6878,		9836, 9882,
6912, 6914,		9932, 10037,
6942, 6944,		10040, 10078,
6958, 6968,		10115, 10158,
6978, 6988, 6989		10218, 10259,
<code>\TX@delta</code>		10286, 10354,
.. 6386, 6437, 6457		10405, 10426,
<code>\TX@endtabularx</code> ...	<code>\unhcopy</code> 6185	10434, 10462,
..... 6400, 6404	<code>\unitlength</code> ... 2938,	10508, 10518,
<code>\TX@error@width</code> ...	2945, 11064,	10524, 10530,
..... 6447, 6456	11072, 11079,	10535, 10542,
<code>\TX@false</code> 6430	11086, 11093,	10588, 10590,
<code>\TX@find@end</code> 6397, 6398	11101, 11113,	10591, 10713,
<code>\TX@ftn</code> . 6426, 6496 ,	11119, 11125,	10721, 10753,
6498, 6500, 6502	11131, 11187,	10755–10757,
<code>\TX@ftntext</code> . 6422, 6497	11195, 11202,	11729, 11730,

U

- 11861, 11871, 11894
`\uppercaseheads` . . . 2657
`\uppermargin`
 . . . 1948, 1959,
 1964, 2052–
 2056, 2108,
 2109, 2141,
 2179, 2184,
 2191, 2198,
 2214, 2245,
 2327, 2329,
 2335, 2336,
 2338, 2344, 2346
`\upshape` 8133
`\usebox` 7570,
 7766, 7769,
 7772, 7781,
 7782, 7788, 7789
`\usepackage` 3–5
`\usethanksrule` . . . 3195
- V**
- `\v@rid` 11032,
 11043, 11049, 11055
`\vadjust` 5044, 5066, 9484
`\valign` 6902, 10072
`\value` 3395,
 3745, 4926,
 4928, 4930,
 4932, 5803,
 6407, 7016,
 7017, 7386, 7388
`\vbadness` 10071
`\vbox` . . . 2549, 2559,
 2569, 2579,
 4487, 6121,
 6176, 7156,
 7503, 9112,
 9360, 9748,
 9771, 9794,
 9931, 10114,
 10157, 10217,
 10258, 10286,
 10341, 10353,
 10360, 10396,
 10404, 10411,
 10487, 10491,
 10505, 10515,
 10528, 10540,
 10719, 11162,
 11173, 11242,
 11253, 11272,
 11573, 11689,
 11729, 11730,
 11815, 11858,
 11871, 11892,
 11946, 11956, 13049
`\vcenter` 6118, 6176, 6374
`\verb` 6406,
 6528, 11592, 11615
`\verb@readstre@m`
 12194, 12197, 12224
`\verbatim` . 11380, 11573
`\verbatim*` 11380
`\verbatim@` . . . 11400,
 11421, 11422,
 11424, 11432,
 11445, 11454,
 11465, 11473, 11487
`\verbatim@@` 11424, 11425
`\verbatim@@@`
 11427, 11428
`\verbatim@t@testend`
 11467, 11477
`\verbatim@addtoline`
 . 11346, 11396,
 11426, 11441,
 11452, 11459,
 11461, 11469,
 11471, 11483,
 11485, 11512, 12208
`\verbatim@finish`
 . 11350, 11398,
 11479, 11506, 12202
`\verbatim@font`
 11352, 11377
`\verbatim@in@stream`
 11492,
 11495, 11496,
 11504, 11509, 11510
`\verbatim@input`
 11522, 11523, 11524
`\verbatim@line`
 . 11296, 11343,
 11344, 11346,
 11348, 11350,
 11561, 11569,
 11570, 11696, 12150
`\verbatim@out` 11553,
 11556, 11560, 11564
`\verbatim@processline`
 11346,
 11351, 11382,
 11387, 11397,
 11430, 11443,
 11463, 11513,
 11520, 11559,
 11568, 11687,
 12149, 12185, 12209
`\verbatim@read@file`
 11502, 11508
`\verbatim@read@stre@m`
 12200, 12204
`\verbatim@readfile`
 11493, 11526, 11750
`\verbatim@rescan`
 11481, 11490
`\verbatim@start`
 11384,
 11389, 11418,
 11562, 11714, 12151
`\verbatim@startline`
 11346,
 11395, 11419,
 11431, 11444,
 11464, 11494,
 11514, 12198, 12210
`\verbatim@tabexpand`
 11308, 11343
`\verbatim@test`
 11436, 11438
`\verbatim@testend`
 11450, 11457
`\verbatimbreakchar`
 11528, 11545
`\verbatimindent`
 11528, 11546
`\verbatiminput` . . . 11518
`\verbatimoutput` . . . 11554
`verbatimoutput (environment)` . . . 11553
`\verbfootnote` 9711
`verse (environment)` 5681
`\verselinebreak`
 5641, 5665
`\verselinenumbersleft`
 5650
`\verselinenumbersright`
 5650
`\versewidth` 5632

- `\vfuzz` 12472
`\vgap` [5633](#), 5819
`\vin` [5633](#), 5673
`\vindented` [5633](#), 5687,
 5696, 5700, 5703
`\vinphantom` [5636](#)
`\vleftmargin` [5638](#), 5689
`\vleftoffline` [5637](#)
`\vleftskip` . [5620](#), 5629
`\vline` 6129
`\vlvnumfont`
 ... [5570](#), 5657,
 5661, 11660,
 11671, 11677, 13247
`vminipage` (environ-
 ment) [3100](#)
`\vphantom` 3886,
 3912, 3941,
 3958, 3978,
 4024, 4103, 4179
`vplace` (environment)
 [12424](#)
`\vrb@catcodes`
 11410, 11413
`\vrightskip`
 .. [5620](#), 5655, 5659
`\vrule` 3040,
 3110, 5047,
 6158, 6165,
 6195, 6763,
 9103, 9105,
 9374, 11737,
 11738, 11757,
 11758, 11778,
- 11779, 11947, 12019
`\vsize` 11830, 12358, 13090
`\vsplit` 10077,
 10462, 11867, 13065
`\vss` 4487, 4489,
 9125, 9503,
 9507, 9599,
 10714, 11181, 11261
`\vtop` ... 5127, 5128,
 6120, 6176,
 7502, 9498,
 9504, 9595, 10701
- W**
- `\widowpenalty` [1370](#), 8815
`\width` 11790,
 11816, 12010,
 12020, 12038, 12051
`\wlog` 148
`\wo@daddtodef` [358](#)
`\wo@daddtoiargdef` . [369](#)
`\wo@difmacro@begingroup`
 [337](#), 360, 371
`\wo@dmacro` [333](#), 340, 347
`\wo@dparsemacro` 340, [345](#)
`\wrappingoff` [11537](#)
`\wrappingon` [11537](#)
`\wrapright` 11379, [11537](#)
`\write` 206, 210, 6468,
 8983, 9233,
 10928, 11560,
 12150, 12158,
 12332, 13146, 13153
`\writes` 203
`\writeverbatim` .. 12145
- `writeverbatim` (envi-
 ronment) . [12145](#)
- X**
- `\X` [8224](#)
`X` (environment) ... [7039](#)
`\x` 339, 342, 346, 348,
 363, 365, 374,
 376, 11293, 11294
`X*` (environment) .. [7039](#)
`\xdef` 145, 6023,
 6171, 6217,
 6737, 6770,
 7420, 10333,
 10386, 10464,
 10477, 10498,
 11586, 11625,
 11629, 12247, 12272
`\xetexfalse` [290](#)
`\xetextrue` [290](#)
`\xindyindex` [9029](#)
`\xlvchars`
 [69](#), [1717](#), 1760–1763
- Z**
- `\z@skip` 5031,
 5039, 5087,
 5095, 5096,
 9107, 11702,
 11705, 11948, 11961
`\Zdepth` [8220](#)
`\zero@glue` 11797, [11961](#)
`\zerotrivseps` [5296](#)
`\Zheadstart` [8256](#)
`\Zmark` [8223](#)