The Memoir Class

The Memoir Class

for

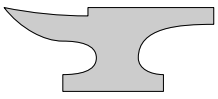
Configurable Typesetting

User Guide

Peter Wilson

Maintained by Lars Madsen

Corresponding to memoir version v3.7n, 2020/10/04



© 2001 — 2010 Peter R. Wilson
© 2011 — Lars Madsen

All rights reserved

The Herries Press, Normandy Park, WA.

Printed in the World

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

10 09 08 07 06 05 04 03 02 01 19 18 17 16 15 14 13

First edition:	3 June 2001
Second impression, with corrections:	2 July 2001
Second edition:	14 July 2001
Second impression, with corrections:	3 August 2001
Third impression, with minor additions:	31 August 2001
Third edition:	17 November 2001
Fourth edition:	16 March 2002
Fifth edition:	10 August 2002
Sixth edition:	31 January 2004
Seventh edition:	10 May 2008
Eighth impression, with very minor corrections:	12 July 2008
Ninth impression, with additions and corrections:	8 July 2009
Eighth edition:	August 2009
Tenth impression, with additions and corrections:	11 November 2015

memoir, *n.* a written record set down as material for a history or biography: a biographical sketch: a record of some study investigated by the writer: (in *pl.*) the transactions of a society. [Fr. *mémoire* — L. *memoria*, memory — *memor*, mindful.]

Chambers Twentieth Century Dictionary, New Edition, 1972.

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

Webster's New World Dictionary, Second College Edition.

memoir, *n.* a fiction designed to flatter the subject and to impress the reader.

With apologies to Ambrose Bierce

Short contents

Short contents · vii
Contents · ix
List of Figures · xvii
List of Tables · xx
List of typeset examples · xxii
Preface · xxiii
Introduction to the eighth edition · xxv
Remarks to the user · xxxiii
Terminology · xxxv
1 Starting off · 1
2 Laying out the page · 7
3 Text and fonts · 35
4 Titles · 57
5 Abstracts · 71
6 Document divisions · 75
7 Pagination and headers · 107
8 Paragraphs and lists · 127
9 Contents lists · 145
10 Floats and captions · 173

11 Rows and columns · 211

12 Page notes · 237

13 Decorative text · 257

14 Poetry · 263

15 Boxes, verbatims and files · 279

16 Cross referencing · 301

17 Back matter · 305

18 Miscellaneous · 325

19 For package users · 351

20 An example book design · 355

21 An example thesis design · 363

A Packages and macros · 385

B Showcases · 389

C Sniplets · 407

D Pictures · 415

E LaTeX and TeX · 435

F The terrors of errors · 453

G Comments · 477

Notes · 479

Command summary · 481

Bibliography · 523

Index · 531

Index of first lines · 575

Contents

Short contents vii

Contents ix

List of Figures xvii

List of Tables xx

List of typeset examples xxii

Preface xxiii

Introduction to the eighth edition xxv
- General considerations . . . xxv
- Class options . . . xxvi
- Sectioning styles . . . xxvi
- Captions . . . xxvii
- Tables . . . xxvii
- Verse . . . xxviii
- End matter . . . xxix
- Miscellaneous . . . xxix
- Packages . . . xxix
- Resources . . . xxx
- Type conventions . . . xxxi
- Acknowledgements . . . xxxi

Remarks to the user xxxiii

Terminology xxxv
- Units of measurement . . . xxxvi

1 Starting off 1
- 1.1 Stock paper size options . . . 1
- 1.2 Type size options . . . 2
  - *1.2.1 Extended font sizes* 3
- 1.3 Printing options . . . 4
- 1.4 Other options . . . 5

1.5 Remarks 5

2 Laying out the page 7
2.1 Introduction 7
2.2 Stock material 7
2.3 The page 8
2.4 The typeblock 13
2.4.1 A note on the width of the typeblock 13, *2.4.2 Specifying the typeblock size* 16
2.5 Headers, footers and marginal notes 20
2.6 Other 22
2.7 Putting it together 22
2.8 Side margins 26
2.9 Emitting the page size 26
2.10 Example 27
2.10.1 The page layout of this manual 27
2.11 Predefined layouts 29
2.12 Place trimmed page on the stock 32

3 Text and fonts 35
3.1 Fonts 35
3.2 Font sizes 44
3.3 Spaces 49
3.3.1 Paragraphs 49, *3.3.2 Double spacing* 50
3.4 Overfull lines 52
3.5 Sloppybottom 52
3.6 Text case 54
3.6.1 Nested text 55

4 Titles 57
4.1 Styling the titling 63
4.2 Styling the thanks 67

5 Abstracts 71
5.1 Styling 71
5.2 One column abstracts 73

6 Document divisions 75
6.1 Logical divisions 75
6.2 Sectional divisions 76
6.2.1 Appendices 78
6.3 Numbering 79
6.4 Book and part headings 80
6.4.1 Leadpage 83
6.5 Chapter headings 84
6.5.1 Defining a chapter style 87, *6.5.2 Further chapterstyles* 90, *6.5.3 Chapter precis* 92

6.6 Lower level headings . . . 94
6.7 Fancy anonymous breaks . . . 100
6.8 Footnotes in division headings . . . 102
6.9 Predefined heading styles . . . 103

7 Pagination and headers 107
7.1 Pagination and folios . . . 107
7.2 Page styles . . . 108
7.3 Making headers and footers . . . 111
7.3.1 Example pagestyles 116, *7.3.2 Index headers* 121, *7.3.3 Float pages* 121
7.4 The showlocs pagestyle . . . 124
7.5 Other things to do with page styles . . . 124

8 Paragraphs and lists 127
8.1 Paragraphs . . . 127
8.1.1 Block paragraph 127, *8.1.2 Hanging paragraphs* 128
8.2 Flush and ragged . . . 129
8.3 Quotations . . . 130
8.4 Some less common paragraph shapes . . . 131
8.4.1 Last line not short 132, *8.4.2 Russian typography* 132, *8.4.3 Fill with rules* 133, *8.4.4 Some ragged paragraphs* 133, *8.4.5 Left spring right* 134
8.5 Changing the textwidth . . . 135
8.6 Lists . . . 137

9 Contents lists 145
9.1 General ToC methods . . . 145
9.2 The class ToC methods . . . 150
9.2.1 Changing the titles 151, *9.2.2 Typesetting the entries* 153, *9.2.3 Example: No section number* 163, *9.2.4 Example: Multicolumn entries* 163, *9.2.5 Example: Multiple contents* 164
9.3 New 'List of…' and entries . . . 167
9.3.1 Example: plates 170
9.4 Chapter precis . . . 171
9.5 Contents lists and bookmarks . . . 172

10 Floats and captions 173
10.1 New float environments . . . 173
10.1.1 Margin floats 175
10.2 Setting off a float . . . 175
10.3 Multiple floats . . . 177
10.4 Where LaTeX puts floats . . . 181
10.5 Captions . . . 187
10.6 Caption styling . . . 187
10.7 Continuation captions and legends . . . 191
10.8 Bilingual captions . . . 196
10.9 Subcaptions . . . 199
10.10 Side captions . . . 202

10.10.1 Tweaks 204
10.11 How LaTeX makes captions . . . 206
10.12 Footnotes in captions . . . 209
10.13 The class versus the caption package (and its friends) . . . 210

11 Rows and columns 211
11.1 General . . . 211
11.2 The preamble . . . 212
11.2.1 D column specifiers 213, *11.2.2 Defining new column specifiers* 216, *11.2.3 Surprises* 217
11.3 The array environment . . . 218
11.4 Tables . . . 220
11.5 Fear's rules . . . 222
11.5.1 Fills 224
11.6 Tabular environments . . . 226
11.6.1 Examples 226
11.7 Spaces and rules . . . 230
11.7.1 Spacing 230, *11.7.2 Special variations on horizontal lines* 230, *11.7.3 Handling of rules* 231
11.8 Free tabulars . . . 231
11.8.1 Continuous tabulars 232, *11.8.2 Automatic tabulars* 233

12 Page notes 237
12.1 Footnotes . . . 237
12.1.1 A variety of footnotes 238, *12.1.2 Styling* 240
12.2 Marginal notes . . . 244
12.3 Side notes . . . 245
12.4 Sidebars . . . 246
12.5 Side footnotes . . . 248
12.5.1 Bottom aligned side footnotes 249, *12.5.2 Setting the layout for* `\sidefootnote` 249, *12.5.3 Styling* `\sidefootnote` 250, *12.5.4 Side footnote example* 251
12.6 Endnotes . . . 251
12.6.1 Changing the appearance 254

13 Decorative text 257
13.1 Epigraphs . . . 257
13.2 General . . . 258
13.3 Epigraphs before chapter headings . . . 259
13.3.1 Epigraphs on book or part pages 261

14 Poetry 263
14.1 Classy verse . . . 265
14.1.1 Indented lines 269, *14.1.2 Numbering* 270
14.2 Titles . . . 270
14.2.1 Main Poem Title layout parameters 271, *14.2.2 Detailed Poem Title layout parameters* 272

14.3 Examples . . . 273
A Limerick 273, *Love's lost* 273, *Fleas* 274, *In the beginning* 274, *Mathematics* 275, *The Young Lady of Ryde* 276, *Clementine* 276, *Mouse's Tale* 278

15 Boxes, verbatims and files 279
15.1 Boxes . . . 280
15.2 Long comments . . . 284
15.3 Verbatims . . . 285
15.3.1 Boxed verbatims 287, *15.3.2 New verbatims* 289, *15.3.3 Example: the* `lcode` *environment* 290
15.4 Files . . . 292
15.4.1 Writing to a file 293, *15.4.2 Reading from a file* 293, *15.4.3 Example: endnotes* 294, *15.4.4 Example: end floats* 294, *15.4.5 Example: questions and answers* 297
15.5 Answers . . . 300

16 Cross referencing 301
16.1 Labels and references . . . 301
16.2 Reference by name . . . 302

17 Back matter 305
17.1 Bibliography . . . 305
17.1.1 BibTex 307
17.2 Index . . . 308
17.2.1 Printing an index 308, *17.2.2 Preparing an index* 309, *17.2.3 MakeIndex* 312, *17.2.4 Controlling MakeIndex output* 315, *17.2.5 Indexing and the natbib package* 318
17.3 Glossaries . . . 318
17.3.1 Controlling the glossary 319

18 Miscellaneous 325
In which we talk of many things, but not shoes or ships or sealing wax, nor cabbages and kings.
18.1 Draft documents . . . 325
18.2 Change marks . . . 325
18.3 Trim marks . . . 326
18.4 Sheet numbering . . . 328
18.5 Gatherings or signatures . . . 330
18.6 Time . . . 330
18.7 Page breaks before lists . . . 330
18.8 Changing counters . . . 331
18.9 New new and provide commands . . . 332
18.10 Changing macros . . . 332
18.11 String arguments . . . 333
18.12 Odd/even page checking . . . 334
18.13 Moving to another page . . . 335
18.14 Number formatting . . . 336

18.14.1 Numeric numbers 336, *18.14.2 Named numbers* 337, *18.14.3 Fractions* 339
18.15 An array data structure . . . 340
18.16 Checking the processor . . . 341
18.16.1 Checking for pdfLaTeX 341, *18.16.2 Checking for etex* 341, *18.16.3 Checking for XeTeX* 342, *18.16.4 Checking for LuaTeX* 342
18.17 Leading . . . 342
18.18 Minor space adjustment . . . 342
18.19 Adding a period . . . 343
18.20 Words and phrases . . . 343
18.21 Symbols . . . 343
18.22 Two simple macros . . . 345
18.23 Vertical centering . . . 345
18.24 For package writers . . . 345
18.24.1 Emulating packages 345, *18.24.2 Inserting code before and after a file, package or class* 346
18.25 Heading hooks . . . 348
18.26 Documenting LaTeX commands . . . 349

19 For package users 351
19.1 Class/package name clash . . . 351
19.2 Support for bididirectional typesetting . . . 352

20 An example book design 355
20.1 Introduction . . . 355
20.2 Design requirements . . . 355
20.3 Specifying the page and typeblock . . . 356
20.4 Specifying the sectional titling styles . . . 358
20.4.1 The chapter style 358, *20.4.2 Lower level divisions* 358
20.5 Specifying the pagestyle . . . 359
20.6 Captions and the ToC . . . 361
20.7 Preamble or package? . . . 361

21 An example thesis design 363
21.1 Example US thesis typographic requirements . . . 363
21.1.1 General 363, *21.1.2 Preliminary matter* 364, *21.1.3 Table of contents* 365, *21.1.4 Lists* 366, *21.1.5 Main text* 366, *21.1.6 Backmatter* 367
21.2 Code . . . 367
21.2.1 Initialisation 368, *21.2.2 Page layout* 368, *21.2.3 Page styles* 370, *21.2.4 The ToC and friends* 371, *21.2.5 Chapter styling* 372, *21.2.6 Section, etc., styling* 372, *21.2.7 Captions* 373, *21.2.8 The bibliography* 373, *21.2.9 End notes* 373, *21.2.10 Preliminary headings* 374, *21.2.11 Components of the title and approval pages* 375, *21.2.12 The title and approval pages* 375, *21.2.13 The last bits* 379
21.3 Usage . . . 379
21.4 Comments . . . 381

A Packages and macros 385
A.1 Packages . . . 385
A.2 Macros . . . 386

B Showcases 389
B.1 Chapter styles . . . 389
B.1.1 Chappell 402, *B.1.2 Demo, Demo2 and demo3* 403, *B.1.3 Pedersen* 403, *B.1.4 Southall* 404, *B.1.5 Veelo* 405

C Sniplets 407
Sniplet C.1 (Mirroring the output) . . . 407
Sniplet C.2 (Remove pagenumber if only one page) . . . 408
Sniplet C.3 (A kind of draft note) . . . 408
Sniplet C.4 (Adding indentation to footnotes) . . . 409
Sniplet C.5 (Background image and trimmarks) . . . 409
Sniplet C.6 (Autoadjusted number widths in the ToC) . . . 409
Sniplet C.7 (Using class tools to make a chapter ToC) . . . 411
Sniplet C.8 (An appendix ToC) . . . 413

D Pictures 415
D.1 Basic principles . . . 415
D.2 Picture objects . . . 417
D.2.1 Text 417, *D.2.2 Boxes* 418, *D.2.3 Lines* 423, *D.2.4 Arrows* 425, *D.2.5 Circles* 425
D.3 Repetitions . . . 428
D.4 Bezier curves . . . 431

E LaTeX and TeX 435
E.1 The TeX process . . . 436
E.2 LaTeX files . . . 437
E.3 Syntax . . . 438
E.4 (La)TeX commands . . . 439
E.5 Calculation . . . 442
E.5.1 Numbers 442, *E.5.2 Lengths* 443
E.6 Programming . . . 447

F The terrors of errors 453
F.1 TeX messages . . . 454
F.1.1 TeX capacity exceeded 462
F.2 LaTeX errors . . . 464
F.3 LaTeX warnings . . . 468
F.4 Class errors . . . 471
F.5 Class warnings . . . 474

G Comments 477
G.1 Algorithms . . . 477
G.1.1 Autoadjusting `\marginparwidth` 477

Notes 479
Chapter 3 Text and fonts 479
Chapter 12 Page notes 480

Command summary 481

Bibliography 523

Index 531

Index of first lines 575

List of Figures

2.1 LaTeX page layout parameters for a recto page . . . 9
2.2 The memoir class page layout parameters for a verso page . . . 10
2.3 The memoir class page layout parameters for a recto page . . . 11
2.4 The recto page layout for this manual . . . 28
2.5 Default layout for letterpaper . . . 30
2.6 Letterpaper layout: Left `\medievalpage`, Right `\medievalpage[12]` . . . 30
2.7 Letterpaper layout: Left `\isopage`, Right `\isopage[12]` . . . 30
2.8 Letterpaper layout: Left `\semiisopage`, Right `\semiisopage[12]` . . . 30
2.9 Default layout for a4paper . . . 31
2.10 A4paper layout: Left `\medievalpage`, Right `\medievalpage[12]` . . . 31
2.11 A4paper layout: Left `\isopage`, Right `\isopage[12]` . . . 31
2.12 A4paper layout: Left `\semiisopage`, Right `\semiisopage[12]` . . . 31
2.13 Example of the nine `\setpageXX` macros for placing a trimmed page on the stock. These are all odd pages, under twoside, even pages will have the left/right trims reversed. The outer fra indicate the stock size. Inside the trim marks hint at where the page is positioned on the stock. . . . 34

4.1 Layout of a title page for a book on typography . . . 58
4.2 Example of a mandated title page style for a doctoral thesis . . . 59
4.3 Example of a Victorian title page . . . 60
4.4 Layout of a title page for a book on book design . . . 61
4.5 Layout of a title page for a book about books . . . 62

6.1 Class layout parameters for chapter titles. Working with `\beforechapskip` need a little thought, see the text. . . . 85
6.2 Displayed sectional headings . . . 94
6.3 Run-in sectional headings . . . 95

7.1 Header and footer slots . . . 111

8.1 Paragraphing parameters . . . 127
8.2 The layout parameters for general lists . . . 141

9.1 Example extracts from `toc`, `lof` and `lot` files . . . 147
9.2 Layout of a ToC (LoF, LoT) entry . . . 147

10.1 Example framed figure . . . 176

10.2 Example framed figure and caption . . . 176
10.3 Example ruled figure . . . 177
10.4 Example ruled figure and caption . . . 177
10.5 Example float with two illustrations . . . 177
10.6 Graphic 1 in a float . . . 178
10.7 Graphic 2 in same float . . . 178
10.8 Left center aligned . . . 179
10.9 Right figure. This has more text than the adjacent caption (10.8) so the heights are unequal . . . 179
10.10 Left top aligned . . . 180
10.11 Right figure. This has more text than the adjacent caption (10.10) so the heights are unequal . . . 180
10.12 Left bottom aligned . . . 181
10.13 Right figure. This has more text than the adjacent caption (10.12) so the heights are unequal . . . 181
10.14 Float and text page parameters . . . 183
10.15 Float parameters . . . 184
10.16 Long `\bitwonumcaption` . . . 197
10.16 Lang `\bitwonumcaption` . . . 197
10.17 Long English `\bionenumcaption` . . . 197
Lang Deutsch `\bionenumcaption` . . . 197
10.18 Short English `\bicaption` . . . 198
10.19 Figure with two subfigures . . . 201
(a) Subfigure 1 . . . 201
(b) Subfigure 2 . . . 201
10.20 A picture is worth a thousand words . . . 208
10.21 A different kind of figure caption . . . 208

11.1 Example of a regular `tabular` . . . 226
11.2 Example `tabularx` and `tabular*` with widths of 250pt . . . 227
11.3 Example `tabularx` and `tabular*` with widths of 300pt . . . 228
11.4 Changing the width of a row ordered table . . . 234
11.5 Changing the width of a column ordered table . . . 235

12.1 Footnote layout parameters . . . 241
12.2 Interpretation of the arguments to the `\Xmargin` commands for specifying the side in which to place side note like material. `X` here equals `marginpar`, `sidepar`, `sidebar`, or `sidefoot`. . . . 245
12.3 Example endnote listing . . . 253

17.1 Raw indexing: (left) index commands in the source text; (right) `idx` file entries 312
17.2 Processed index: (left) alphabeticized `ind` file; (right) typeset index . . . 313

18.1 The four trimmark types . . . 329
(a) `\trimXmarks` (default) . . . 329
(b) `\trimLmarks` . . . 329
(c) `\trimFrame` . . . 329

(d) `\quarkmarks` 329

21.1 Example Archibald Smythe University title page 376
21.2 Example Archibald Smythe University approval page 377

B.1 The default chapterstyle 389
B.2 The section chapterstyle 390
B.3 The hangnum chapterstyle 390
B.4 The companion chapterstyle 391
B.5 The article chapterstyle 391
B.6 The bianchi chapterstyle 392
B.7 The bringhurst chapterstyle 392
B.8 The brotherton chapterstyle 393
B.9 The chappell chapterstyle 393
B.10 The crosshead chapterstyle 394
B.11 The culver chapterstyle 394
B.12 The dash chapterstyle 394
B.13 The demo2 chapterstyle 395
B.14 The dowding chapterstyle 395
B.15 The ell chapterstyle 396
B.16 The ger chapterstyle 396
B.17 The komalike chapterstyle 397
B.18 The lyhne chapterstyle. This style requires the graphicx package 397
B.19 The madsen chapterstyle. This style requires the graphicx package 398
B.20 The ntglike chapterstyle 399
B.21 The southall chapterstyle 399
B.22 The tandh chapterstyle 400
B.23 The thatcher chapterstyle 400
B.24 The veelo chapterstyle. This style requires the graphicx package 401
B.25 The verville chapterstyle 401
B.26 The wilsondob chapterstyle 402

D.1 Some points in the cartesian coordinate system 416
D.2 Specification of a line or arrow 425
D.3 Sloping lines and arrows 426
D.4 Some measuring scales 429
D.5 Two Bezier curves 433

List of Tables

1 Traditional font size designations . . . xxxvi
2 Printers units . . . xxxvi

1.1 Class stock metric paper size options, and commands . . . 1
1.2 Class stock US paper size options, and commands . . . 2
1.3 Class stock British paper size options, and commands . . . 2

2.1 Arguments and results for `\settrimmedsize` and `\settypeblocksize` . . . 12
2.2 Average characters per line . . . 14
2.3 Lowercase alphabet lengths, in points, for various fonts . . . 16
2.4 Arguments and results for `\setlrmargins` . . . 18
2.5 Arguments and results for `\setlrmarginsandblock` . . . 18
2.6 Arguments and results for `\setulmargins` . . . 20
2.7 Arguments and results for `\setulmarginsandblock` . . . 20
2.8 Arguments and results for `\setheaderspaces` . . . 21
2.9 The class and LaTeX page layout parameters . . . 24
2.10 Results from sample `\textheight` adjustments . . . 25

3.1 Glyphs in the LaTeX supplied Palatino roman font . . . 36
3.2 Glyphs in the LaTeX distributed Symbol font . . . 37
3.3 Glyphs in the LaTeX distributed Zapf Dingbat font . . . 38
3.4 Font categorisation and commands . . . 42
3.5 Font declarations . . . 42
3.6 Standard font size declarations . . . 45
3.7 Standard font sizes . . . 45
3.8 The memoir class font size declarations . . . 46
3.9 The memoir class font sizes . . . 46

6.1 Division levels . . . 79
6.2 Default display sectioning layout parameter values . . . 95
6.3 Default run-in sectioning layout parameter values . . . 95
6.4 Values for `S` in section styling macro names. . . . 95
6.5 Default fonts for sectional headings . . . 103
6.6 Fonts used by different headstyles . . . 105

7.1 The use of `\thispagestyle` . . . 110
7.2 Mark macros for page headers . . . 111

9.1 Indents and Numwidths . . . 148
9.2 Values for X in macros for styling the titles of 'List of…' . . . 151
9.3 Value of K in macros for styling entries in a 'List of…' . . . 154

10.1 Float placement parameters . . . 185
10.2 Float spacing parameters . . . 185
10.3 Redesigned table caption style . . . 190
10.4 A multi-part table . . . 192
10.5 Another table . . . 193
Legendary table (toc 1) . . . 193
Legendary table (toc 2) . . . 193
10.6 Permitted arguments for some sidecaption related commands . . . 205

11.1 The array and tabular preamble options. . . . 212
11.2 Demonstrating the parts of a table . . . 220
11.3 Two views of one table . . . 221
11.4 Micawber's law . . . 222
11.5 A narrow table split half and half . . . 222
11.6 Example table with fills . . . 225
11.7 Example automatic row ordered table . . . 233

12.1 Some footnote text styles . . . 242

17.1 MakeIndex configuration file input parameters . . . 313
17.2 MakeIndex configuration file output parameters . . . 316

18.1 Defined words and phrases . . . 344

E.1 Some internal macros for numbers . . . 443

List of typeset examples

3.1 Badly mixed fonts . . . 42
3.2 Sometimes mixed fonts work . . . 43
3.3 Emphasis upon emphasis . . . 44
4.1 Example `\maketitle` title . . . 64
6.1 A variety of subhead styles . . . 99
8.1 Setting the source of a quotation . . . 130
8.2 Paragraph's line not too short . . . 132
8.3 Rules for spaces . . . 133
8.4 Ragged paragraphs . . . 134
8.5 A sprung paragraph . . . 135
8.6 Smallcap quote style description list . . . 139
8.7 Changing space before and after lists . . . 143
11.1 Tabular with narrow and wide headings . . . 216
14.1 Phantom text in verse . . . 268
14.2 Verse with regular quote marks . . . 269
14.3 Verse with hanging left quote marks . . . 269
16.1 Named references should be to titled elements . . . 303
16.2 Current title . . . 304
18.1 TeX's minimum number in words (English style) . . . 337
18.2 TeX's maximum number in words (American style) . . . 338
18.3 Varieties of fractions in text . . . 339
18.4 Super- and subscripts in text . . . 340
D.1 Picture: text . . . 418
D.2 Picture: text in boxes . . . 419
D.3 Picture: positioning text . . . 420
D.4 Picture: dashed box . . . 420
D.5 Picture: framing . . . 421
D.6 Picture: stacking . . . 422
D.7 Picture: saved boxes . . . 423
D.8 Picture: circles . . . 426
D.9 Picture: ovals . . . 427
D.10 Picture: text in oval . . . 428
D.11 Picture: repetitions . . . 431

Preface

From personal experience and also from lurking on the comp.text.tex newsgroup the major problems with using LaTeX are related to document design. Some years ago most questions on CTT were answered by someone providing a piece of code that solved a particular problem, and again and again. More recently these questions are answered along the lines of 'Use the ——— package', and again and again.

I have used many of the more common of these packages but my filing system is not always well ordered and I tend to mislay the various user manuals, even for the packages I have written. The memoir class is an attempt to integrate some of the more design-related packages with the LaTeX book class. I chose the book class as the report class is virtually identical to book, except that book does not have an `abstract` environment while report does; however it is easy to fake an `abstract` if it is needed. With a little bit of tweaking, book class documents can be made to look just like article class documents, and the memoir class is designed with tweaking very much in mind.

The memoir class effectively incorporates the facilties that are usually accessed by using external packages. In most cases the class code is new code reimplementing package functionalities. The exceptions tend to be where I have cut and pasted code from some of my packages. I could not have written the memoir class without the excellent work presented by the implementors of LaTeX and its many packages.

Apart from packages that I happen to have written I have gained many ideas from the other packages listed in the Bibliography. One way or another their authors have all contributed, albeit unknowingly. The participants in the comp.text.tex newsgroup have also provided valuable input, partly by questioning how to do something in LaTeX, and partly by providing answers. It is a friendly and educational forum.

PETER WILSON
Seattle, WA
June 2001

Introduction to the eighth edition

The memoir class and this manual have seen many changes since they first saw the light of day. The major functions, and extensions to them, were listed in the various introductions to the previous editions of this manual and it would now be tedious to read them.

The memoir class was first released in 2001 and since then has proven to be reasonably popular. The class can be used as a replacement for the book and report classes, by default generating documents virtually indistinguisable from ones produced by those classes. The class includes some options to produce documents with other appearances; for example an article class look or one that looks as though the document was produced on a typewriter with a single font, double spacing, no hyphenation, and so on. In the following I use the term 'standard class' to denote the book and report classes and, when appropriate, the article class as well.

The memoir class includes the functionality of many packages, for instance the tocloft package for controlling the table of contents or methods similar to the fancyhdr package for designing your own headers. The built-in package functions are mainly related to document design and layout; memoir does not touch upon areas like those that are covered by the babel or hyperref packages or any related to typesetting mathematics. On the other hand it is easy to configure a work produced with memoir to meet a university's thesis layout requirements.

The memoir class has improved substantially since it was first released — over 50 LaTeXers have provided code or suggestions for improvements. The class is included in the TeX Users Group TeX distributions and the latest version of the class and its supporting documentation is always available from CTAN at `latex/contrib/memoir`.

This is not a guide to the general use of LaTeX but rather concentrates on where the memoir class differs from the standard LaTeX book and report classes. There are other sources that deal with LaTeX in general, some of which are noted later. I assume that you have already used LaTeX and therefore know how to prepare a LaTeX manuscript, how to run LaTeX and print the resulting document, and that you can also use auxiliary programs like MakeIndex and BibTeX.

General considerations

The class is a large one consisting of about 10,000 lines of LaTeX code documented in a 400 page report; there is no need for most users to look at this [Wil19a]. However if you want to see exactly how some part, or all of, memoir is defined it is there for you to peruse. The document you are now reading is the separate comprehensive User Manual [Wil19b] which runs to about 500 pages, and from time to time an Addendum is released noting

extensions to the class.[1] Again, if you want to see how something was done in this Manual, which of course was prepared using memoir itself, the source is available for you to read.

The previous editions of the Manual consisted of two parts. The first discussing some aspects of book design and typography in general, something that I hadn't come across in the usual LaTeX books and manuals. That was intended to provide a little background for when you design your own printed documents. The second, and by far the longest part, described the capabilities of memoir and how to use them.

The Manual kept growing until it was approaching 700 pages and I decided that it was better to put the original discussion on typography into a separate document [Wil18], which is independent of memoir, and in this edition concentrate on how to use memoir. This has reduced the size of this document, but it is still large.

This manual is not a LaTeX tutorial; I assume that you already know the basics. If you don't then there are several free tutorials available. In some instances I show you the internal code for the class which may involve LaTeX commands that you won't come across in the tutorials and also sometimes basic TeX commands. Information on these, if you want it, is obtained from reading the LaTeX source itself and the TeXbook, and perhaps one of the free TeX manuals such as *TeX for the Impatient* [AHK90] or *TeX by Topic* [Eij92].

CLASS OPTIONS

The standard classes provide point options of 10, 11, or 12 points for the main body font. memoir extends this by also providing a 9 point option, and options ranging from 14 to 60 points. The width of the text block is automatically adjusted according to the selected point size to try and keep within generally accepted typographical limits for line lengths; you can override this if you wish. The class also provides easy methods for specifying the page layout parameters such as the margins — both the side margins and those at the top and bottom of the page; the methods are similar to those of the geometry package.

The page layout facilities also include methods, like those provided by the fancyhdr package, for defining your own header and footer styles, and you can have as many different ones as you wish. In fact the class provides seven styles to choose from before having to create your own if none of the built-in styles suit you.

Sometimes it is useful, or even required, to place trimming marks on each page showing the desired size of the final page with respect to the sheet of paper that is used in the printer. This is provided by the showtrims option. A variety of trim marks are provided and you can define your own if you need some other kind.

SECTIONING STYLES

Handles are provided for designing and using your own styles for chapter titles and such. The class comes with over 20 predefined chapter styles ranging from the default look to a style that mimics that used in the *Companion* series of LaTeX books. There are even a couple which use words instead of numerals for chapter numbers.

For those who like putting quotations near chapter titles the `epigraph` environment can be used.

[1] Currently not in use.

The options for changing `\section` and lower level titles are more constrained, but generally speaking document design, unless for advertisements or other eye-catching ephemera, should be constrained. The class does provide 9 integrated sets of sectional heading styles instead of the usual single set.

Sometimes, but particularly in novels, a sectional division is indicated by just leaving a blank line or two between a pair of paragraphs, or there might be some decorative item like three or four asterisks, or a fleuron or two. (A *fleuron* is a printers ornament looking like a leaf, such as ❦ or ❧.) Commands are available for typesetting such anonymous divisions.

In the standard classes the sectioning commands have an optional argument which can be used to put a short version of the section title into the table of contents and the page header. memoir extends this with a second optional argument so you can specify one short version for the contents and an even shorter one for page headers where space is at a premium.

CAPTIONS

memoir incorporates the code from my ccaption package which lets you easily modify the appearance of figure and table captions; bilingual captions are available if required, as are captions placed at the side of a figure or table or continuation captions from, say, one illustration to another. Captioning can also be applied to 'non-floating' illustrations or as legends (i.e., unnumbered captions) to the regular floats. The captioning system also supports subfigures and subtables along the lines of the subfig package, plus letting you define your own new kinds of floats together with the corresponding "List of…".

TABLES

The array, dcolumn, delarray, tabularx are automatically loaded by the class. In earlier days their code were embedded into the class, nowadays we just load the respective packages. To improve the appearance of rules in tabular material the booktabs package is also included (as a code copy, this may change in the future).

Multipage tabulations are often set with the longtable or xtab packages, which can of course be used with the class. For simple tabulations that may continue from one page to the next, memoir offers a 'continuous tabular' environment. This doesn't have all the flexibility provided by the packages but can often serve instead of using them.

More interestingly, but more limited, the class provides 'automatic tabulars'. For these you provide a list of simple entries, like a set of names, and a number of columns and the entries are automatically put into the appropriate column. You choose whether the entries should be added row-by-row, like this with the `\autorows` command:

```
\autorows{c}{5}{l}{one, two, three, four,
    five, six, seven, eight, nine, ten,
    eleven, twelve, thirteen }
```

one	two	three	four	five
six	seven	eight	nine	ten
eleven	twelve	thirteen		

Or if you use the `\autocols` command the entries are listed column-by-column, like this :

```
\autocols{c}{5}{l}{one, two, three, four,
    five, six, seven, eight, nine, ten,
    eleven, twelve, thirteen }
```

one	four	seven	ten	thirteen
two	five	eight	eleven	
three	six	nine	twelve	

VERSE

The standard classes provide a very simple `verse` environment for typesetting poetry. This is greatly extended in memoir. For example in the standard classes the verse stanzas are at a fixed indentation from the left margin whereas memoir lets you control the amount of indentation so that you can make a poem appear optically centered within the textwidth.

Stanzas may be numbered, as can individual lines within a poem. There is a special environment for stanzas where lines are alternately indented. Also you can define an indentation pattern for stanzas when this is not regular as, for example, in a limerick where the 3rd and 4th of the five lines are indented with respect to the other three as shown below.

```
\indentpattern{00110}
\begin{verse}
\begin{patverse}
There was a young man of Quebec \\
Who was frozen in snow to his neck. \\
When asked: ‘Are you friz?’ \\
He replied: ‘Yes, I is, \\
But we don’t call this cold in Quebec.’
\end{patverse}
\end{verse}
```

There was a young man of Quebec
Who was frozen in snow to his neck.
When asked: ‘Are you friz?’
He replied: ‘Yes, I is,
But we don’t call this cold in Quebec.’

It is not always possible to fit a line into the available space and you can specify the particular indentation to be used when a ‘logical’ verse line spills over the available textwidth, thus forming two or more typeset ‘physical’ lines. On other occasions where there are two half lines the poet might want the second half line to start where the first one finished, like this:

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

Come away with me.
Impossible!

END MATTER

Normally appendices come after the main body of a book. The class provides various methods for introducing appendices at the end, or you can place one or more appendices at the end of selected chapters if that suits you better.

memoir also lets you have more than one index and an index can be set in either the normal double column style or as a single column which would be more appropriate, say, for an index of first lines in a book of poetry. The titles of any bibliography or indexes are added to the table of contents, but you can prevent this if you wish.

The class provides a set of tools for making glossaries or lists of symbols, the appearance of which can, of course, be easily altered. The MakeIndex program is used to sort the entries. Also, the class provides configurable end notes which can be used as well as, or instead of, footnotes.

MISCELLANEOUS

Hooks and macros are provided for most aspects of document layout; for instance, footnotes can be as normal, typeset in two or three columns, or all run into a single paragraph. There is a `\sidepar` macro which is a non-floating `\marginpar` as well as the `\sidebar` macro for typesetting sidebars in the margin, starting at the top of the text block. You can create new verbatim-like environments, read and write information in external files, design your own style of `\maketitle`, convert numbers to words, reserve space at the bottom of a page, and so on and so forth.

PACKAGES

Most packages work with the memoir class, the main exception being the hyperref package. This package modifies many of the internals of the standard classes but does not cater for all of the differences between memoir and the standard ones.

If you use hyperref with memoir then the memhfixc package[2] is automatically loaded by hyperref to provide some class specific alterations.

The memoir class includes code either equivalent to, or extensions of, the following packages; that is, the set of commands and environments is at least the same as those in the packages:

abstract, appendix, booktabs, ccaption, chngcntr, chngpage, enumerate, epigraph, framed, ifmtarg, index, makeidx, moreverb, needspace, newfile, nextpage, parskip, patchcmd, setspace, shortvrb, showidx, titleref, titling, tocbibind, tocloft, verbatim, verse.

The class automatically ignores any `\usepackage` or `\RequirePackage` related to these. However, if you want to specifically use one of these packages rather than the integrated version then you can do so. For arguments sake, assuming you really want to use the titling package you can do this:

```
\documentclass[...]{memoir}
\DisemulatePackage{titling}
\usepackage{titling}
```

[2] memhfixc is supplied as part of the memoir distribution.

The memoir class incorporates a version of the setspace package, albeit using different names for the macros. The package enables documents to be set double spaced but leaves some document elements, like captions for example, single spaced. To do this it has to make some assumptions about how the document class works. I felt that this kind of capability should be part of the class and not depend on assumptions. In the particular case of the setspace package, even with the standard classes, there can be some unexpected spacing around displayed material; this has not occured with memoir's implementation.

The class also provides functions similar to those provided by the following packages, although the commands are different:

crop, fancyhdr, geometry, sidecap, subfigure, titlesec.

You can use these packages if you wish, or just use the capabilities of the memoir class.

The class has built in support for the bidi package for bidirectional typesetting [Kha10].

The following packages are automatically loaded by the class:

array, dcolumn, delarray, etex, iftex, tabularx, textcase (with `overload` option)

RESOURCES

Scattered throughout are comments about aspects of book design and typography, in some cases accompanied by examples of better and poorer practice. If you want more comments and examples there are some notes on the topic [Wil18], and for authorative remarks there are several books on the subject listed in the Bibliography; I prefer Bringhurst's *The Elements of Typographic Style* [Bri99], while Derek Birdsall's *notes on book design* [Bir04] is much more oriented to illustrated works, like museum catalogues and art books.

LaTeX is based on the TeX program which was designed principally for typesetting documents containing a lot of mathematics. In such works the mathematics breaks up the flow of the text on the page, and the vertical space required for displayed mathematics is highly dependent on the mathematical particularities. Most non-technical books are typeset on a fixed grid as they do not have arbitrary insertions into the text; it is these kinds of publications that typographers are most comfortable talking about.

There are other sources that deal with LaTeX in general, some of which are listed in the Bibliography. Lamport [Lam94] is of course the original user manual for LaTeX, while the Companion series [MG+04, GM+07, GR99] go into further details and auxiliary programs. George Grätzer's *Math into LaTeX* is valuable if you typeset a lot of mathematics with excellent coverage of the American Mathematical Society's packages.

The Comprehensive TeX Archive Network (CTAN) is an invaluable source of free information and of the LaTeX system itself. For general questions see the FAQ (Frequently Asked Questions, and answers) maintained by Robin Fairbairns [FAQ], which also has pointers to many information sources. Among these are *The Not So Short Introduction to LaTeX2e* [Oet], Keith Reckdahl's *Using imported graphics in LaTeX2e* [Rec97] and Piet van Oostrum's *Page layout in LaTeX* [Oos96]. Peter Flynn's *Formatting information* [Fly98] is unique in that it describes how to install a LaTeX system and editors for writing your documents as well as how to use LaTeX. There are a myriad of packages and software tools freely available to enhance any LaTeX system; the great majority of these are listed in Graham Williams' magnificent on line searchable catalogue [Wil00], which also links directly

to CTAN. This is just one of the services offered by the TeX Users Group (TUG) and information on how to access it is available at http://www.tug.org which is the homepage for the TeX Users Group.

The most recent crops of messages on the comp.text.tex newsgroup (CTT) show an increasing interest in using a wider range of fonts with LaTeX. This is a question that I have left alone. Alan Hoenig's book [Hoe98] is the best guide to this that I know of. CTAN hosts Philipp Lehman's font installation guide [Leh04]; this is well worth looking at just as an example of fine typesetting.

The source code for the memoir class is, of course, freely available from CTAN if you wish to see exactly what it does and how it does it.

For a more interactive resource you can ask questions on

http://tex.stackexchange.com.

– if the question is memoir related, please tag it so.

TYPE CONVENTIONS

The following conventions are used:

- The names of LaTeX classes and packages are typeset in this font.
- Class options are typeset in this font.
- *The names of chapterstyles and pagestyles are typeset in this font.*
- `LaTeX code is typeset in this font.`
- The names of programs are in this font.

Macro command syntax is enclosed in a rectangular box.
For referential purposes, arguments are denoted by ⟨*arg*⟩

ACKNOWLEDGEMENTS

Many people have contributed to the memoir class and this manual in the forms of code, solutions to problems, suggestions for new functions, bringing my attention to errors and infelicities in the code and manual, and last but not least in simply being encouraging. I am very grateful to the following for all they have done, whether they knew it or not: Paul Abrahams, William Adams, Tim Arnold, Donald Arseneau, Stephan von Bechtolsheim, Jens Berger, Karl Berry, Ingo Beyritz, Javier Bezos, Stefano Bianchi, Sven Bovin, Alan Budden, Ignasi Furió Caldentey, Ezequiel Martín Cámara, David Carlisle, Gustafo Cevolani, Jean-Côme Charpentier, Michael A. Cleverly, Steven Douglas Cochran, Frederic Connes, Žarko F. Čučej, Christopher Culver, Iain Dalton, Michael W. Daniels, Michael Downes, Christopher Dutchyn, Thomas Dye, Victor Eijkhout, Roman Eisele, Danie Els, Robin Fairbairns, Simon Fear, António Ferreira, Kai von Fintel, Ivars Finvers, Ulrike Fischer, Matthew Ford, Musa Furber, Daniel Richard G, Ignacio Fernández Galván, Gerardo Garcia, Romano Giannetti, Kheng-Swee Goh, Donald Goodman, Gabriel Guernik, Matthias Haldiman, Kathryn Hargreaves, Sven Hartrumpf, hazydirk, Carsten Heinz, Florence Henry, Peter Heslin, Timo Hoenig, Morten Høgholm, Henrik Holm, Vladimir G. Ivanović, Martin Jørgensen, Stefan Kahrs, Christian Keil, Marcus Kohm, Flavian Lambert, Jøgen Larsen, Kevin Lin, Matthew Lovell, Daniel Luecking, Anders Lyhne, Lars Hendrik

Gam Madsen, Lars Madsen, Vittorio De Martino, Ben McKay, Frank Mittelbach, Wilhelm Müller, Vilar Camara Neto, Rolf Niepraschk, Patrik Nyman, Heiko Oberdiek, Scott Pakin, Adriano Pascoletti, Paul, Ted Pavlic, Troels Pedersen, Steve Peter, François Poulain, Erik Quaeghebeur, Bernd Raichle, Martin Reinders, Aaron Rendahl, René, Alan Ristow, Robert, Chris Rowley, Gary Ruben, Robert Schlicht, Doug Schenck, Dirk Schlimm, Arnaud Schmittbuhl, Rainer Schöpf, Paul Stanley, Per Starbäck, James Szinger, Jens Taprogge, Ajit Thakkar, Scott Thatcher, Reuben Thomas, Bastiaan Niels Veelo, Guy Verville, Emanuele Vicentini, Jörg Vogt, Jürgen Vollmer, M J Williams, and David Wilson.

If I have inadvertently left anyone off the list I apologise, and please let me know so that I can correct the omisssion.[3] Along those lines, if you have any questions you may post them on `http://tex.stackexchange.com` as you are likely to get a satisfactory and timely response.

Of course, none of this would have been possible without Donald Knuth's TeX system and the subsequent development of LaTeX by Leslie Lamport.

[3] Please write the maintainer at `daleif at math dot au dot dk`

Remarks to the user

The memoir class gives you many ways to change the appearance of your document, and also provides some ready-made styles that might be appropriate for your purposes.

As you can see, this manual is not slim and attempts to describe in some detail how the various aspects of memoir work and gives examples of how you can change these to better match your needs. However, there are a myriad of different things that users might wish to do and it is not possible either for the class to provide ready made simple, or even complex, methods to directly support these, or for this manual to give examples of how everything might be accomplished.

If many want a particular facility that is not available, then it may be possible to add that. If it is only one who wishes it then, unless the one is the author, it is unlikely to be provided. But don't let this stop you from asking, especially if you can provide the necessary code.

The complete documented code for the class is available in the file `memoir.dtx`. If you want to know how something is done then you can read the code for *all* the details. If you want to do something different, then the code is there for you to look at and experiment with. You should, though, not change any of the code in the class. If you need to do so, then copy the code you wish to change into the document's preamble or a package of your own or a class of your own (with a different name) and make the changes there. Do not expect any help if you change the memoir class code directly.

As the years go by support for memoir will devolve from one person to another.[4] Therefore it is probably safer to ask questions, complain, make suggestions, etc., on a Q&A site like `http://tex.stackexchange.com`, which is archived and read by many, than correspond directly with the maintainer, who might well be away for some considerable time and perhaps not notice your email after having returned to base.

In either case please include the word `memoir` in the subject, and if possible, please *tag* the question with the `memoir` tag. That will help the memoir maintainer keep track of memoir related matters.

From the maintainer: Most memoir related questions should go to `http://tex.stackexchange.com`, please remember to tag them properly, that really helps locating the memoir related questions. If no-one comes up with an answer, you can also write me directly via `daleif (at) math dot au dot dk`.

[4] Lars Madsen took over from Peter Wilson in 2009.

Terminology

Like all professions and trades, typographers and printers have their specialised vocabulary.

First there is the question of pages, leaves and sheets. The trimmed sheets of paper that make up a book are called *leaves*, and I will call the untrimmed sheets the *stock* material. A leaf has two sides, and a *page* is one side of a leaf. If you think of a book being opened flat, then you can see two leaves. The front of the righthand leaf, is called the *recto* page of that leaf, and the side of the lefthand leaf that you see is called the *verso* page of that leaf. So, a leaf has a recto and a verso page. Recto pages are the odd-numbered pages and verso pages are even-numbered.

Then there is the question of folios. The typographical term for the number of a page is *folio*. This is not to be confused with the same term as used in 'Shakespeare's First Folio' where the reference is to the height and width of the book, nor to its use in the phrase '*folio* signature' where the term refers to the number of times a printed sheet is folded. Not every page in a book has a printed folio, and there may be pages that do not have a folio at all. Pages with folios, whether printed or not, form the *pagination* of the book. Pages that are not counted in the pagination have no folios.

I have not been able to find what I think is a good definition for 'type' as it seems to be used in different contexts with different meanings. It appears to be a kind of generic word; for instance there are type designers, type cutters, type setters, type foundries,... For my purposes I propose that *type* is one or more printable characters (or variations or extensions to this idea). Printers use the term *sort* to refer to one piece of lead type.

A *typeface* is a set of one or more fonts, in one or more sizes, designed as a stylistic whole.

A *font* is a set of characters. In the days of metal type and hot lead a font meant a complete alphabet and auxiliary characters in a given size. More recently it is taken to mean a complete set of characters regardless of size. A font of roman type normally consists of CAPITAL LETTERS, SMALL CAPITALS, lowercase letters, numbers, punctuation marks, ligatures (such as 'fi' and 'ffi'), and a few special symbols like &.

A *font family* is a set of fonts designed to work harmoniously together, such as a pair of roman and italic fonts.

The size of a font is expressed in points (72.27 points equals 1 inch equals 25.4 millimeters). The size is a rough indication of the height of the tallest character, but different fonts with the same size may have very different actual heights. Traditionally font sizes were referred to by names (see Table 1) but nowadays just the number of points is used.

The typographers' and printers' term for the vertical space between the lines of normal text is *leading*, which is also usually expressed in points and is usually larger than the font

Table 1: Traditional font size designations

Points	Name	Points	Name
3	Excelsior	11	Small Pica
$3\frac{1}{2}$	Brilliant	12	Pica
4	Diamond	14	English
5	Pearl	18	Great Primer
$5\frac{1}{2}$	Agate	24	Double (or Two Line) Pica
6	Nonpareil	28	Double (or Two Line) English
$6\frac{1}{2}$	Mignonette	36	Double (or Two Line) Great Primer
7	Minion	48	French Canon (or Four Line Pica)
8	Brevier	60	Five Line Pica
9	Bourgeois	72	Six line Pica
10	Long Primer	96	Eight Line Pica

Table 2: Printers units

Name (abbreviation)	Value
point (pt)	
pica (pc)	1pc = 12pt
inch (in)	1in = 72.27pt
centimetre (cm)	2.54cm = 1in
millimetre (mm)	10mm = 1cm
big point (bp)	72bp = 72.27pt
didot point (dd)	1157dd = 1238pt
cicero (cc)	1cc = 12dd

size. A convention for describing the font and leading is to give the font size and leading separated by a slash; for instance $10/12$ for a 10pt font set with a 12pt leading, or $12/14$ for a 12pt font set with a 14pt leading.

The normal length of a line of text is often called the *measure* and is normally specified in terms of picas where 1 pica equals 12 points (1pc = 12pt).

Documents may be described as being typeset with a particular font with a particular size and a particular leading on a particular measure; this is normally given in a shorthand form. A 10pt font with 11pt leading on a 20pc measure is described as $10/11 \times 20$, and $14/16 \times 22$ describes a 14pt font with 16pt leading set on a a 22pc measure.

UNITS OF MEASUREMENT

Typographers and printers use a mixed system of units, some of which we met above. The fundamental unit is the point; Table 2 lists the most common units employed.

Points and picas are the traditional printers units used in English-speaking countries. The didot point and cicero are the corresponding units used in continental Europe. In Japan 'kyus' (a quarter of a millimetre) may be used as the unit of measurement. Inches and centimetres are the units that we are all, or should be, familiar with.

The point system was invented by Pierre Fournier le jeune in 1737 with a length of 0.349mm. Later in the same century François-Ambroise Didot introduced his point system with a length of 0.3759mm. This is the value still used in Europe. Much later, in 1886, the American Type Founders Association settled on 0.013837in as the standard size for the point, and the British followed in 1898. Conveniently for those who are not entirely metric in their thinking this means that six picas are approximately equal to one inch.

The big point is somewhat of an anomaly in that it is a recent invention. It tends to be used in page markup languages, like PostScript[5], in order to make calculations quicker and easier.

The above units are all constant in value. There are also some units whose value depends on the particular font being used. The *em* is the nominal height of the current font; it is used as a width measure. An *en* is half an em. The *ex* is nominally the height of the letter 'x' in the current font. You may also come across the term *quad*, often as in a phrase like 'starts with a quad space'. It is a length defined in terms of an em, often a quad is 1em.

[5]PostScript is a registered trademark of Adobe Systems Incorporated.

One

Starting off

As usual, the memoir class is called by `\documentclass[`⟨*options*⟩`]{memoir}`. The ⟨*options*⟩ include being able to select a paper size from among a range of sizes, selecting a type size, selecting the kind of manuscript, and some related specifically to the typesetting of mathematics.

1.1 Stock paper size options

The stock size is the size of a single sheet of the paper you expect to put through the printer. There is a range of stock paper sizes from which to make a selection. These are listed in Table 1.1 through Table 1.3. Also included in the tables are commands that will set the stock size or paper size to the same dimensions.

There are two options that don't really fit into the tables.

ebook for a stock size of 6×9 inches, principally for 'electronic books' intended to be displayed on a computer monitor

landscape to interchange the height and width of the stock.

Table 1.1: Class stock metric paper size options, and commands

Option	Size	stock size command	page size command
a7paper	105×74 mm	`\stockavii`	`\pageavii`
a6paper	148×105 mm	`\stockavi`	`\pageavi`
a5paper	210×148 mm	`\stockav`	`\pageav`
a4paper	297×210 mm	`\stockaiv`	`\pageaiv`
a3paper	420×297 mm	`\stockaiii`	`\pageaiii`
b7paper	125×88 mm	`\stockbvii`	`\pagebvii`
b6paper	176×125 mm	`\stockbvi`	`\pagebvi`
b5paper	250×176 mm	`\stockbv`	`\pagebv`
b4paper	353×250 mm	`\stockbiv`	`\pagebiv`
b3paper	500×353 mm	`\stockbiii`	`\pagebiii`
mcrownvopaper	186×123 mm	`\stockmetriccrownvo`	`\pagemetriccrownvo`
mlargecrownvopaper	198×129 mm	`\stockmlargecrownvo`	`\pagemlargecrownvo`
mdemyvopaper	216×138 mm	`\stockmdemyvo`	`\pagemdemyvo`
msmallroyalvopaper	234×156 mm	`\stockmsmallroyalvo`	`\pagemsmallroyalvo`

Table 1.2: Class stock US paper size options, and commands

Option	Size	stock size command	page size command
dbillpaper	7×3 in	\stockdbill	\pagedbill
statementpaper	8.5×5.5 in	\stockstatement	\pagestatement
executivepaper	10.5×7.25 in	\stockexecutive	\pageexecutive
letterpaper	11×8.5 in	\stockletter	\pageletter
oldpaper	12×9 in	\stockold	\pageold
legalpaper	14×8.5 in	\stocklegal	\pagelegal
ledgerpaper	17×11 in	\stockledger	\pageledger
broadsheetpaper	22×17 in	\stockbroadsheet	\pagebroadsheet

Table 1.3: Class stock British paper size options, and commands

Option	Size	stock size command	page size command
pottvopaper	6.25×4 in	\stockpottvo	\pagepottvo
foolscapvopaper	6.75×4.25 in	\stockfoolscapvo	\pagefoolscapvo
crownvopaper	7.5×5 in	\stockcrownvo	\pagecrownvo
postvopaper	8×5 in	\stockpostvo	\pagepostvo
largecrownvopaper	8×5.25 in	\stocklargecrownvo	\pagelargecrownvo
largepostvopaper	8.25×5.25 in	\stocklargepostvo	\pagelargepostvo
smalldemyvopaper	8.5×5.675 in	\stocksmalldemyvo	\pagesmalldemyvo
demyvopaper	8.75×5.675 in	\stockdemyvo	\pagedemyvo
mediumvopaper	9×5.75 in	\stockmediumvo	\pagemediumvo
smallroyalvopaper	9.25×6.175 in	\stocksmallroyalvo	\pagesmallroyalvo
royalvopaper	10×6.25 in	\stockroyalvo	\pageroyalvo
superroyalvopaper	10.25×6.75 in	\stocksuperroyalvo	\pagesuperroyalvo
imperialvopaper	11×7.5 in	\stockimperialvo	\pageimperialvo

All the options, except for landscape, are mutually exclusive. The default stock paper size is letterpaper.

If you want to use a stock size that is not listed there are methods for doing this, which will be described later.

1.2 TYPE SIZE OPTIONS

The type size option sets the default font size throughout the document. The class offers a wider range of type sizes than usual. These are:

9pt for 9pt as the normal type size

10pt for 10pt as the normal type size

11pt for 11pt as the normal type size

12pt for 12pt as the normal type size

14pt for 14pt as the normal type size[1]

17pt for 17pt as the normal type size

20pt for 20pt as the normal type size

25pt for 25pt as the normal type size

30pt for 30pt as the normal type size

36pt for 36pt as the normal type size

48pt for 48pt as the normal type size

60pt for 60pt as the normal type size

*pt for an author-defined size as the normal type size

extrafontsizes Using scalable fonts that can exceed 25pt.

Note that this includes `\huge`*,* `\Huge` *and* `\HUGE` *under 14pt. For 17pt and up, an error is thrown if used without extrafontsizes, no error is given for 14pt, there sizes above* `\LARGE` *will just be unavailable unless extrafontsizes is used.*

These options, except for extrafontsizes, are mutually exclusive. The default type size is 10pt.

Options greater than 17pt or 20pt are of little use unless you are using scalable fonts — the regular Computer Modern bitmap fonts only go up to 25pt. The option extrafontsizes indicates that you will be using scalable fonts that can exceed 25pt. By default this option makes Latin Modern in the T1 encoding as the default font (normally Computer Modern in the OT1 encoding is the default).

1.2.1 Extended font sizes

By default, if you use the extrafontsizes option the default font for the document is Latin Modern in the T1 font encoding. This is like putting

```
\usepackage{lmodern}\usepackage[T1]{fontenc}
```

in the documents's preamble (but with the extrafontsizes option you need not do this).

```
\newcommand*{\memfontfamily}{⟨fontfamily⟩}
\newcommand*{\memfontenc}{⟨fontencoding⟩}
\newcommand*{\memfontpack}{⟨package⟩}
```

Internally the class uses `\memfontfamily` and `\memfontenc` as specifying the new font and encoding, and uses `\memfontpack` as the name of the package to be used to implement the font. The internal definitions are:

```
\providecommand*{\memfontfamily}{lmr}
\providecommand*{\memfontenc}{T1}
\providecommand*{\memfontpack}{lmodern}
```

which result in the `lmr` font (Latin Modern) in the T1 encoding as the default font, which is implemented by the lmodern package. If you want a different default, say New Century Schoolbook (which comes in the T1 encoding), then

[1] Note that for 14pt, `\huge`, `\Huge` and `\HUGE` will be the same as `\LARGE`, unless the extrafontsizes option is also is activated.

```
\newcommand*{\memfontfamily}{pnc}
\newcommand*{\memfontpack}{newcent}
\documentclass[...]{memoir}
```

will do the trick, where the `\newcommand*`s are put *before* the `\documentclass` declaration (they will then override the `\provide...` definitions within the class code).

If you use the *pt option then you have to supply a `clo` file containing all the size and space specifications for your chosen font size, and also tell memoir the name of the file. *Before* the `\documentclass` command define two macros, `\anyptfilebase` and `\anyptsize` like:

```
\newcommand*{\anyptfilebase}{⟨chars⟩}
\newcommand*{\anyptsize}{⟨num⟩}
```

When it comes time to get the font size and spacing information memoir will try and input a file called `\anyptfilebase\anyptsize.clo` which you should have made available; the `\anyptsize` ⟨*num*⟩ must be an integer.[2] Internally, the class specifies

```
\providecommand*{\anyptfilebase}{mem}
\providecommand*{\anyptsize}{10}
```

which names the default as `mem10.clo`, which is for a 10pt font. If, for example, you have an 18pt font you want to use, then

```
\newcommand*{\anyptfilebase}{myfont}
\newcommand*{\anyptsize}{18}
\documentclass[...*pt...]{memoir}
```

will cause LaTeX to try and input the `myfont18.clo` file that you should have provided. Use one of the supplied `clo` files, such as `mem10.clo` or `mem60.clo` as an example of what must be specified in your `clo` file.

1.3 PRINTING OPTIONS

This group of options includes:

twoside for when the document will be published with printing on both sides of the paper.

oneside for when the document will be published with only one side of each sheet being printed on.
The twoside and oneside options are mutually exclusive.

onecolumn only one column of text on a page.

twocolumn two equal width columns of text on a page.
The onecolumn and twocolumn options are mutually exclusive.

openright each chapter will start on a recto page.

openleft each chapter will start on a verso page.

openany a chapter may start on either a recto or verso page.
The openright, openleft and openany options are mutually exclusive.

[2] If it is not an integer then TeX could get confused as to the name of the file — it normally expects there to be only one period (.) in the name of a file.

final for camera-ready copy of your labours.

draft this marks overfull lines with black bars and enables some change marking to be shown. There may be other effects as well, particularly if some packages are used.

ms this tries to make the document look as though it was prepared on a typewriter. Some publishers prefer to receive poor looking submissions.

The final, draft and ms options are mutually exclusive.

showtrims this option prints marks at the corners of the sheet so that you can see where the stock must be trimmed to produce the final page size.

The defaults among the printing options are twoside, onecolumn, openright, and final.

1.4 OTHER OPTIONS

The remaining options are:

leqno equations will be numbered at the left (the default is to number them at the right).

fleqn displayed math environments will be indented an amount `\mathindent` from the left margin (the default is to center the environments).

openbib each part of a bibliography entry will start on a new line, with second and succeding lines indented by `\bibindent` (the default is for an entry to run continuously with no indentations).

article typesetting *simulates* the article class, but the `\chapter` command is not disabled, basically `\chapter` will behave as if it was `\section`. Chapters do not start a new page and chapter headings are typeset like a section heading. The numbering of figures, etc., is continuous and not per chapter. However, a `\part` command still puts its heading on a page by itself.

oldfontcommands makes the old, deprecated LaTeX version 2.09 font commands available. Warning messages will be produced whenever an old font command is encountered.

fullptlayout disable point trunction of certain layout lengths, for example `\textwidth`. The default is to round these of to a whole number of points, this option disables this feature.

None of these options are defaulted.

1.5 REMARKS

Calling the class with no options is equivalent to:

```
\documentclass[letterpaper,10pt,twoside,onecolumn,openright,final]{memoir}
```

The source file for this manual starts

```
\documentclass[letterpaper,10pt,extrafontsizes]{memoir}
```

which is overkill as both letterpaper and 10pt are among the default options.

Actual typesetting only occurs within the `document` environment. The region of the file between the `\documentclass` command and the start of the `document` environment is called the *preamble*. This is where you ask for external packages and define your own macros if you feel so inclined.

`\flushbottom` `\raggedbottom`

When the twoside or twocolumn option is selected then typesetting is done with `\flushbottom`, otherwise it is done with `\raggedbottom`.

When `\raggedbottom` is in effect LaTeX makes little attempt to keep a constant height for the typeblock; pages may run short.

When `\flushbottom` is in effect LaTeX ensures that the typeblock on each page is a constant height, except when a page break is deliberately introduced when the page might run short. In order to maintain a constant height it may stretch or shrink some vertical spaces (e.g., between paragraphs, around headings or around floats or other inserts like displayed maths). This may have a deleterious effect on the color of some pages.

If you get too many strung out pages with `\flushbottom` you may want to put `\raggedbottom` in the preamble.

If you use the ebook option you may well also want to use the 12pt and oneside options.

Two

Laying out the page

Up until this chapter the *headings* pagestyle has been used; pagestyles are described in §7.2. This, and later chapters, are typeset with the *ruled* pagestyle.

2.1 Introduction

The class provides a default page layout, in which the page size is the same as the stock size and the typeblock is roughly in the middle of the page. This chapter describes the commands provided by the class to help you produce your own page layout if the default is inappropriate.

If you are happy with the default layout you may skip the rest of this chapter.

The pages of a book carry the text which is intended to educate, entertain and/or amuse the reader. The page must be designed to serve the purposes of the author and to ease the reader's task in assimilating the author's ideas. A good page design is one which the general reader does not notice. If the reader is constantly noticing the page layout, even unconsciously, it distracts from the purpose of the book. It is not the job of the designer to shout, or even to murmur, 'look at my work'.

There are three main parts to a page: the page itself, the typeblock, and the margins separating the typeblock from the edges of the page. Of slightly lesser importance are the running headers and footers, and possibly marginal notes. The art of page design is obtaining a harmonious balance or rhythm between all these.

Although the form is different, the facilities described in this chapter are similar to those provided by the geometry package [Ume99].

> *Note, if your paper choice matches one of the class paper options, then you can skip forward to section 2.4 – The typeblock, as you have already chosen your stock size and does not need trimming.*

On the other hand, if, say, you are designing a document, that is to be printed on one type of paper (the stock), and then trimmed to another, please read on.

2.2 Stock material

Printing is the act of laying symbols onto a piece of stock material. Some print on T-shirts by a process called silk screening, where the shapes of the symbols are made in a screen and then fluid is squeezed through the screen onto the stock material — in this case the fabric of the T shirt. Whether or not this is of general interest it is not the sort of printing

or stock material that is normally used in book production. Books, except for the very particular, are printed on paper.

In the desktop publishing world the stock paper is usually one from a range of standard sizes. In the USA it is typically letterpaper (11 by 8.5 inches) and in the rest of the world A4 paper (297 by 210 mm), with one page per piece of stock. In commercial printing the stock material is much larger with several pages being printed on each stock piece; the stock is then folded, cut and trimmed to form the final pages for binding. The class assumes that desktop publishing is the norm.

2.3 THE PAGE

The class assumes that there will be only a single page on a side of each piece of stock; two sides means that there can be two pages, one on the front and the other on the back.

The parameters used by LaTeX itself to define the page layout are illustrated in Figure 2.1. LaTeX does not actually care about the physical size of a page — it assumes that, with respect to the top lefthand corner, the sheet of paper to be printed is infinitely wide and infinitely long. If you happen to have a typeblock that is too wide or too long for the sheet, LaTeX will merrily position text outside the physical boundaries.

The LaTeX parameters are often not particularly convenient if, say, the top of the text must be a certain distance below the top of the page and the fore-edge margin must be twice the spine margin. It is obviously possible to calculate the necessary values for the parameters, but it is not a pleasurable task.

The class provides various means of specifying the page layout, which are hopefully more convenient to use than the standard ones. Various adjustable parameters are used that define the stock size, page size, and so on. These differ in some respects from the parameters in the standard classes, although the parameters for marginal notes are the same in both cases. Figure 2.3 shows the main class layout parameters for a recto page. These may be changed individually by `\setlength` or by using the commands described below. Figure 2.2 illustrates the same parameters on a verso page.

The first step in designing the page layout is to decide on the page size and then pick an appropriate stock size. Selecting a standard stock size will be cheaper than having to order specially sized stock material.

`\setstocksize{`⟨*height*⟩`}{`⟨*width*⟩`}`

The class options provide for some common stock sizes. So if you have specified the class option `a4paper` and intend to print on A4, then you have no need for `\setstocksize`, proceed directly to section 2.4. Alternatively, the class provide settings for many other standard stock sizes, see Table 1.1 through Table 1.3.

If you have some other size that you want to use, the command `\setstocksize` can be used to specify that the stock size is ⟨*height*⟩ by ⟨*width*⟩. For example the following specifies a stock of 9 by 4 inches:

```
\setstocksize{9in}{4in}
```

The size of the page must be no larger than the stock but may be smaller which means that after printing the stock must be trimmed down to the size of the page. The page may be positioned anywhere within the bounds of the stock.

The circle is at 1 inch from the top and left of the page. Dashed lines represent (`\hoffset + 1 inch`) and (`\voffset + 1 inch`) from the top and left of the page.

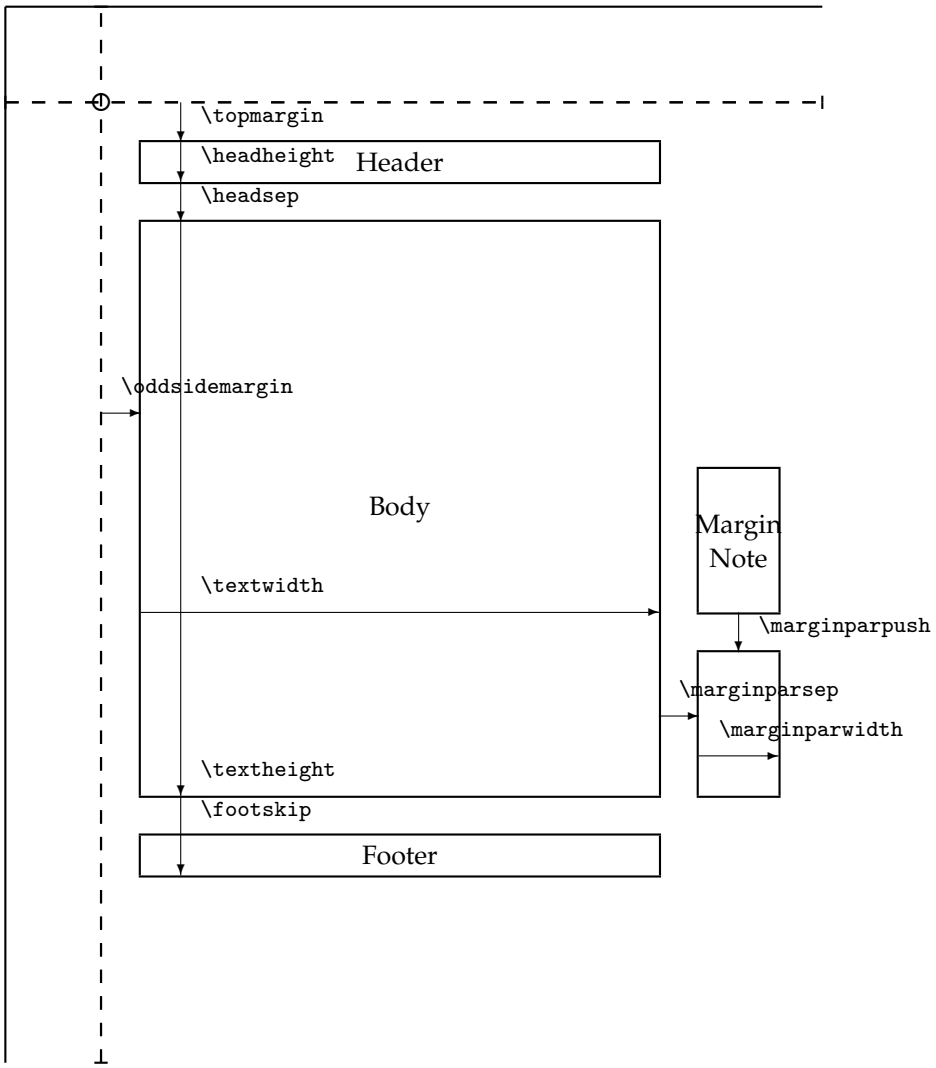


Figure 2.1: LaTeX page layout parameters for a recto page

Dashed lines represent the actual page size after trimming the stock.

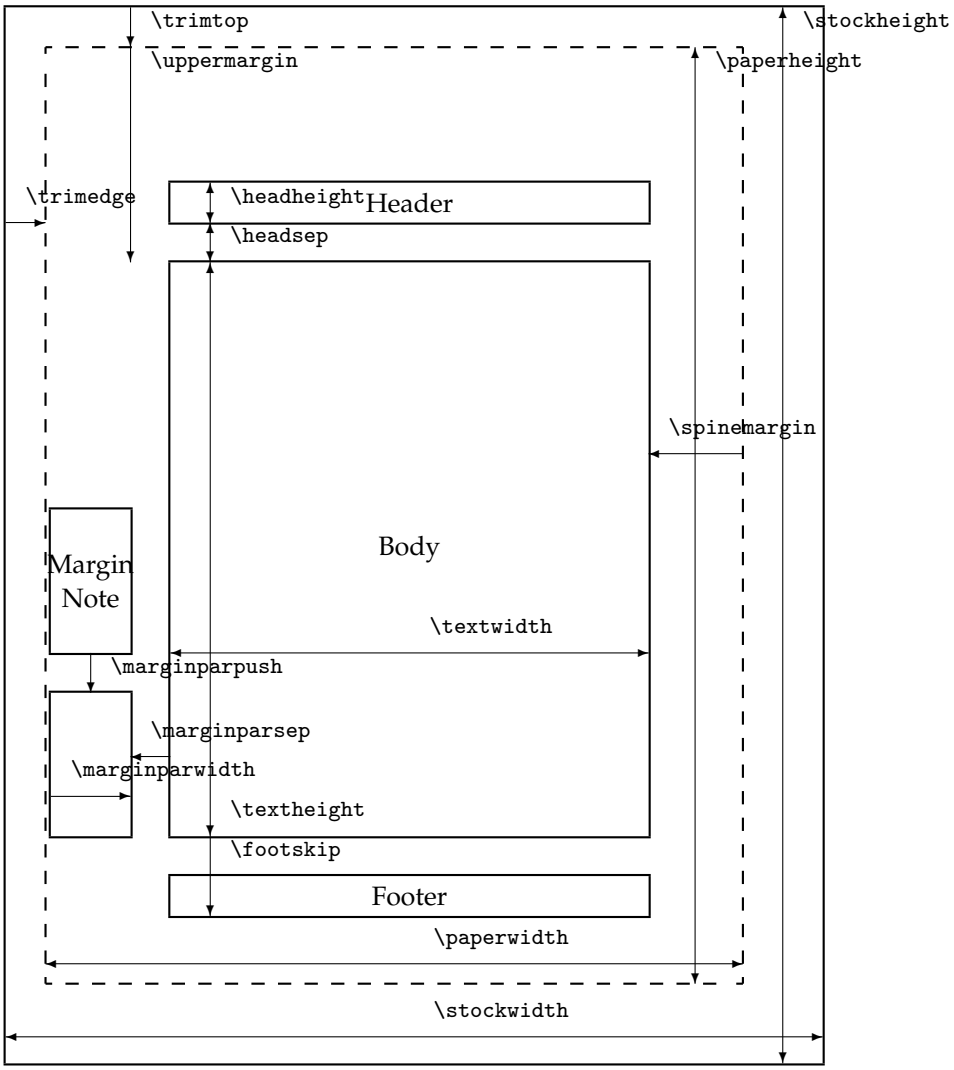


Figure 2.2: The memoir class page layout parameters for a verso page

Dashed lines represent the actual page size after trimming the stock.

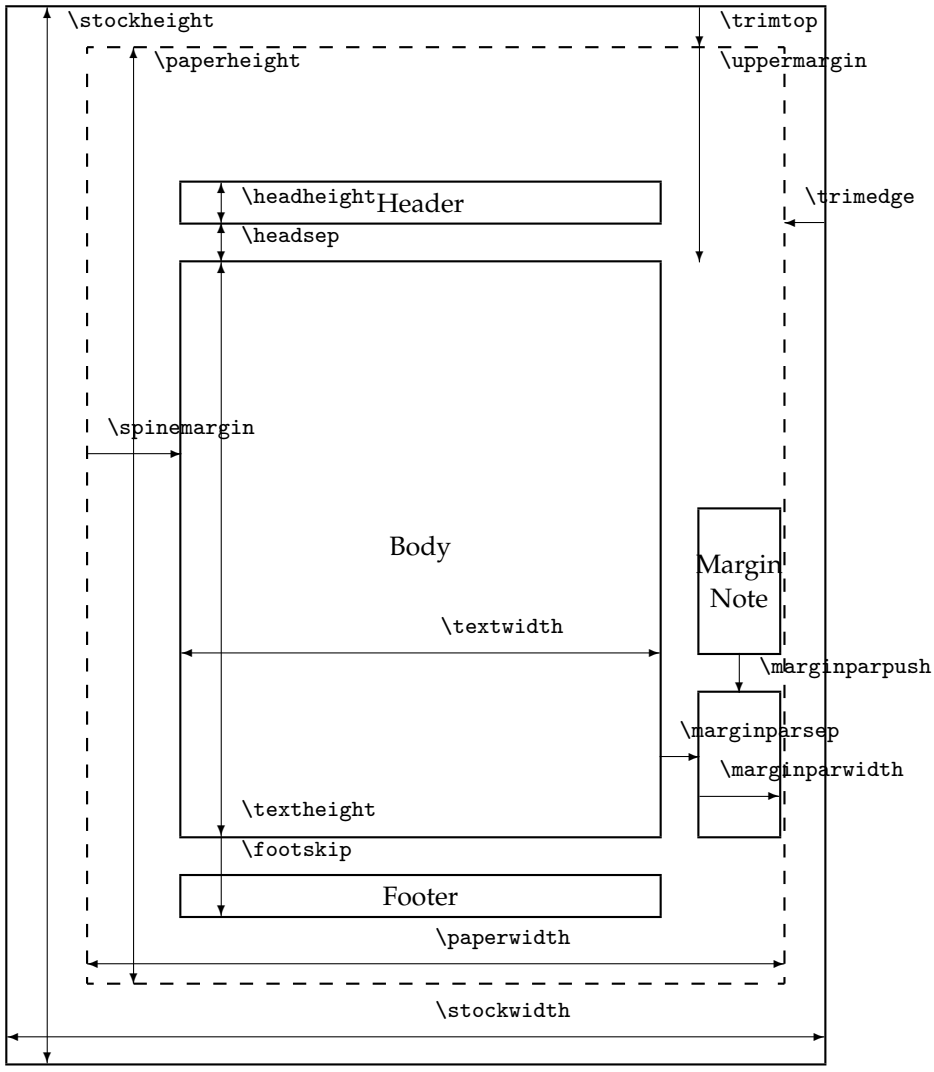


Figure 2.3: The memoir class page layout parameters for a recto page

Table 2.1: Arguments and results for `\settrimmedsize` and `\settypeblocksize`

⟨*height*⟩	⟨*width*⟩	⟨*ratio*⟩	Result
H	W	r	ambiguous
H	W	*	H, W
H	*	r	$W = rH$
H	*	*	ambiguous
*	W	r	$H = rW$
*	W	*	ambiguous
*	*	r	ambiguous
*	*	*	ambiguous

Page layout should be conceived in terms of a double spread; when you open a book in the middle what you see is a double spread — a verso page on the left and a recto page on the right with the spine between them. Most books when closed are taller than they are wide; this makes them easier to hold when open for reading. A squarish page when opened out into a wide spread makes for discomfort unless the book is supported on a table.

`\settrimmedsize{`⟨*height*⟩`}{`⟨*width*⟩`}{`⟨*ratio*⟩`}`

Initially the page size is made the same as the stock size, as set by the paper size option. The command `\settrimmedsize` can be used to specify the height and width of the page (after any trimming). The ⟨*ratio*⟩ argument is the amount by which the ⟨*height*⟩ or the ⟨*width*⟩ must be multiplied by to give the width or the height. Only two out of the three possible arguments must be given values with the other (unvalued) argument given as `*` (an asterisk). The lengths `\paperheight` and `\paperwidth` are calculated according to the given arguments. That is, the command enables the `\paperheight` and `\paperwidth` to be specified directly or as one being in a given ratio to the other. The potential combinations of arguments and the corresponding results are listed in Table 2.1.

If you have used `\setstocksize` to redefine the stock, then to get the same page size, do:

```
\settrimmedsize{\stockheight}{\stockwidth}{*}
```

or for the page dimensions to be 90% of the stock dimensions:

```
\settrimmedsize{0.9\stockheight}{0.9\stockwidth}{*}
```

The following are three different ways of defining an 8 by 5 inch page.

```
\settrimmedsize{8in}{5in}{*}
\settrimmedsize{8in}{*}{0.625}  % 5 = 0.625 times 8
\settrimmedsize{*}{5in}{1.6}    % 8 = 1.6 times 5
```

If you look at a well bound hardback book you can see that the sheets are folded so that they are continuous at the spine, where they are sewn together into the binding. The top of the pages should be smooth so that when the book is upright on a bookshelf dust has a harder job seeping between the pages than if the top was all raggedy. Thus, if the stock is trimmed it will be trimmed at the top. It will also have been cut at the fore-edges of the pages and at the bottom, otherwise the book would be unopenable and unreadable.

`\settrims{`⟨*top*⟩`}{`⟨*foredge*⟩`}`

The command `\settrims` can be used to specify the amount intended to be removed from the top (⟨*top*⟩) and fore-edge (⟨*foredge*⟩) of the stock material to produce the top and fore-edge of a recto page. Note that the combination of `\settrims` and `\settrimmedsize` locate the page with respect to the stock. By default the top and edge trims are zero, which means that if any trimming is required it will be at the spine and bottom edges of the stock unless `\settrims` is used to alter this.

You can either do any trim calculation for youself or let LaTeX do it for you. For example, with an 8in by 5in page on 10in by 7in stock

```
\settrims{2in}{2in}
```

specifies trimming 2in from the top and fore-edge of the stock giving the desired page size. Taking a design where, say, the page is 90% of the stock size it's easy to get LaTeX to do the calculation:

```
\setlength{\trimtop}{\stockheight}    % \trimtop = \stockheight
\addtolength{\trimtop}{-\paperheight}  %          - \paperheight
\setlength{\trimedge}{\stockwidth}    % \trimedge = \stockwidth
\addtolength{\trimedge}{-\paperwidth} %           - \paperwidth
```

which will set all the trimming to be at the top and fore-edge. If you wanted, say, equal trims at the top and bottom you could go on and specify

```
\settrims{0.5\trimtop}{\trimedge}
```

See also section 2.12 – *Place trimmed page on the stock*, where we present some extra commands. For example `\setpagecc` for placing a trimmed page centered on the stock.

After trimming the various methods to specify the typeblock, now refers to the *trimmed* page, not the stock material.

2.4 The typeblock

Like the page, the typeblock is normally rectangular with the height greater than the width. The lines of text must be laid out so that they are easy to read. Common practice, and more recently psychological testing, has shown that long lines of text are difficult to read. Thus, there is a physiological upper limit to the width of the typeblock. From a practical viewpoint, a line should not be too short because then there is difficulty in justifying the text.

2.4.1 A note on the width of the typeblock

Experiments have shown that the number of characters in a line of single column text on a page should be in the range 60 to 70 for ease of reading. The range may be as much as 45 to 75 characters but 66 characters is often considered to be the ideal number. Much shorter and the eye is dashing back and forth between each line. Much longer and it is hard to pick up the start of the next line if the eye has to jump back too far — the same line may be read twice or the following line may be inadvertently jumped over. For double column text the ideal number of characters is around 45, give or take 5 or so.

Bringhurst [Bri99] gives a method for determining the number of characters in a line for any font: measure the length of the lowercase alphabet and use a copyfitting table that

Table 2.2: Average characters per line

Pts.	Line length in picas							
	10	14	18	22	26	30	35	40
80	*40*	**56**	**72**	88	104			
85	*38*	*53*	**68**	83	98	113		
90	*36*	*50*	**64**	79	86	107		
95	34	*48*	**62**	75	89	103		
100	33	*46*	**59**	**73**	86	99	116	
105	32	*44*	57	**70**	82	95	111	
110	30	*43*	55	**67**	79	92	107	
115	29	*41*	53	**64**	76	88	103	
120	28	*39*	*50*	**62**	73	84	98	112
125	27	38	*48*	**59**	**70**	81	94	108
130	26	36	*47*	57	**67**	78	91	104
135	25	35	*45*	55	**65**	75	88	100
140	24	34	*44*	53	**63**	73	85	97
145	23	33	*42*	51	**61**	**70**	82	94
150	23	32	*41*	*51*	**60**	**69**	81	92
155	22	31	*39*	*49*	58	**67**	79	90
160	22	30	39	*48*	56	**65**	76	87
165	21	30	38	*46*	55	**63**	74	84
170	21	29	37	*45*	53	**62**	72	82
175	20	28	36	*44*	52	**60**	**70**	80
180	20	27	35	*43*	51	59	**68**	78
185	19	27	34	*42*	*49*	57	**67**	76
190	19	26	33	41	*48*	56	**65**	74
195	18	25	32	40	*47*	54	**63**	72
200	18	25	32	39	*46*	53	**62**	**70**
220	16	22	29	35	*41*	*48*	56	**64**
240	15	20	26	32	38	*44*	51	58
260	14	19	24	30	35	41	*48*	54
280	13	18	23	28	33	38	*44*	50
300	12	17	21	26	31	35	41	*47*
320	11	16	20	25	29	34	39	*45*
340	10	15	19	23	27	32	37	42

shows for a given alphabet length and line length, the average number of characters in that line. Table 2.2 is an abridged version of Bringhurt's copyfitting table. For example, it suggests that a font with a length of 130pt should be set on a measure of about 26pc for a single column or in an 18pc wide column if there are multiple columns.

Morten Høgholm has done some curve fitting to the data. He determined that the expressions

$$L_{65} = 2.042\alpha + 33.41 \tag{2.1}$$

and

$$L_{45} = 1.415\alpha + 23.03 \tag{2.2}$$

fitted aspects of the data, where α is the length of the alphabet in points, and L_i is the suggested width in points, for a line with i characters (remember that 1pc = 12pt).

Table 2.3 gives the lowercase alphabet lengths for some typefaces over a range of font sizes; this may be used in conjunction with Table 2.2 on page 14 when deciding on an appropriate textwidth. I have grouped the listed typefaces into roman, sans-serif, and monospaced, and they are all available in a standard LaTeX system. The Computer Modern Roman, Concrete Roman, Computer Sans, and Typewriter typefaces were all designed by Donald Knuth using Metafont, specifically for use with TeX. The other font families are PostScript outline fonts and can be used in many document publishing systems. These particular fonts are available for use in LaTeX via the packages in the psnfss bundle. Be aware that the Knuthian fonts were designed to form a font family — that is, they were designed to work together and complement each other — while the listed PostScript fonts were designed by different people at different times and for different purposes. Bringhurst [Bri99, p. 96] memorably says 'Baskerville, Helvetica, Palatino and Times Roman, for example — which are four of the most widely available typefaces — are four faces with nothing to offer one another except public disagreement'.

The monospaced fonts, Courier and Typewriter have no place in high quality typesetting except when typesetting computer code or the like, or when trying to fake text written on a real typewriter. Ignoring these, a quick glance at the Table shows that Bookman is a broad font while Times is narrow as befits its original design intent for typesetting narrow columns in newspapers. Computer Modern tends towards the narrow end of the range.

`\xlvchars` `\lxvchars`

Based on Table 2.3, the two lengths `\xlvchars` and `\lxvchars` are initially set to approximately the lengths of a line of text with 45 or 65 characters, respectively, for Computer Modern Roman in the type size selected for the document.

If you are using a different font or size you can use something like the following to calculate and print out the length for you.

```
\newlength{\mylen}                % a length
\newcommand{\alphabet}{abc...xyz} % the lowercase alphabet
\begingroup                       % keep font change local
% font specification e.g., \Large\sffamily
\settowidth{\mylen}{\alphabet}
The length of this alphabet is \the\mylen. % print in document
\typeout{The length of the Large sans alphabet
         is \the\mylen}                    % put in log file
\endgroup                         % end the grouping
```

The `\typeout` macro prints its argument to the terminal and the log file. There is, however, an easier method.

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

The macros `\setxlvchars` and `\setlxvchars`, which were suggested by Morten Høgholm, set the lengths `\xlvchars` and `\lxvchars` respecively for the font ⟨*fontspec*⟩.

Table 2.3: Lowercase alphabet lengths, in points, for various fonts

	8pt	9pt	10pt	11pt	12pt	14pt	17pt	20pt
Bookman	113	127	142	155	170	204	245	294
Charter	102	115	127	139	152	184	221	264
Computer Modern	108	118	127	139	149	180	202	242
Concrete Roman	109	119	128	140	154	185	222	266
New Century Schoolbook	108	122	136	149	162	194	234	281
Palatino	107	120	133	146	160	192	230	276
Times Roman	96	108	120	131	143	172	206	247
Utopia	107	120	134	146	161	193	232	277
Avant Garde Gothic	113	127	142	155	169	203	243	293
Computer Sans	102	110	120	131	140	168	193	233
Helvetica	102	114	127	139	152	184	220	264
Courier	125	140	156	170	187	224	270	324
Typewriter	110	122	137	149	161	192	232	277

The default for ⟨*fontspec*⟩ is `\normalfont`. For example, the values of `\lxvchars` and `\xlvchars` after calling:

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

are: `\lxvchars` = 305.42249pt, and `\xlvchars` = 190.52705pt.

Morten Høgholm[1] also commented:

> ...I was defining some environments that had to have `\parindent` as their indentation. For some reason I just wrote `1.5em` instead of `\parindent` because I 'knew' that was the value. What I had overlooked was that I had loaded the mathpazo package [Pug02], thus, among other things, altering `\parindent`. Conclusion: the environment would use 1.5em = 18.0pt, whereas the `\parindent` was only 17.607pt.
>
> This, and other related situations can be avoided if one places
>
> `\RequirePackage{`⟨*font-package*⟩`}\normalfont`
>
> *before* `\documentclass`.

Note that, in general, it is inadviseable to put any commands before `\documentclass`.

2.4.2 Specifying the typeblock size

The height of the typeblock should be equivalent to an integral number of lines. The class provides two different methods to specify the typeblock. `\settypeblocksize` (below) will set the size of the typeblock, other commands will then be used to place it on the paper. Alternatively the typeblock size can be determined by setting the margins around it. We present this later on in this section (see `\setlrmarginsandblock` and `\setulmarginsandblock`).

`\settypeblocksize{`⟨*height*⟩`}{`⟨*width*⟩`}{`⟨*ratio*⟩`}`

[1] Private communication

The command \settypeblocksize is similar to \settrimmedsize except that it sets the \textheight and \textwidth for the typeblock. The potential combinations of arguments and the corresponding results are listed in Table 2.1 on page 12. For instance, here are three ways of specifying a 6in by 3in typeblock:

```
\settypeblocksize{6in}{3in}{*}
\settypeblocksize{6in}{*}{0.5}
\settypeblocksize{*}{3in}{2}
```

The typeblock has to be located on the page. There is a relationship between the page, typeblock and margins. The sum of the spine, or inner, margin, the fore-edge, or outer, margin and the width of the typeblock must equal the width of the page. Similarly the sum of the upper margin, the lower margin and the height of the typeblock must equal the height of the page. The process of locating the typeblock with respect to the page can be viewed either as positioning the typeblock with respect to the edges of the page or as setting the margins between the page and the typeblock.

Remembering that the page layout should be defined in terms of the appearance as a spread, the spine margin is normally half the fore-edge margin, so that the white space is equally distributed around the sides of the text.

Note. *One will often find that using* \settypeblocksize *without subsequent use of* \setlrmargins *and* \setulmargins *will result in errors as the relationships mentioned above are not met (the* \textwidth *has changed, but not the margins).*

We may add auto adjustments to a future version of memoir.

There is more latitude in choosing the proportions of the upper and lower margins, though usually the upper margin is less than the lower margin so the typeblock is not vertically centered.

Two methods are provided for setting the horizontal dimensions on a page. One is where the width of the typeblock is fixed and the margins are adjustable. The other method is where the size of the margins determines the width of the typeblock.

\setlrmargins{⟨*spine*⟩}{⟨*edge*⟩}{⟨*ratio*⟩}

The command \setlrmargins can be used to specify the side margins[2] with the width of the page and the typeblock being fixed.

Not more than one one argument value is required, with any unvalued arguments being denoted by an asterisk. There are several cases to consider and these are tabulated in Table 2.4.

In the Table, S is the calculated spine margin, E is the calculated fore-edge margin, and P_w and B_w are respectively the page and typeblock widths. The \setlrmargins command maintains the relationship

$$S + E = K_w = \text{constant } (= P_w - B_w).$$

The cases marked ambiguous in the Table are where the particular combination of argument values may make it impossible to guarantee the relationship.

Assuming that we have a 3in wide typeblock on a 5in wide page and we want the spine margin to be 0.8in and the fore-edge margin to be 1.2in (i.e., the fore-edge margin

[2]Only the spine margin is noted in Figure 2.3 and 2.2; the fore-edge margin is at the opposite side of the typeblock.

Table 2.4: Arguments and results for `\setlrmargins`

⟨*spine*⟩	⟨*edge*⟩	⟨*ratio*⟩	Result
S	E	r	ambiguous
S	E	*	ambiguous
S	*	r	ambiguous
S	*	*	$E = K_w - S$
*	E	r	ambiguous
*	E	*	$S = K_w - E$
*	*	r	$E + S = K_w$, $E = rS$
*	*	*	$E + S = K_w$, $E = S$

Table 2.5: Arguments and results for `\setlrmarginsandblock`

⟨*spine*⟩	⟨*edge*⟩	⟨*ratio*⟩	Result
S	E	r	S, E
S	E	*	S, E
S	*	r	$E = rS$
S	*	*	$E = S$
*	E	r	$S = rE$
*	E	*	$S = E$
*	*	r	ambiguous
*	*	*	ambiguous

is half as big again as the spine margin) this can be accomplished in three ways (with the `\paperwidth` and `\textwidth` being previously specified and fixed):

```
% specify spine margin
    \setlrmargins{0.8in}{*}{*}
% specify fore-edge margin
    \setlrmargins{*}{1.2in}{*}
% specify fore-edge/spine ratio
    \setlrmargins{*}{*}{1.5}
```

`\setlrmarginsandblock{`⟨*spine*⟩`}{`⟨*edge*⟩`}{`⟨*ratio*⟩`}`

The command `\setlrmarginsandblock` can be used to specify the spine and fore-edge margins, where the page width is fixed and the width of the typeblock depends on the margins. Results for this command are given in Table 2.5. The same notation is used, but in this case `\setlrmarginsandblock` maintains the relationship

$$S + B_w + E = \text{constant } (= P_w).$$

The width of the typeblock is calculated from $B_w = P_w - S - E$.

Assuming that we want a 3in wide typeblock on a 5in wide page and we want the spine margin to be 0.8in and the fore-edge margin to be 1.2in (i.e., the fore-edge margin is half as big again as the spine margin) this can be accomplished in the following ways

(with the \textwidth being calculated from the previously specified \paperwidth and the specified margins):

```
% specify both margins
    \setlrmarginsandblock{0.8in}{1.2in}{*}
% specify spine & fore-edge/spine ratio
    \setlrmarginsandblock{0.8in}{*}{1.5}
% specify fore-edge & spine/fore-edge ratio
    \setlrmarginsandblock{*}{1.2in}{0.667}
```

If we wanted the margins to be both 1in instead then any of the following will do it:

```
% specify both margins
    \setlrmarginsandblock{1in}{1in}{*}
% specify spine & fore-edge/spine ratio
    \setlrmarginsandblock{1in}{*}{1}
% specify spine (fore-edge = spine)
    \setlrmarginsandblock{1in}{*}{*}
% specify fore-edge & spine/fore-edge ratio
    \setlrmarginsandblock{*}{1in}{1}
% specify fore-edge (spine = fore-edge)
    \setlrmarginsandblock{*}{1in}{*}
```

`\setbinding{`⟨*length*⟩`}`

In some cases, for example when doing a Japanese stab binding, it may be desireable to add a small allowance to the spine margin for the binding. You can use the command \setbinding for this purpose. It decreases the effective page width by ⟨*length*⟩ and later this length will be added on to the spine margin, thus restoring the page width to its original value. If you use \setbinding than it must be *after* setting the page width and *before* setting the spine and fore-edge margins.

That completes the methods for specifying the horizontal spacings. There are similar commands for setting the vertical spacings which are described below.

`\setulmargins{`⟨*upper*⟩`}{`⟨*lower*⟩`}{`⟨*ratio*⟩`}`

The command \setulmargins can be used to specify the upper and lower margins[3] where the heights of the page and the typeblock are fixed. This is similar to \setlrmargins. Using a slightly different notation this time, with U being the upper margin, L being the lower margin, and P_h and B_h being the *height* of the page and typeblock, respectively, the results are shown in Table 2.6. The \setulmargins command maintains the relationship

$$U + L = K_h = \text{constant } (= P_h - B_h).$$

Note that in terms of the traditional LaTeX parameters memoir's \uppermargin is (\topmargin + \headheight + \headsep).

`\setulmarginsandblock{`⟨*upper*⟩`}{`⟨*lower*⟩`}{`⟨*ratio*⟩`}`

[3]Only the upper margin is noted in Figure 2.3 and 2.2; the lower margin is the distance between the bottom of the typeblock and the bottom of the page.

Table 2.6: Arguments and results for `\setulmargins`

⟨*upper*⟩	⟨*lower*⟩	⟨*ratio*⟩	Result
U	L	r	ambiguous
U	L	*	ambiguous
U	*	r	ambiguous
U	*	*	$L = K_h - U$
*	L	r	ambiguous
*	L	*	$U = K_h - L$
*	*	r	$L + U = K_h$, $L = rU$
*	*	*	$L + U = K_h$, $L = U$

Table 2.7: Arguments and results for `\setulmarginsandblock`

⟨*upper*⟩	⟨*lower*⟩	⟨*ratio*⟩	Result
U	L	r	U, L
U	L	*	U, L
U	*	r	$L = rU$
U	*	*	$L = U$
*	L	r	$U = rL$
*	L	*	$U = L$
*	*	r	ambiguous
*	*	*	ambiguous

The command `\setulmarginsandblock` can be used to specify the upper and lower margins, where the page height is fixed and the height of the typeblock depends on the margins. Results for this command are given in Table 2.7. The same notation is used, but in this case `\setulmarginsandblock` maintains the relationship

$$U + B_h + L = \text{constant } (P_h).$$

The height of the typeblock is calculated from $B_h = P_h - U - L$.

Note. *Readers may find several folio designs in [Wil18].*

See also Section 2.11 (*Predefined layouts*) for some predefined layouts.

This completes the methods for specifying the layout of the main elements of the page — the page size, the size of the typeblock and the margins surrounding the typeblock.

2.5 HEADERS, FOOTERS AND MARGINAL NOTES

A page may have two additional items, and usually has at least one of these. They are the running header and running footer. If the page has a folio then it is located either in the header or in the footer. The word 'in' is used rather lightly here as the folio may not be actually *in* the header or footer but is always located at some constant relative position. A common position for the folio is towards the fore-edge of the page, either in the header or the footer. This makes it easy to spot when thumbing through the book. It may be placed

Table 2.8: Arguments and results for `\setheaderspaces`

⟨*headdrop*⟩	⟨*headsep*⟩	⟨*ratio*⟩	Result
D	H_s	r	ambiguous
D	H_s	*	ambiguous
D	*	r	ambiguous
D	*	*	$H_s = C_h - D$
*	H_s	r	ambiguous
*	H_s	*	$D = C_h - H_s$
*	*	r	$H_s + D = C_h$, $H_s = rD$
*	*	*	$H_s + D = C_h$, $H_s = D$

at the center of the footer, but unless you want to really annoy the reader do not place it near the spine.

Often a page header contains the current chapter title, with perhaps a section title on the opposite header, as aids to the reader in navigating around the book. Some books put the book title into one of the headers, usually the verso one, but I see little point in that as presumably the reader knows which particular book he is reading, and the space would be better used providing more useful signposts.

`\setheadfoot{`⟨*headheight*⟩`}{`⟨*footskip*⟩`}`

The `\setheadfoot` macro sets the `\headheight` parameter to the value ⟨*headheight*⟩ and the `\footskip` parameter to ⟨*footskip*⟩. It is usual to set the `\headheight` to at least the value of the `\baselineskip` of the normal body font.

`\setheaderspaces{`⟨*headdrop*⟩`}{`⟨*headsep*⟩`}{`⟨*ratio*⟩`}`

The command `\setheaderspaces` is similar to `\setulmargins`. Using the notation U for the upper margin (as before), H_h for the `\headheight`, H_s for the `\headsep` and D for the `\headdrop`, where the `\headdrop` is the distance between the top of the trimmed page and the top of the header[4], then the macro `\setheaderspaces` maintains the relationship

$$D + H_s = C_h = \text{constant } (= U - H_h).$$

The macro `\setheaderspaces` is for specifying the spacing above and below the page header. The possible arguments and results are shown in Table 2.8. Typically, the `\headsep` is of more interest than the `\headdrop`.

Finally, some works have marginal notes. These really come last in the design scheme, providing the margins have been made big enough to accomodate them. Figure 2.2 shows the marginal note parameters on a verso page, and also illustrates that some parameters control different positions on the stock.

`\setmarginnotes{`⟨*separation*⟩`}{`⟨*width*⟩`}{`⟨*push*⟩`}`

The command `\setmarginnotes` sets the parameters for marginal notes. The distance `\marginparsep` between the typeblock and any note is set to ⟨*separation*⟩, the maximum width for a note, `\marginparwidth`, is set to ⟨*width*⟩ and the minimum vertical distance between notes, `\marginparpush`, is set to ⟨*push*⟩.

[4]The head drop is not shown in Figure 2.3 or 2.2.

Note. *As of memoir v3.6k, we have added an auto adjustment feature for* `\marginparwidth`*, such that unless* `\setmarginnotes` *have been used to make a specific choice, the* `\marginparwidth` *is chosen according to the algorithm described in Section G.1.1. The algorithm relies on* `\marginparmargin` *(if used) being set* before `\checkandfixthelayout`*.*

We may add other auto adjusting features to future memoir releases.

2.6 OTHER

`\setcolsepandrule{`⟨*colsep*⟩`}{`⟨*thickness*⟩`}`

For twocolumn text the width of the gutter between the columns must be specified. LaTeX also lets you draw a vertical rule in the middle of the gutter. The macro `\setcolsepandrule` sets the gutter width, `\columnsep`, to ⟨*colsep*⟩ and the thickness of the rule, `\columnseprule`, to ⟨*thickness*⟩. A ⟨*thickness*⟩ of 0pt means that the rule will be invisible. Visible rules usually have a thickness of about 0.4pt. The total width of the twocolumns of text and the gutter equals the width of the typeblock.

`\setfootins{`⟨*length for normal*⟩`}{`⟨*length for minipage*⟩`}`

When footnotes are added to the text block they are added `\skip\footins` below the text. Since this is a skip it usually needs special syntax to change it. Instead we have provided an interface to set it.[5] The default sizes are `\bigskipamount`.

2.7 PUTTING IT TOGETHER

The page layout parameters just discussed are not always the same as those that are expected by LaTeX, or by LaTeX packages. The parameters that LaTeX expects are shown in Figure 2.1. You can either use the class commands for changing the page layout or change the LaTeX parameters directly using either `\setlength` or `\addtolength` applied to the parameter(s) to be modified. If you choose the latter route, those class parameters which differ from the standard LaTeX parameters will *not* be modified.

The general process of setting up your own page layout is along these lines:

- Decide on the required finished page size, and pick a stock size that is at least as large as the page.
- If needed, use `\setstocksize` to set the values of `\stockheight` and `\stockwidth`, followed by `\settrimmedsize` to specify the values of `\paperheight` and `\paperwidth`.

  If you use and print on, say, A4, the `a4paper` class option is enough, no `\setstocksize` needed.
- Decide on the location of the page with respect to the stock. If the page and stock sizes are the same, then call `\settrims{0pt}{0pt}`. If they are not the same size then decide on the appropriate values for `\trimtop` and `\trimedge` to position the page on the stock, and then set these through `\settrims`.

[5] This interface also sets the equivalent lengths used when `\twocolumnfootnotes` and friends are being used.

- Decide on the size of the typeblock and use `\settypeblocksize` to specify the values of `\textheight` and `\textwidth`.
- If you need a binding allowance, now is the time for `\setbinding`.
- Pick the value for the spine margin, and use `\setlrmargins` to set the values for the `\spinemargin` and `\foremargin`.

  An alternative sequence is to use `\setlrmarginsandblock` to set the `\textwidth` for a particular choice of side margins.
- Pick the value for the upper margin and use `\setulmargins` to set the values for the `\uppermargin` and `\lowermargin`.

  An alternative sequence is to use `\setulmarginsandblock` to set the `\textheight` for a particular choice of upper and lower margins.

  The typeblock is now located on the page.
- Pick the values for the `\headheight` and `\footskip` and use `\setheadfoot` to specify these.
- Pick your value for `\headskip` and use `\setheaderspaces` to set this and `\headmargin`.
- If you are going to have any marginal notes, use `\setmarginnotes` to specify the values for `\marginparsep`, `\marginparwidth` and `\marginparpush`.

You can plan and specify your layout in any order that is convenient to you. Each of the page layout commands is independent of the others; also if a value is set at one point, say the `\textwidth`, and is then later changed in some way, only the last of the settings is used as the actual value.

Comparing Figures 2.3 and 2.1 you can see the different layout parameters provided by the class and used by standard LaTeX. For convenience, and because the figures do not show all the parameters, the two sets of parameters are listed in Table 2.9.

Unless you are satisfied with the default page layout, after specifying the layout that you want you have to call the `\checkandfixthelayout` command to finally implement your specification.

`\checkandfixthelayout[`⟨*algorithm*⟩`]`
`\checkthelayout[`⟨*algorithm*⟩`]`
`\fixthelayout`
`\baselineskip` `\topskip`

The `\checkandfixthelayout` macro uses `\checkthelayout` to check the page layout specification you have given, and then calls `\fixthelayout` to finally implement it.

The `\checkthelayout` macro checks the class layout parameters to see whether they have 'sensible' values (e.g., the `\textwidth` is not negative) and, depending on the ⟨*algorithm*⟩ argument, it may modify the `\textheight`. It does not actually implement the layout.

When using `\flushbottom` LaTeX expects that the `\textheight` is such that an integral number of text lines in the body font will fit exactly into the height. If not, then it issues 'underfull vbox' messages. More precisely, if b is the `\baselineskip` and t is the `\topskip`, N is an integer (the number of lines in the typeblock), and T is the `\textheight` then to avoid underfull vboxes the following relationship must hold

$$T = (N - 1)b + t \tag{2.3}$$

Table 2.9: The class and LaTeX page layout parameters

Class	LaTeX
\stockheight	
\trimtop	
\trimedge	
\stockwidth	
\paperheight	\paperheight
\paperwidth	\paperwidth
\textheight	\textheight
\textwidth	\textwidth
\columnsep	\columnsep
\columnseprule	\columnseprule
\spinemargin	
\foremargin	
	\oddsidemargin
	\evensidemargin
\uppermargin	
\headmargin	
	\topmargin
\headheight	\headheight
\headsep	\headsep
\footskip	\footskip
\marginparsep	\marginparsep
\marginparwidth	\marginparwidth
\marginparpush	\marginparpush

By default \checkthelayout ensures that the final \textheight meets this criterion. The optional ⟨*algorithm*⟩ argument lets you control just how it does this. In the following H is your requested value for the \textheight and the other symbols are as before, with T as the adjusted value, and using integer arithmetic.[6] The permissible values for ⟨*algorithm*⟩ are:

`fixed` The \textheight is not altered.

$$T = H \tag{2.4}$$

If you use this option you may find that underfull vboxes are reported for \flushbottom pages.

`classic` This is the default and is the one used by the standard classes.

$$T = b\lfloor H/b \rfloor + t \tag{2.5}$$

The relationship (2.3) is maintained. This algorithm gets as close to H as possible from below.

[6] In this context 'integer arithmetic' means that the result of a division will be rounded down. For example $99/10$ in 'real arithmetic' results in 9.9, whereas with integer arithmetic the result is 9, not 10.

Table 2.10: Results from sample `\textheight` adjustments

	Algorithm			
	fixed	classic	lines	nearest
Requested height	adjusted height in pts, (lines)			
10.0`\baselineskip`	120.0pt, (10)	130pt, (11)	118pt, (10)	118pt, (10)
10.2`\baselineskip`	122.4pt, (10)	130pt, (11)	118pt, (10)	118pt, (10)
10.4`\baselineskip`	124.8pt, (10)	130pt, (11)	118pt, (10)	130pt, (11)
10.6`\baselineskip`	127.2pt, (10)	130pt, (11)	118pt, (10)	130pt, (11)
10.8`\baselineskip`	129.6pt, (10)	130pt, (11)	118pt, (10)	130pt, (11)
11.0`\baselineskip`	132.0pt, (11)	142pt, (12)	130pt, (11)	130pt, (11)

`lines` This is similar to `classic`, but results in a smaller final value.

$$T = b\lfloor (H - b)/b \rfloor + t \tag{2.6}$$

The relationship (2.3) is maintained.

`nearest` The calculated value is the nearest to the given value while still maintaining the relationship (2.3).

$$T = b\lfloor (H - t + b/2)/b \rfloor + t \tag{2.7}$$

In contrast to `classic`, `nearest` will get as close to H as possible even if this means that T ends up being slightly larger than H.

Table 2.10 shows the results from the various `\textheight` adjustment calculations[7] where the `\baselineskip` is 12pt and the `\topskip` is 10pt, which are the normal values for a Computer Modern 10pt font. In all cases the `fixed` algorithm resulted in underfull vboxes. If you know the number of lines that you want, say 42, then requesting

```
%% setting equivalent to \setlength{\textheight}{42\baselineskip}
\checkandfixthelayout[lines]
```

will result in the most appropriate `\textheight`.

If you use the calc package you can use constructs like the following in a page layout specification:

```
\setlength{\textheight}{41\baselineskip + \topskip}
\settypeblocksize{41\baselineskip + \topskip}{33pc}{*}
```

The `\fixthelayout` macro finally implements the layout, making due adjustement for any binding allowance, and calculates the values for all the standard LaTeX layout parameters (such that packages can use these expected values). If you have used the class macros to change the layout in any way, you must call `\checkandfixthelayout` after you have made all the necessary changes. As an aid, the final layout parameter values are displayed on the terminal and written out to the log file.

At the beginning of the document we will automatically add the page/stock sizes etc to the output such that viewers etc reacts accordingly. In incase you need to manually have the class rewrite these sizes to the document, you can use

[7] For comparison the optimum heights from equation 2.3 for 10, 11 and 12 lines are respectively 118pt, 130pt and 142pt.

```
\refixpagelayout
```

Please note that this macro (and its name), may change in the future.[8]

```
\typeoutlayout
\typeoutstandardlayout
\settypeoutlayoutunit{⟨unit⟩}
```

`\typeoutlayout` writes the current class layout parameter values to the terminal and the log file. It is called by `\checkandfixthelayout` but you can use it yourself at any time. The macro `\typeoutstandardlayout` writes the standard layout parameter values to the terminal and log file so that you can compare the two sets of parameter values.

By using the macro `\settypeoutlayoutunit`, the user can change the unit in which the layout list is shown. Very handy when designing in, say, centimeters. Supported units are `pt`, `pc`, `mm`, `cm`, `in`, `bp`, `dd` and `cc`, default being `pt`, see Table 2 for more information about the units.

2.8 SIDE MARGINS

In twoside printing the spine margin is normally the same on both recto and verso pages and, unless the spine and fore-edge margins are the same, the typeblock is shifted side to side when printing the recto and verso pages. Additionally you can have different headers and footers for the recto and verso pages. However, in oneside printing the typeblock is not moved and the headers and footers are the same for both odd and even pages.

Some documents are designed to have, say, a very wide righthand margin in which to put illustrations; this leads to needing the spine margin on verso pages to be much larger than the spine margin on recto pages. This can be done with the oneside option. However, different headers and footers are required for the recto and verso pages, which can only be done with the twoside option. The way to get the desired effects is like this (twoside is the default class option):

```
\documentclass{memoir}
%%% set up the recto page layout
\checkandfixthelayout%     or perhaps \checkandfixthelayout[lines]
\setlength{\evensidemargin}{\oddsidemargin}% after \checkandfix...
...
```

2.9 EMITTING THE PAGE SIZE

At the start of the document the class will automatically emit the chosen stocksize to the output format (DVI or PDF).[9]

[8] In a comming version, in later in 2020, we will hopefully be able to support changing the stock and layout mid document, like what is provided by the geometry.

[9] In earlier versions we had macros `\fixpdflayout` and `\fixdvipslayout` that held some of the code needed to emit this data. With the newer engines like LuaLaTeX, this became unfeasible thus they we discontinued (their use now emit a warning).

2.10 EXAMPLE

Suppose you want a page that will fit on both A4 and US letterpaper stock, wanting to do the least amount of trimming. The layout requirements are as follows.

- The width of the typeblock should be such that there are the optimum number of characters per line.
- The ratio of the height to the width of the typeblock should equal the golden section.
- The text has to start 50pt below the top of the page. We will use the default `\headheight` and `\footskip`.
- The ratio of the outer margin to the inner margin should equal the golden section, as should the space above and below the header.
- There is no interest at all in marginal notes, so we can ignore any settings for these.

We can either do the maths ourselves or get LaTeX to do it for us. Let's use LaTeX. First we will work out the size of the largest sheet that can be cut from A4 and letterpaper, whose sizes are 297×210 mm and 11×8.5 in; A4 is taller and narrower than letterpaper.

```
\settrimmedsize{11in}{210mm}{*}
```

The stocksize is defined by the class option, which could be either letterpaper or a4paper, but we have to work out the trims to reduce the stock to the page. To make life easier, we will only trim the fore-edge and the bottom of the stock, so the `\trimtop` is zero. The `\trimtop` and `\trimedge` are easily specified by

```
\setlength{\trimtop}{0pt}
\setlength{\trimedge}{\stockwidth}
\addtolength{\trimedge}{-\paperwidth}
```

Or if you are using the calc package, perhaps:

```
\settrims{0pt}{\stockwidth - \paperwidth}
```

Specification of the size of the typeblock is also easy

```
\settypeblocksize{*}{\lxvchars}{1.618}
```

and now the upper and lower margins are specified by

```
\setulmargins{50pt}{*}{*}
```

The spine and fore-edge margins are specified just by the value of the golden section, via

```
\setlrmargins{*}{*}{1.618}
```

The only remaining calculation to be done is the `\headmargin` and `\headsep`. Again this just involves using a ratio

```
\setheaderspaces{*}{*}{1.618}
```

To finish off we have to make sure that the layout is changed

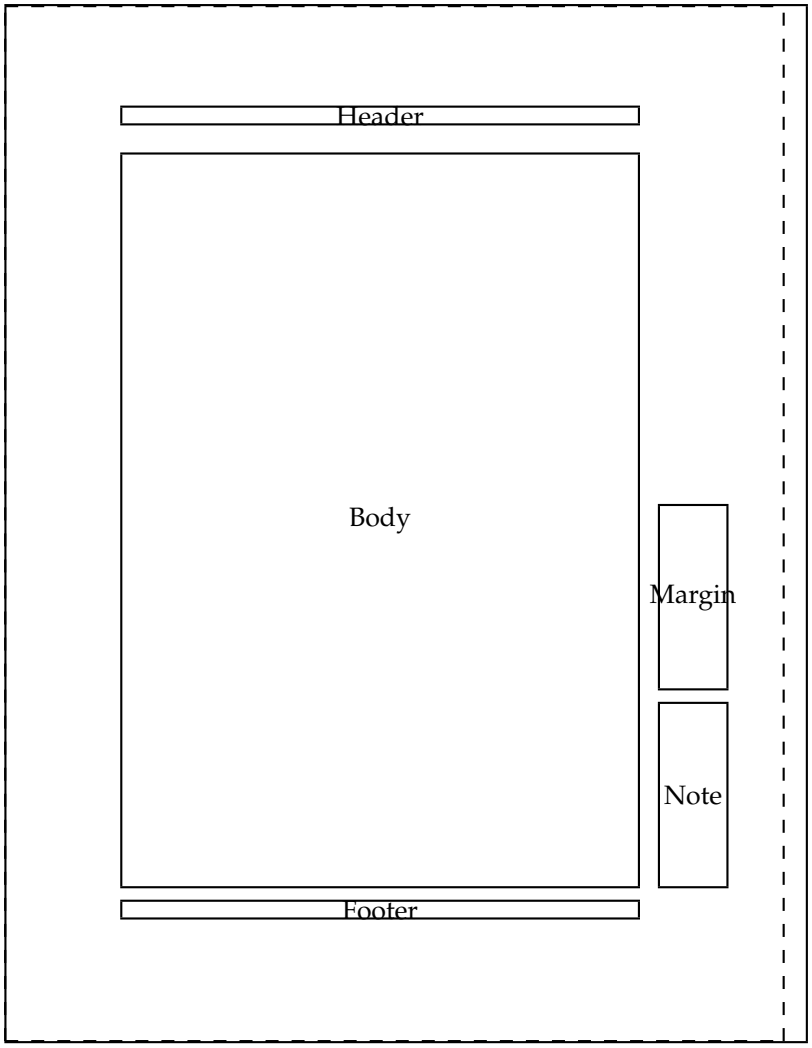
```
\checkandfixthelayout
```

2.10.1 The page layout of this manual

The page layout for this manual is defined in the preamble as:

```
\settrimmedsize{11in}{210mm}{*}
\setlength{\trimtop}{0pt}
```

Dashed lines represent the actual page size after trimming the stock.



Lengths are to the nearest pt.

`\stockheight = 795pt`	`\stockwidth = 614pt`
`\pageheight = 795pt`	`\pagewidth = 598pt`
`\textheight = 562pt`	`\textwidth = 396pt`
`\trimtop = 0pt`	`\trimedge = 17pt`
`\uppermargin = 114pt`	`\spinemargin = 90pt`
`\headheight = 12pt`	`\headsep = 24pt`
`\footskip = 24pt`	`\marginparsep = 17pt`
`\marginparpush = 12pt`	`\columnsep = 10pt`
`\columnseprule = 0.0pt`	

Figure 2.4: The recto page layout for this manual

```
\setlength{\trimedge}{\stockwidth}
\addtolength{\trimedge}{-\paperwidth}
\settypeblocksize{7.75in}{33pc}{*}
\setulmargins{4cm}{*}{*}
\setlrmargins{1.25in}{*}{*}
\setmarginnotes{17pt}{51pt}{\onelineskip}
\setheadfoot{\onelineskip}{2\onelineskip}
\setheaderspaces{*}{2\onelineskip}{*}
\checkandfixthelayout
```

An illustration of the layout is shown in Figure 2.4 which also lists the parameter values resulting from the code above, to the nearest point.

I initially used the layout defined in §2.10, which I thought looked reasonable, but then I decided to use the one above in order to save paper when anyone printed out the manual. The wider typeblock also makes it easier for TeX when dealing with lines that include unhyphenatable text, like the LaTeX code.

Andreas Mathias, via Rolf Niepraschk,[10] has suggested that the following might be better for typesetting the manual on A4 paper.

```
\documentclass[a4paper]{memoir}
...
\settrimmedsize{297mm}{210mm}{*}
\setlength{\trimtop}{0pt}
\setlength{\trimedge}{\stockwidth}
\addtolength{\trimedge}{-\paperwidth}
\settypeblocksize{634pt}{448.13pt}{*}
\setulmargins{4cm}{*}{*}
\setlrmargins{*}{*}{1.5}
\setmarginnotes{17pt}{51pt}{\onelineskip}
\setheadfoot{\onelineskip}{2\onelineskip}
\setheaderspaces{*}{2\onelineskip}{*}
\checkandfixthelayout
```

However, the layout that I have provided will print on both letterpaper and A4 sized stock even if it might look better if it was designed for always being printed on one or the other.

2.11 Predefined layouts

The class, like the standard classes, will automatically produce working layouts for letterpaper and a4paper size options. They might be a bit problematic, though, when the page is much smaller, particularly with respect to the space alloted for marginal notes. You perhaps will find the layouts package [Wil04c] useful for checking the page layout.

Some non-default layouts are provided via the commands `\medievalpage`, `\isopage` and `\semiisopage`; these set the size and position of the typeblock with respect to the page. After using any of these commands you *must* call `\checkandfixthelayout` (and after having made any other changes to match the new layout).

[10] Email from niepraschk@ptb.de on 2002/02/05.

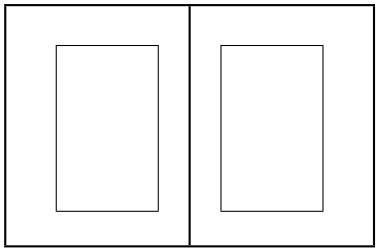


Figure 2.5: Default layout for letterpaper

Figure 2.6: Letterpaper layout: Left `\medievalpage`, Right `\medievalpage[12]`

Figure 2.7: Letterpaper layout: Left `\isopage`, Right `\isopage[12]`

Figure 2.8: Letterpaper layout: Left `\semiisopage`, Right `\semiisopage[12]`

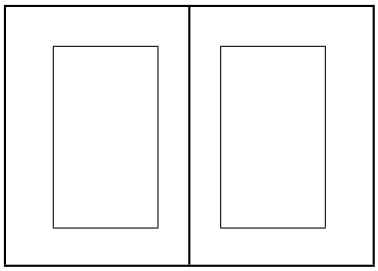


Figure 2.9: Default layout for a4paper

Figure 2.10: A4paper layout: Left `\medievalpage`, Right `\medievalpage[12]`

Figure 2.11: A4paper layout: Left `\isopage`, Right `\isopage[12]`

Figure 2.12: A4paper layout: Left `\semiisopage`, Right `\semiisopage[12]`

`\medievalpage[`⟨*spine*⟩`]`

The `\medievalpage` command generates the position and size of the typeblock according to the principles of medieval scribes and the early printers, as discovered and described by Jan Tschichold [Tsc91]. The basic principle is that the spine, top, fore-edge and bottom margins around the typeblock are in the ratio 2:3:4:6. Typically the spine margin was 1/9 the width of the page, which is what `\medievalpage` assumes by default. This can be changed with the optional ⟨*spine*⟩ argument. For example, to get narrower margins with the spine being 1/12 the page width:

```
\medievalpage[12]
```

Note that ⟨*spine*⟩ must be an integer. See Figures 2.5–2.12.

`\isopage[`⟨*spine*⟩`]`
`\semiisopage[`⟨*spine*⟩`]`

Robert Bringhurst [Bri99] presented a page layout that was especially suitable for ISO proportioned paper, although it can be applied to any page proportion. The `\isopage` command implements this design. The spine margin is normally 1/9 the page width and the top margin is 1/9 the page height, and the fore-edge and bottom margins are respectively twice the spine and top margins.

The `\semiisopage` layout is similar where the spine margin by default is 1/9 the page width, but the top margin is the same as the spine margin. Again the fore-edge and bottom margins are respectively twice the spine and top margins.

The size of the spine (and top) margins can be changed by using the optional ⟨*spine*⟩ argument, which must be an integer. To set the spine margin to be, for example, 1/8 of the page width:

```
\semiisopage[8]% or \isopage[8]
```

The same factor applies to both the spine and top margins in the case of `\isopage`.

Spreads showing a variety of these layouts are in Figure 2.5 through 2.12. These were produced with the aid of the layouts package.

2.12 PLACE TRIMMED PAGE ON THE STOCK

As mentined earlier, when the resulting paper size is smaller than the stock that it is printed on, you'll need to specify the trimmed size (the resulting paper size) using `\settrimmedsize` and you'll want to position that in relation to the stock using `\settrims`.

There are nine common layouts, so we made nine easy tro use macros. The take the same arguments as `\settrimmedsize` and then automatically add an appropriate `\settrims`.

`\setpagetl{`⟨*height*⟩`}{`⟨*width*⟩`}{`⟨*ratio*⟩`}` (top-left)
`\setpagetm{`⟨*height*⟩`}{`⟨*width*⟩`}{`⟨*ratio*⟩`}` (top-middle)
`\setpagetr{`⟨*height*⟩`}{`⟨*width*⟩`}{`⟨*ratio*⟩`}` (top-right)
`\setpagebl{`⟨*height*⟩`}{`⟨*width*⟩`}{`⟨*ratio*⟩`}` (bottom-left)
`\setpagebm{`⟨*height*⟩`}{`⟨*width*⟩`}{`⟨*ratio*⟩`}` (bottom-middle)
`\setpagebr{`⟨*height*⟩`}{`⟨*width*⟩`}{`⟨*ratio*⟩`}` (bottom-right)
`\setpageml{`⟨*height*⟩`}{`⟨*width*⟩`}{`⟨*ratio*⟩`}` (middle-left)
`\setpagecc{`⟨*height*⟩`}{`⟨*width*⟩`}{`⟨*ratio*⟩`}` (center, alias `\setpagemm`)
`\setpagemr{`⟨*height*⟩`}{`⟨*width*⟩`}{`⟨*ratio*⟩`}` (middle-right)

Note: Always remember to combine this with `\checkandfixthelayout`!
For instance, if you are using letterpaper stock but want the final trimmed page size to be A5, then this will put page area at the bottom left of the stock.

```
% letterpaper stock added via class option
\pagebv % sets \paperheight and \paperwidth for A5 paper
\setpagebl{\paperheight}{\paperwidth}{*}
...
\checkandfixthelayout
```

The above macros position the page at the left of the stock because usually trimming of the stock is limited to the top, right, and bottom, the left being the spine when the pages are finally assembled.

In Figure 2.13 we show the effect of the nine `\setpageXX`. The code used for the examples is similar to the code below, and then the result was scaled down.

```
\documentclass[showtrims]{memoir}
\trimLmarks
\setstocksize{18cm}{15cm}
\setpagebl{16cm}{12cm}{*}
\setlrmarginsandblock{15mm}{15mm}{*}
\setulmarginsandblock{15mm}{15mm}{*}
\setheadfoot{5mm}{5mm}
\checkandfixthelayout[fixed]
\pagestyle{empty}
\AtBeginDocument{\LARGE}
\begin{document}
\begin{vplace}
  \centering
  \cs{setpagebl}\marg{height}\marg{width}\marg{ratio}
\end{vplace}
\end{document}
```

Figure 2.13: Example of the nine `\setpageXX` macros for placing a trimmed page on the stock. These are all odd pages, under twoside, even pages will have the left/right trims reversed. The outer fra indicate the stock size. Inside the trim marks hint at where the page is positioned on the stock.

Three

Text and fonts

Presumably you will be creating a document that contains at least some text. In this chapter I talk a little about the kinds of fonts that you might use and how text appears on a page.

3.1 Fonts

(La)TeX comes with a standard set of fonts, designed by Donald Knuth, and known as the Computer Modern font family. The Knuthian fonts were created via the Metafont program [Knu92, Knu87] and are in the form of bitmaps (i.e., each character is represented as a bunch of tiny dots). Fonts of this kind are called *bitmap fonts*. There is also a wide range of Metafont fonts available, created by many others, in addition to the standard set. More modern digital fonts, such as PostScript or TrueType fonts are represented in terms of the curves outlining the character, and it is the job of the printing machine to fill in the outlines (with a bunch of tiny dots). Fonts of this type are called *outline fonts*.

Metafont fonts are designed for a particular display resolution and cannot reasonably be scaled to match an arbitrary display device, whereas outline fonts can be scaled before they are physically displayed.

There is an excessive number of PostScript and TrueType fonts available and these can all, with some amount of effort, be used with LaTeX. How to do that is outside the scope of this work; Alan Hoenig has written an excellent book on the subject [Hoe98] and there is the invaluable *The LaTeX Companion* [MG+04, Chapter 7 Fonts and Encodings]. The original *The LaTeX Graphics Companion* had chapters on PostScript fonts and tools but these were dropped in the second edition to keep it below an overwhelming size. This material has been updated and is available free from `http://xml.cern.ch/lgc2`; you can also get a 'work in progress' about XeTeX, Unicode, and Opentype fonts from the same source. There is less detailed, but also free, information available via CTAN, for example Philipp Lehman's *Font Installation Guide* [Leh04]; even if you are not interested in installing PostScript fonts this is well worth looking at just as an example of the kind of elegant document that can be achieved with LaTeX. If you choose one of the popular PostScript fonts, such as those built into PostScript printers, you may well find that the work has been done for you and it's just a question of using the appropriate package.

A standard LaTeX distribution includes some PostScript fonts and the packages to support them are in the psnfss bundle. Most of the fonts are for normal text work but two supply symbols rather than characters. Table 3.1, although it is specifically for Palatino, shows the glyphs typically available. Tables 3.2 and 3.3 show the glyphs in the two symbol fonts.

These supplied PostScript fonts, their respective LaTeX fontfamily names, and running text examples of each, are:

Table 3.1: Glyphs in the LaTeX supplied Palatino roman font

0	1	2	3	4	5	6	7
8	9	10	11	12	' 13	¡ 14	¿ 15
ı 16	17	` 18	´ 19	ˇ 20	˘ 21	¯ 22	° 23
¸ 24	ß 25	æ 26	œ 27	ø 28	Æ 29	Œ 30	Ø 31
32	! 33	" 34	# 35	$ 36	% 37	& 38	’ 39
(40	) 41	* 42	+ 43	, 44	- 45	. 46	/ 47
0 48	1 49	2 50	3 51	4 52	5 53	6 54	7 55
8 56	9 57	: 58	; 59	< 60	= 61	> 62	? 63
@ 64	A 65	B 66	C 67	D 68	E 69	F 70	G 71
H 72	I 73	J 74	K 75	L 76	M 77	N 78	O 79
P 80	Q 81	R 82	S 83	T 84	U 85	V 86	W 87
X 88	Y 89	Z 90	[91	\ 92	] 93	^ 94	_ 95
‘ 96	a 97	b 98	c 99	d 100	e 101	f 102	g 103
h 104	i 105	j 106	k 107	l 108	m 109	n 110	o 111
p 112	q 113	r 114	s 115	t 116	u 117	v 118	w 119
x 120	y 121	z 122	{ 123	| 124	} 125	~ 126	¨ 127
^ 128	~ 129	Ç 130	Í 131	Î 132	ã 133	ë 134	è 135
š 136	ž 137	Đ 138	139	140	141	142	143
144	145	Š 146	147	148	149	150	151
Ÿ 152	153	Ž 154	155	156	157	158	159
160	161	¢ 162	£ 163	⁄ 164	¥ 165	ƒ 166	§ 167
¤ 168	© 169	“ 170	« 171	‹ 172	› 173	fi 174	fl 175
° 176	– 177	† 178	‡ 179	· 180	181	¶ 182	• 183
‚ 184	„ 185	” 186	» 187	… 188	‰ 189	190	191
À 192	Á 193	Â 194	Ã 195	Ä 196	Å 197	198	˙ 199
È 200	É 201	Ê 202	Ë 203	Ì 204	˝ 205	˛ 206	Ï 207
— 208	Ñ 209	Ò 210	Ó 211	Ô 212	Õ 213	Ö 214	215
216	Ù 217	Ú 218	Û 219	Ü 220	Ý 221	Þ 222	223
à 224	á 225	â 226	ª 227	ä 228	å 229	230	ç 231
Ł 232	é 233	ê 234	º 235	ì 236	í 237	î 238	ï 239
240	ñ 241	ò 242	ó 243	ô 244	õ 245	ö 246	247
ł 248	ù 249	ú 250	û 251	ü 252	ý 253	þ 254	ÿ 255

Table 3.2: Glyphs in the LaTeX distributed Symbol font

32	! 33	∀ 34	# 35	∃ 36	% 37	& 38	∋ 39
(40	) 41	* 42	+ 43	, 44	− 45	. 46	/ 47
0 48	1 49	2 50	3 51	4 52	5 53	6 54	7 55
8 56	9 57	: 58	; 59	< 60	= 61	> 62	? 63
≅ 64	Α 65	Β 66	Χ 67	Δ 68	Ε 69	Φ 70	Γ 71
Η 72	Ι 73	ϑ 74	Κ 75	Λ 76	Μ 77	Ν 78	Ο 79
Π 80	Θ 81	Ρ 82	Σ 83	Τ 84	Υ 85	ς 86	Ω 87
Ξ 88	Ψ 89	Ζ 90	[91	∴ 92	] 93	⊥ 94	_ 95
‾ 96	α 97	β 98	χ 99	δ 100	ε 101	φ 102	γ 103
η 104	ι 105	φ 106	κ 107	λ 108	μ 109	ν 110	ο 111
π 112	θ 113	ρ 114	σ 115	τ 116	υ 117	ϖ 118	ω 119
ξ 120	ψ 121	ζ 122	{ 123	| 124	} 125	∼ 126	127
160	ϒ 161	′ 162	≤ 163	⁄ 164	∞ 165	ƒ 166	♣ 167
♦ 168	♥ 169	♠ 170	↔ 171	← 172	↑ 173	→ 174	↓ 175
° 176	± 177	″ 178	≥ 179	× 180	∝ 181	∂ 182	• 183
÷ 184	≠ 185	≡ 186	≈ 187	… 188	| 189	— 190	↵ 191
ℵ 192	ℑ 193	ℜ 194	℘ 195	⊗ 196	⊕ 197	∅ 198	∩ 199
∪ 200	⊃ 201	⊇ 202	⊄ 203	⊂ 204	⊆ 205	∈ 206	∉ 207
∠ 208	∇ 209	® 210	© 211	™ 212	∏ 213	√ 214	· 215
¬ 216	∧ 217	∨ 218	⇔ 219	⇐ 220	⇑ 221	⇒ 222	⇓ 223
◊ 224	⟨ 225	® 226	© 227	™ 228	∑ 229	⎛ 230	⎜ 231
⎝ 232	⎡ 233	⎢ 234	⎣ 235	⎧ 236	⎨ 237	⎩ 238	⎪ 239
240	⟩ 241	∫ 242	⌠ 243	⎮ 244	⌡ 245	⎞ 246	⎟ 247
⎠ 248	⎤ 249	⎥ 250	⎦ 251	⎫ 252	⎬ 253	⎭ 254	255

Table 3.3: Glyphs in the LaTeX distributed Zapf Dingbat font

32	✁ 33	✂ 34	✃ 35	✄ 36	☎ 37	✆ 38	✇ 39
✈ 40	✉ 41	☛ 42	☞ 43	✌ 44	✍ 45	✎ 46	✏ 47
✐ 48	✑ 49	✒ 50	✓ 51	✔ 52	✕ 53	✖ 54	✗ 55
✘ 56	✙ 57	✚ 58	✛ 59	✜ 60	✝ 61	✞ 62	✟ 63
✠ 64	✡ 65	✢ 66	✣ 67	✤ 68	✥ 69	✦ 70	✧ 71
★ 72	☆ 73	✪ 74	✫ 75	✬ 76	✭ 77	✮ 78	✯ 79
✰ 80	✱ 81	✲ 82	✳ 83	✴ 84	✵ 85	✶ 86	✷ 87
✸ 88	✹ 89	✺ 90	✻ 91	✼ 92	✽ 93	✾ 94	✿ 95
❀ 96	❁ 97	❂ 98	❃ 99	❄ 100	❅ 101	❆ 102	❇ 103
❈ 104	❉ 105	❊ 106	❋ 107	● 108	❍ 109	■ 110	❑ 111
❒ 112	❑ 113	❒ 114	▲ 115	▼ 116	◆ 117	❖ 118	◗ 119
❙ 120	❚ 121	❯ 122	❮ 123	❯ 124	❰ 125	❱ 126	127
160	❡ 161	❢ 162	❣ 163	❤ 164	❥ 165	❦ 166	❧ 167
♣ 168	♦ 169	♥ 170	♠ 171	① 172	② 173	③ 174	④ 175
⑤ 176	⑥ 177	⑦ 178	⑧ 179	⑨ 180	⑩ 181	❶ 182	❷ 183
❸ 184	❹ 185	❺ 186	❻ 187	❼ 188	❽ 189	❾ 190	❿ 191
① 192	② 193	③ 194	④ 195	⑤ 196	⑥ 197	⑦ 198	⑧ 199
⑨ 200	⑩ 201	❶ 202	❷ 203	❸ 204	❹ 205	❺ 206	❻ 207
❼ 208	❽ 209	❾ 210	❿ 211	➔ 212	→ 213	↔ 214	↕ 215
➘ 216	➙ 217	➚ 218	➛ 219	➜ 220	→ 221	⟶ 222	⟹ 223
⟹ 224	➡ 225	➢ 226	➢ 227	➤ 228	➥ 229	➦ 230	➧ 231
➨ 232	➩ 233	➪ 234	➫ 235	➬ 236	➭ 237	➮ 238	➯ 239
240	➱ 241	➲ 242	➳ 243	➴ 244	➵ 245	➶ 246	➷ 247
➸ 248	➹ 249	➺ 250	➻ 251	➼ 252	➽ 253	➾ 254	255

ITC Avant Garde Gothic is a geometric sans type designed by Herb Lubalin and Tom Carnase and based on the logo of the *Avant Garde* magazine. The fontfamily name is pag.

It was a dark and stormy night. While all the good men were coming to the aid of the party, the quick brown dog had jumped over the fast red fox to its great surprise. The cattle had wound slowly o'er the lea and I was in the dark.

'But aren't Kafka's Schloß and Æsop's Œuvres often naïve vis-à-vis the dæmonic phœnix's official rôle in fluffy soufflés?'

Ångelå Beatrice Claire Diana Érica Françoise Ginette Hélène Iris Jackie Kāren Łauŕa María Nătaŀie Øctave Pauline Quêneau Roxanne Sabine Tāja Uršula Vivian Wendy Xanthippe Yvønne Zäzilie

ITC Bookman was originally sold in 1860 by the Miller & Richard foundry in Scot-

land; it was designed by Alexander Phemister. The ITC revival is by Ed Benguiat. The fontfamily name is pbk.

It was a dark and stormy night. While all the good men were coming to the aid of the party, the quick brown dog had jumped over the fast red fox to its great surprise. The cattle had wound slowly o'er the lea and I was in the dark.

'But aren't Kafka's Schloß and Æsop's Œuvres often naïve vis-à-vis the dæmonic phœnix's official rôle in fluffy soufflés?'

Ångelå Beatrice Claire Diana Érica Françoise Ginette Hélène Iris Jackie Kāren Łauŕa María Nătałĭe Øctave Pauline Quêneau Roxanne Sabine Tāja Uršula Vivian Wendy Xanthippe Yvønne Zäzilie

Bitstream Charter was designed by Matthew Carter for display on low resolution devices, and is useful for many applications, including bookwork. The fontfamily name is bch.

It was a dark and stormy night. While all the good men were coming to the aid of the party, the quick brown dog had jumped over the fast red fox to its great surprise. The cattle had wound slowly o'er the lea and I was in the dark.

'But aren't Kafka's Schloß and Æsop's Œuvres often naïve vis-à-vis the dæmonic phœnix's official rôle in fluffy soufflés?'

Ångelå Beatrice Claire Diana Érica Françoise Ginette Hélène Iris Jackie Kāren Łauŕa María Nătałĭe Øctave Pauline Quêneau Roxanne Sabine Tāja Uršula Vivian Wendy Xanthippe Yvønne Zäzilie

Courier is a monospaced font that was originally designed by Howard Kettler at IBM and then later redrawn by Adrian Frutiger. The fontfamily name is pcr.

It was a dark and stormy night. While all the good men were coming to the aid of the party, the quick brown dog had jumped over the fast red fox to its great surprise. The cattle had wound slowly o'er the lea and I was in the dark.

'But aren't Kafka's Schloß and Æsop's Œuvres often naïve vis-à-vis the dæmonic phœnix's official rôle in fluffy soufflés?'

Ångelå Beatrice Claire Diana Érica Françoise Ginette Hélène Iris Jackie Kāren Łauŕa María Nătałĭe Øctave Pauline Quêneau Roxanne Sabine Tāja Uršula Vivian Wendy Xanthippe Yvønne Zäzilie

Helvetica was originally designed for the Haas foundry in Switzerland by Max Miedinger; it was later extended by the Stempel foundry and further refined by Linotype. The fontfamily name is phv.

It was a dark and stormy night. While all the good men were coming to the aid of the party, the quick brown dog had jumped over the fast red fox to its great surprise. The cattle had wound slowly o'er the lea and I was in the dark.

‘But aren’t Kafka’s Schloß and Æsop’s Œuvres often naïve vis-à-vis the dæmonic phœnix’s official rôle in fluffy soufflés?’

Ångelå Beatrice Claire Diana Érica Françoise Ginette Hélène Iris Jackie Kāren Łauŕa María Nătałĭe Øctave Pauline Quêneau Roxanne Sabine Tãja Uršula Vivian Wendy Xanthippe Yvønne Zäzilie

New Century Schoolbook was designed by Morris Benton for ATF (American Type Founders) in the early 20th century. As its name implies it was designed for maximum legibility in schoolbooks. The fontfamily name is `pnc`.

It was a dark and stormy night. While all the good men were coming to the aid of the party, the quick brown dog had jumped over the fast red fox to its great surprise. The cattle had wound slowly o’er the lea and I was in the dark.

‘But aren’t Kafka’s Schloß and Æsop’s Œuvres often naïve vis-à-vis the dæmonic phœnix’s official rôle in fluffy soufflés?’

Ångelå Beatrice Claire Diana Érica Françoise Ginette Hélène Iris Jackie Kāren Łauŕa María Nătałĭe Øctave Pauline Quêneau Roxanne Sabine Tãja Uršula Vivian Wendy Xanthippe Yvønne Zäzilie

Palatino was designed by Hermann Zapf and is one of the most popular typefaces today. The fontfamily name is `ppl`.

It was a dark and stormy night. While all the good men were coming to the aid of the party, the quick brown dog had jumped over the fast red fox to its great surprise. The cattle had wound slowly o’er the lea and I was in the dark.

‘But aren’t Kafka’s Schloß and Æsop’s Œuvres often naïve vis-à-vis the dæmonic phœnix’s official rôle in fluffy soufflés?’

Ångelå Beatrice Claire Diana Érica Françoise Ginette Hélène Iris Jackie Kāren Łauŕa María Nătałĭe Øctave Pauline Quêneau Roxanne Sabine Tãja Uršula Vivian Wendy Xanthippe Yvønne Zäzilie

Times Roman is Linotype’s version of the Times New Roman face designed by Stanley Morison for the Monotype Corporation for printing *The Times* newspaper. The fontfamily name is `ptm`.

It was a dark and stormy night. While all the good men were coming to the aid of the party, the quick brown dog had jumped over the fast red fox to its great surprise. The cattle had wound slowly o’er the lea and I was in the dark.

‘But aren’t Kafka’s Schloß and Æsop’s Œuvres often naïve vis-à-vis the dæmonic phœnix’s official rôle in fluffy soufflés?’

Ångelå Beatrice Claire Diana Érica Françoise Ginette Hélène Iris Jackie Kāren Łauŕa María Nătałĭe Øctave Pauline Quêneau Roxanne Sabine Tãja Uršula Vivian Wendy Xanthippe Yvønne Zäzilie

Utopia was designed by Robert Slimbach and combines Transitional features and contemporary details. The fontfamily name is `put`.

It was a dark and stormy night. While all the good men were coming to the aid of the party, the quick brown dog had jumped over the fast red fox to its great surprise. The cattle had wound slowly o'er the lea and I was in the dark.
'But aren't Kafka's Schloß and Æsop's Œuvres often naïve vis-à-vis the dæmonic phœnix's official rôle in fluffy soufflés?'
Ångelå Beatrice Claire Diana Érica Françoise Ginette Hélène Iris Jackie Kāren Łauŕa María Nătałĭe Øctave Pauline Quêneau Roxanne Sabine Tāja Uršula Vivian Wendy Xanthippe Yvønne Zäzilie

ITC Zapf Chancery *is a Script type fashioned after the chancery handwriting styles of the Italian Renaissance. It was created by Hermann Zapf. The fontfamily name is* `pzc`.

It was a dark and stormy night. While all the good men were coming to the aid of the party, the quick brown dog had jumped over the fast red fox to its great surprise. The cattle had wound slowly o'er the lea and I was in the dark.
'But aren't Kafka's Schloß and Æsop's Œuvres often naïve vis-à-vis the dæmonic phœnix's official rôle in fluffy soufflés?'
Ångelå Beatrice Claire Diana Érica Françoise Ginette Hélène Iris Jackie Kāren Łauŕa María Nătałĭe Øctave Pauline Quêneau Roxanne Sabine Tāja Uršula Vivian Wendy Xanthippe Yvønne Zäzilie

Symbol contains various symbols and Greek letters for mathematical work; these are most easily accessible via the pifont package. The fontfamily name is `psy`.
The available glyphs are shown in Table 3.2.

Zapf Dingbats contains a variety of dingbats which, like the Symbol characters, are most easily accessible via the pifont package. The fontfamily name is `pzd`.
The available glyphs are shown in Table 3.3.

In LaTeX there are three characteristics that apply to a font. These are: (a) the shape, (b) the series (or weight), and (c) the family. Table 3.4 illustrates these and lists the relevant commands to access the different font categories.

The normal body font — the font used for the bulk of the text — is an upright, medium, roman font of a size specified by the font size option for the `\documentclass`.

```
\normalfont
```

The declaration `\normalfont` sets the font to be the normal body font.

There is a set of font declarations, as shown in Table 3.5, that correspond to the commands listed in Table 3.4. The commands are most useful when changing the font for a word or two, while the declarations are more convenient when you want to typeset longer passages in a different font.

Do not go wild seeing how many different kinds of fonts you can cram into your work as in example 3.1.

Source for example 3.1

```
Mixing \textbf{different series, \textsf{families}} and
\textsl{\texttt{shapes,}} \textsc{especially in one sentence,}
is usually \emph{highly inadvisable!}
```

Table 3.4: Font categorisation and commands

Shape	
Upright shape	`\textup{Upright shape}`
Italic shape	`\textit{Italic shape}`
Slanted shape	`\textsl{Slanted shape}`
SMALL CAPS SHAPE	`\textsc{Small Caps shape}`
Series or weight	
Medium series	`\textmd{Medium series}`
Bold series	`\textbf{Bold series}`
Family	
Roman family	`\textrm{Roman family}`
Sans serif family	`\textsf{Sans serif family}`
`Typewriter family`	`\texttt{Typewriter family}`

Table 3.5: Font declarations

Shape	
Upright shape	`{\upshape Upright shape}`
Italic shape	`{\itshape Italic shape}`
Slanted shape	`{\slshape Slanted shape}`
SMALL CAPS SHAPE	`{\scshape Small Caps shape}`
Series or weight	
Medium series	`{\mdseries Medium series}`
Bold series	`{\bfseries Bold series}`
Family	
Roman family	`{\rmfamily Roman family}`
Sans serif family	`{\sffamily Sans serif family}`
`Typewriter family`	`{\ttfamily Typewriter family}`

Typeset example 3.1: Badly mixed fonts

Mixing **different series, families** and *shapes*, ESPECIALLY IN ONE SENTENCE, is usually *highly inadvisable!*

Typeset example 3.2: Sometimes mixed fonts work

DES DAMES DU TEMPS JADIS

Prince, n'enquerez de sepmaine
Ou elles sont, ne de cest an,
Qu'a ce reffrain ne vous remaine:
Mais ou sont les neiges d'antan?

Prince, do not ask in a week
Or yet in a year where they are,
I could only give you this refrain:
But where are the snows of yesteryear?

François Villon [1431–1463?]

On the other hand there are occasions when several fonts may be used for a reasonable effect, as in example 3.2.

Source for example 3.2

```
\begin{center}
\textsc{Des Dames du Temps Jadis}
\end{center}%
\settowidth{\versewidth}{Or yet in a year where they are}
\begin{verse}[\versewidth] \begin{itshape}
Prince, n'enquerez de sepmaine \\*
Ou elles sont, ne de cest an, \\*
Qu'a ce reffrain ne vous remaine: \\*
Mais ou sont les neiges d'antan?
\end{itshape}

Prince, do not ask in a week \\*
Or yet in a year where they are, \\*
I could only give you this refrain: \\*
But where are the snows of yesteryear?
\end{verse}
\begin{flushright}
{\bfseries Fran\c{c}ois Villon} [1431--1463?]
\end{flushright}
```

`\emph{`⟨*text*⟩`}`

The `\emph` command is a font changing command that does not fit into the above scheme of things. What it does is to typeset its ⟨*text*⟩ argument using a different font than the sur-

Typeset example 3.3: Emphasis upon emphasis

The \emph command is used to produce some text that should be *emphasised for some reason and can be* infrequently interspersed *with some further emphasis* just like in this sentence.

rounding text. By default, \emph switches between an upright shape and an italic shape. The commands can be nested to produce effects like those in the next example.

Source for example 3.3

```
The \verb?\emph? command is used to produce some text that
should be \emph{emphasised for some reason and can be
\emph{infrequently interspersed} with some further emphasis}
just like in this sentence.
```

\eminnershape{⟨*shape*⟩}

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

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

which you can change if you wish.

3.2 FONT SIZES

The Computer Modern Metafont fonts come in a fixed number of sizes, with each size being subtly different in shape so that they blend harmoniously. Traditionally, characters were designed for each size to be cut, and Computer Modern follows the traditional type design. For example, the smaller the size the more likely that the characters will have a relatively larger width. Outline fonts can be scaled to any size, but as the scaling is typically linear, different sizes do not visually match quite as well.

Computer Modern fonts come in twelve sizes which, rounded to a point, are: 5, 6, 7, 8, 9, 10, 11, 12, 14, 17, 20 and 25pt. In LaTeX the size for a particular font is specifed by a macro name like \scriptsize and not by points; for example \scriptsize, not 7pt.[1] The actual size of, say, \scriptsize characters, is not fixed but depends on the type size option given for the document.

Standard LaTeX provides ten declarations, illustrated in Table 3.6, for setting the type size, which means that two of the sizes are not easily accessible. Which two depend on the class and the selected point size option. However, for normal typesetting four different sizes should cover the majority of needs, so there is plenty of scope with a mere ten to choose from.

[1] It is possible to use points but that is outside the scope of this manual.

Table 3.6: Standard font size declarations

`\tiny`	tiny	`\scriptsize`	scriptsize
`\footnotesize`	footnotesize	`\small`	small
`\normalsize`	normalsize	`\large`	large
`\Large`	Large	`\LARGE`	LARGE
`\huge`	huge	`\Huge`	Huge

Table 3.7: Standard font sizes

Class option	10pt	11pt	12pt
`\tiny`	5pt	6pt	6pt
`\scriptsize`	7pt	8pt	8pt
`\footnotesize`	8pt	9pt	10pt
`\small`	9pt	10pt	11pt
`\normalsize`	10pt	11pt	12pt
`\large`	12pt	12pt	14pt
`\Large`	14pt	14pt	17pt
`\LARGE`	17pt	17pt	20pt
`\huge`	20pt	20pt	25pt
`\Huge`	25pt	25pt	25pt

The `\normalsize` is the size that is set as the class option and is the size used for the body text. The `\footnotesize` is the size normally used for typesetting footnotes. The standard classes use the other sizes, usually the larger ones, for typesetting certain aspects of a document, for example sectional headings.

With respect to the standard classes, the memoir class provides a wider range of the document class type size options and adds two extra font size declarations, namely `\miniscule` and `\HUGE`, one at each end of the range.

The memoir class font size declarations names are given in Table 3.8 together with the name set in the specified size relative to the manual's `\normalsize` font. font. The corresponding actual sizes are given in Table 3.9.

Whereas the standard font sizes range from 5pt to 25pt, memoir provides for fonts ranging from 4pt to 132pt. That is:

from the 4pt size (the 9pt `\miniscule` size)
through the 9pt normal size

through the

Table 3.8: The memoir class font size declarations

`\miniscule`	miniscule	`\tiny`	tiny
`\scriptsize`	scriptsize	`\footnotesize`	footnotesize
`\small`	small	`\normalsize`	normalsize
`\large`	large	`\Large`	Large
`\LARGE`	LARGE	`\huge`	huge
`\Huge`	Huge	`\HUGE`	HUGE

Table 3.9: The memoir class font sizes

Class option	9pt	10pt	11pt	12pt	14pt	17pt	20pt	25pt	30pt	36pt	48pt	60pt
`\miniscule`	4pt	5pt	6pt	7pt	8pt	9pt	10pt	11pt	12pt	14pt	17pt	20pt
`\tiny`	5pt	6pt	7pt	8pt	9pt	10pt	11pt	12pt	14pt	17pt	20pt	25pt
`\scriptsize`	6pt	7pt	8pt	9pt	10pt	11pt	12pt	14pt	17pt	20pt	25pt	30pt
`\footnotesize`	7pt	8pt	9pt	10pt	11pt	12pt	14pt	17pt	20pt	25pt	30pt	36pt
`\small`	8pt	9pt	10pt	11pt	12pt	14pt	17pt	20pt	25pt	30pt	36pt	48pt
`\normalsize`	9pt	10pt	11pt	12pt	14pt	17pt	20pt	25pt	30pt	36pt	48pt	60pt
`\large`	10pt	11pt	12pt	14pt	17pt	20pt	25pt	30pt	36pt	48pt	60pt	72pt
`\Large`	11pt	12pt	14pt	17pt	20pt	25pt	30pt	36pt	48pt	60pt	72pt	84pt
`\LARGE`	12pt	14pt	17pt	20pt	25pt	30pt	36pt	48pt	60pt	72pt	84pt	96pt
`\huge`	14pt	17pt	20pt	25pt	30pt	36pt	48pt	60pt	72pt	84pt	96pt	108pt
`\Huge`	17pt	20pt	25pt	30pt	36pt	48pt	60pt	72pt	84pt	96pt	108pt	120pt
`\HUGE`	20pt	25pt	30pt	36pt	48pt	60pt	72pt	84pt	96pt	108pt	120pt	132pt

60pt normal size

to the 132pt size

(the
60pt
\HUGE

size).

This extended range, though, is only accessible if you are using outline fonts and the extrafontsizes class option. If you are using bitmap fonts then, for example, the \HUGE font will be automatically limited to 25pt, and the minimum size of a \miniscule font is 5pt.

3.3 Spaces

3.3.1 Paragraphs

In traditional typography the first line of a paragraph, unless it comes immediately after a chapter or section heading, is indented. Also, there is no extra space between paragraphs. Font designers go to great pains to ensure that they look good when set with the normal leading. Sometimes, such as when trying to meet a University's requirements for the layout of your thesis, you may be forced to ignore the experience of centuries.

If you like the idea of eliminating paragraph indentation and using extra inter-paragraph space to indicate where paragraphs start and end, consider how confused your reader will be if the last paragraph on the page ends with a full line; how will the reader know that a new paragraph starts at the top of the following page?

```
\par
\parskip
\abnormalparskip{⟨length⟩}
\nonzeroparskip
\traditionalparskip
```

In the input text the end of a paragraph is indicated either by leaving a blank line, or by the \par command. The length \parskip is the inter-paragraph spacing, and is normally 0pt. You can change this by saying, for example:

```
\setlength{\parskip}{2\baselineskip}
```

but you are likely to find that many things have changed that you did not expect, because LaTeX uses the \par command in many places that are not obvious.

If, in any event, you wish to do a disservice to your readers you can use \abnormalparskip to set the inter-paragraph spacing to a value of your own choosing. Using the \nonzeroparskip will set the spacing to what might be a reasonable non-zero value.[2] Both these macros try and eliminate the worst of the side effects that occur if you just simply change \parskip directly.

[2] Except that all values except zero are unreasonable.

Following the \traditionalparskip declaration all will be returned to their traditional values.

I based the code for these functions upon the NTG classes [LEB04] which indicated some of the pitfalls in increasing the spacing. The difficulty is that \par, and hence \parskip, occurs in many places, some unexpected and others deeply buried in the overall code.

\parindent

The length \parindent is the indentation at the start of a paragraph's first line. This is usually of the order of 1 to $1\frac{1}{2}$ em. To make the first line of a paragraph flushleft set this to zero:

```
\setlength{\parindent}{0pt}
```

3.3.2 Double spacing

Some of those that have control over the visual appearance of academic theses like them to be 'double spaced'. This, of course, will make the thesis harder to read[3] but perhaps that is the purpose, or maybe they have stock (shares) in papermills and lumber companies, as the theses were usually required to be printed single sided as well.

\baselineskip \onelineskip

The length \baselineskip is the space, or leading, between the baselines of adjacent text lines, and is constant throughout a paragraph. The value may change depending on the size of the current font. More precisely, the \baselineskip depends on the font being used at the *end* of the paragraph. The class also provides the length \onelineskip which is the default leading for the normal body font.[4] As far as the class is concerned this is a constant value; that is, unlike \baselineskip, it never alters \onelineskip. You can use (fractions) of \onelineskip to specify vertical spaces in terms of normal text lines.

The following is heavily based on the setspace package [Tob00], but the names have been changed to avoid any clashes. Like the nonzero \parskip, the \baselineskip rears its head in many places, and it is hard for a package to get at the internals of the overlying class and kernel code. This is not to say that all is well with trying to deal with it at the class level.

\OnehalfSpacing \OnehalfSpacing*
\DoubleSpacing \DoubleSpacing*

The declaration \OnehalfSpacing increases the spacing between lines so that they appear to be double spaced (especially to the thesis layout arbiters), while the declaration \DoubleSpacing really doubles the spacing between lines which really looks bad; but if you have to use it, it is there. The spacing in footnotes and floats (e.g., captions) is unaltered, which is usually required once the controllers see what a blanket double spacing brings. Sometimes it is also required to make page notes and floats (including captions) in 'double' spacing. The starred version of the two macros above takes care of this. Alternatively the spacing in page notes (i.e. footnotes and friends) or floats an be set explicitly using

[3] I certainly found them so when I was having to read them before examining the candidates for their degrees. The writers of the regulations, which were invariably single spaced, seemed immune to any suggestions.

[4] That is \onelineskip is set in the memX.clo file corresponding to the font size class option. For 10pt, the size is set via mem10.clo.

```
\setPagenoteSpacing{⟨factor⟩}
\setFloatSpacing{⟨factor⟩}
```

`\setFloatSpacing` should go after say `\OnehalfSpacing*` if used.

```
\SingleSpacing
\SingleSpacing*
\setSingleSpace{⟨factor⟩}
```

The `\setSingleSpace` command is meant to be used to adjust *slightly* the normal spacing betwen lines, perhaps because the font being used looks too crampled or loose. The effect is that the normal `\baselineskip` spacing will be multiplied by ⟨*factor*⟩, which should be close to 1.0. Using `\setSingleSpace` will also reset the float and page note spacings.

The declaration `\SingleSpacing` returns everthing to normal, or at least the setting from `\setSingleSpace` if it has been used. It will also reset float and page note spacings to the same value.

Note. `\SingleSpacing` *will also issue a* `\vskip\baselineskip` *at the end (which is ignored if* `\SingleSpacing` *is used in the preamble). This skip makes sure that comming from* `\DoubleSpacing` *to* `\SingleSpacing` *still looks ok.*

But in certain cases, this skip is unwanted. Therefore as of 2018 we added a `\SingleSpacing*` *that is equal to* `\SingleSpacing` *but* does not *add this skip.*

```
\begin{SingleSpace} ...\end{SingleSpace}
\begin{Spacing}{⟨factor⟩} ... \end{Spacing}
\begin{OnehalfSpace} ... \end{OnehalfSpace}
\begin{OnehalfSpace*} ... \end{OnehalfSpace*}
\begin{DoubleSpace} ... \end{DoubleSpace}
\begin{DoubleSpace*} ... \end{DoubleSpace*}
```

These are the environments corresponding to the declarations presented earlier, for when you want to change the spacing locally.

```
\setDisplayskipStretch{⟨fraction⟩}
\memdskipstretch
\noDisplayskipStretch
\memdskips
```

If you have increased the interlinear space in the text you may wish, or be required, to increase it around displays (of maths). The declaration `\setDisplayskipStretch` will increase the before and after displayskips by ⟨*fraction*⟩, which must be at least 0.0. More precisely, it defines `\memdskipstretch` to be ⟨*fraction*⟩. The `\noDisplayskipStretch` declaration sets the skips back to their normal values. It is equivalent to

```
\setDisplayskipStretch{0.0}
```

The skips are changed within the macro `\memdskips` which, in turn, is called by `\everydisplay`. If you find odd spacing around displays then redefine `\memdskips` to do nothing. Its orginal specification is:

```
\newcommand*{\memdskips}{%
  \advance\abovedisplayskip \memdskipstretch\abovedisplayskip
  \advance\belowdisplayskip \memdskipstretch\belowdisplayskip
```

```
\advance\abovedisplayshortskip \memdskipstretch\abovedisplayshortskip
\advance\belowdisplayshortskip \memdskipstretch\belowdisplayshortskip}
```

If you need to use a `minipage` as a stand-alone item in a widely spaced text then you may need to use the `vminipage` environment instead to get the before and after spacing correct.

3.4 OVERFULL LINES

TeX tries very hard to keep text lines justified while keeping the interword spacing as constant as possible, but sometimes fails and complains about an overfull hbox.

```
\fussy \sloppy
\begin{sloppypar} ... \end{sloppypar}
\midsloppy
\begin{midsloppypar} ... \end{midsloppypar}
```

The default mode for LaTeX typesetting is `\fussy` where the (variation of) interword spacing in justified text is kept to a minimum. Following the `\sloppy` declaration there may be a much looser setting of justified text. The `sloppypar` environment is equivalent to:

```
{\par \sloppy ... \par}
```

Additionally the class provides the `\midsloppy` declaration (and the `midsloppypar` environment) which allows a setting somewhere between `\fussy` and `\sloppy`. Using `\midsloppy` you will get fewer overfull lines compared with `\fussy` and fewer obvious large interword spaces than with `\sloppy`. I have used `\midsloppy` for this manual; it hasn't prevented overfull lines or noticeably different interword spaces, but has markedly reduced them compared with `\fussy` and `\sloppy` respectively.

3.5 SLOPPYBOTTOM

TeX does its best to avoid widow and orphan lines — a widow is where the last line of a paragraph ends up at the top of a page, and an orphan[5] is when the first line of a paragraph is at the bottom of a page.

The following is the generally suggested method of eliminating widows and orphans, but it may well result in some odd looking pages, especially if `\raggedbottom` is not used.

```
\clubpenalty=10000
\widowpenalty=10000
\raggedbottom
```

```
\enlargethispage{⟨length⟩}
```

You can use `\enlargethispage` to add or subtract to the text height on a particular page to move a line forwards or backwards between two pages.

Here is one person's view on the matter:

[5] Knuth uses the term 'club' instead of the normal typographers' terminology.

> . . . 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.
> /iaw (from CTT, *widow handling?*, May 2006)

As so often happens, Donald Arseneau came up with a solution.

`\sloppybottom`

The declaration `\sloppybottom` lets TeX put an extra line at the bottom of a page to avoid a widow on the following page.

The `\topskip` must have been increased beforehand for this to work (a 60% increase is reasonable) and this will push the text lower on the page. Run `\checkandfixthelayout` after the change (which may reduce the number of lines per page). For example, in the preamble:

```
\setlength{\topskip}{1.6\topskip}
\checkandfixthelayout
\sloppybottom
```

The late Michael Downes provided the following (from CTT *widow/orphan control package (for 2e)?*, 1998/08/31):

> For what it's worth here are the penalty values that I use when I don't [want] to *absolutely* prohibit widow/orphan break, but come about as close as TeX permits otherwise. This is copied straight out of some code that I had lying around. I guess I could wrap it into package form and post it to CTAN.
> Michael Downes

```
% set \clubpenalty, etc. to distinctive values for use
% in tracing page breaks. These values are chosen so that
% no single penalty will absolutely prohibit a page break, but
% certain combinations of two or more will.
\clubpenalt=9996
\widowpenalty=9999
\brokenpenalty=4991
% Reiterate the default value of \redisplaypenalty, for
% completeness.
% Set postdisplaypenalty to a fairly high value to discourage a
% page break between a display and a widow line at the end of a
% paragraph.
\predisplaypenalty=10000
\postdisplaypenalty=1549
% And then \displaywidowpenalty should be at least as high as
% \postdisplaypenalty, otherwise in a situation where two displays
% are separated by two lines, TeX will prefer to break between the
% two lines, rather than before the first line.
\displaywidowpenalty=1602
```

As you can see, perfect automatic widow/orphan control is problematic though typographers are typically more concerned about widows than orphans — a single line of a paragraph somehow looks worse at the top of a page than at the bottom. If all else fails, the solution is either to live with the odd line or to reword the text.

3.6 TEXT CASE

The standard kernel `\MakeUppercase{`⟨*text*⟩`}` and `\MakeLowercase{`⟨*text*⟩`}` basically upper or lower case everything it can get its hands on. This is not particularly nice if the ⟨*text*⟩ contain, say, math.

In order to help with this we load the textcase package ([Car19]) by David Carlisle (with the `overload` option) which provide the `\MakeTextUppercase` and `\MakeTextLowercase` macros.[6]

> ***Caveat.*** This change was made in October 2019, earlier we embedded a copy of `textcase` into the class it self. Now we check to see of you have a copy and use that, if found, if not we use our old embedded copy.
>
> If, for some reason, you want the original `\MakeUppercase` and `\MakeLowercase` macros back, add
>
> ```
> \MemRestoreOrigMakecase
> ```
>
> to your preamble.

The following is DCs own documentation of the provided code changed to match the typography we use.[7]

```
\MakeTextUppercase{⟨text⟩}
\MakeTextLowercase{⟨text⟩}
```

`\MakeTextUppercase` and `\MakeTextLowercase` are versions of the standard `\MakeUppercase` and `\MakeLowercase` that do not change the case of any math sections in their arguments.

```
\MakeTextUppercase{abc\ae\ \( a = b \)  and $\alpha \neq a$
  or even \ensuremath{x=y} and $\ensuremath{x=y}$}
```

Should produce:

ABCÆ $a = b$ AND $\alpha \neq a$ OR EVEN $x = y$ AND $x = y$

We incorporates some changes suggested by Donald Arseneau so that as well as math mode, the arguments of `\cite`, `\label` and `\ref` are also prevented from being uppercased. So you can now go

```
\MakeTextUppercase{%
   Text in section~\ref{sec:text-case}, about \cite[pp 2--4]{textcase}}
```

which produces

[6] Overload causes the original `\MakeUppercase` and `\MakeLowercase` to be copies of `\MakeTextUppercase` and `\MakeTextLowercase`.

[7] Note: This documentation will probably be removed in a future version and replaced by a summary and a reference to [Car19].

TEXT IN SECTION 3.6, ABOUT [Car19, PP 2–4]

If, instead, the standard `\MakeUppercase` were used here, the keys 'sec:text-case' and 'textcase' would be uppercased and generate errors about undefined references to SEC:TEXT-CASE and TEXTCASE.

`\NoCaseChange{`⟨*text*⟩`}`

Sometimes there may be a special section of text that should not be uppercased. This can be marked with `\NoCaseChange`, as follows.

```
\MakeTextUppercase{%
   Text \NoCaseChange{More Text} yet more text}
```

which produces

TEXT More Text YET MORE TEXT

`\NoCaseChange` has other uses. If for some reason you need a tabular environment within an uppercased section, then you need to ensure that the name 'tabular' and the preamble (eg 'll') does not get uppercased:

```
\MakeTextUppercase{%
   Text \NoCaseChange{\begin{tabular}{ll}}%
                        table&stuff\\goes&here
        \NoCaseChange{\end{tabular}}
   More text}
```

which produces

TEXT TABLE STUFF
GOES HERE MORE TEXT

3.6.1 Nested text

The commands defined here only skip math sections and `\ref` arguments if they are not 'hidden' inside a `{ }` brace group. All text inside such a group will be made uppercase just as with the standard `\MakeUppercase`.

```
\MakeTextUppercase{a b {c $d$} $e$}
```

produces

A B C D e

Of course, this restriction does not apply to the arguments of the supported commands `\ensuremath`, `\label`, `\ref`, and `\cite`.

If you cannot arrange for your mathematics to be at the outer level of brace grouping, you should use the following basic technique (which works even with the standard `\MakeUppercase` command). Define a new command that expands to your math expression, and then use that command, with `\protect`, in the text to be uppercased. Note that if the text being uppercased is in a section title or other moving argument you may need to make the definition in the document preamble, rather than just before the section command, so that the command is defined when the table of contents file is read.

```
\MakeTextUppercase{%
       Text \fbox{$a=b$ and $x=y$}}%

\newcommand{\mathexprone}{$a=b$}
\newcommand{\mathexprtwo}{$x=y$}
\MakeTextUppercase{%
       Text \fbox{\protect\mathexprone\ and \protect\mathexprtwo}}%
```

which produces

TEXT $\boxed{A = B \text{ AND } X = Y}$
TEXT $\boxed{a = b \text{ AND } x = y}$

See also [Car19] for some information about upper casing citations using a non-nummeric style.

Four

Titles

The standard classes provide little in the way of help in setting the title page(s) for your work, as the `\maketitle` command is principally aimed at generating a title for an article in a technical journal; it provides little for titles for works like theses, reports or books. For these I recommend that you design your own title page layout[1] using the regular LaTeX commands to lay it out, and ignore `\maketitle`.

Quoting from Ruari McLean [McL80, p. 148] in reference to the title page he says:

> The title-page states, in words, the actual title (and sub-title, if there is one) of the book and the name of the author and publisher, and sometimes also the number of illustrations, but it should do more than that. From the designer's point of view, it is the most important page in the book: it sets the style. It is the page which opens communication with the reader…
>
> If illustrations play a large part in the book, the title-page opening should, or may, express this visually. If any form of decoration is used inside the book, e.g., for chapter openings, one would expect this to be repeated or echoed on the title-page.
>
> Whatever the style of the book, the title-page should give a foretaste of it. If the book consists of plain text, the title-page should at least be in harmony with it. The title itself should not exceed the width of the type area, and will normally be narrower…

A pastiche of McLean's title page is shown in Figure 4.1.

The typeset format of the `\maketitle` command is virtually fixed within the LaTeX standard classes. This class provides a set of formatting commands that can be used to modify the appearance of the title information; that is, the contents of the `\title`, `\author` and `\date` commands. It also keeps the values of these commands so that they may be printed again later in the document. The class also inhibits the normal automatic cancellation of titling commands after `\maketitle`. This means that you can have multiple instances of the same, or perhaps different, titles in one document; for example on a half title page and the full title page. Hooks are provided so that additional titling elements can be defined and printed by `\maketitle`. The `\thanks` command is enhanced to provide various configurations for both the marker symbol and the layout of the thanks notes.

Generally speaking, if you want anything other than minor variations on the `\maketitle` layout then it is better to ignore `\maketitle` and take the whole layout into your own hands so you can place everthing just where you want it on the page.

[1] If you are producing a thesis you are probably told just how it must look.

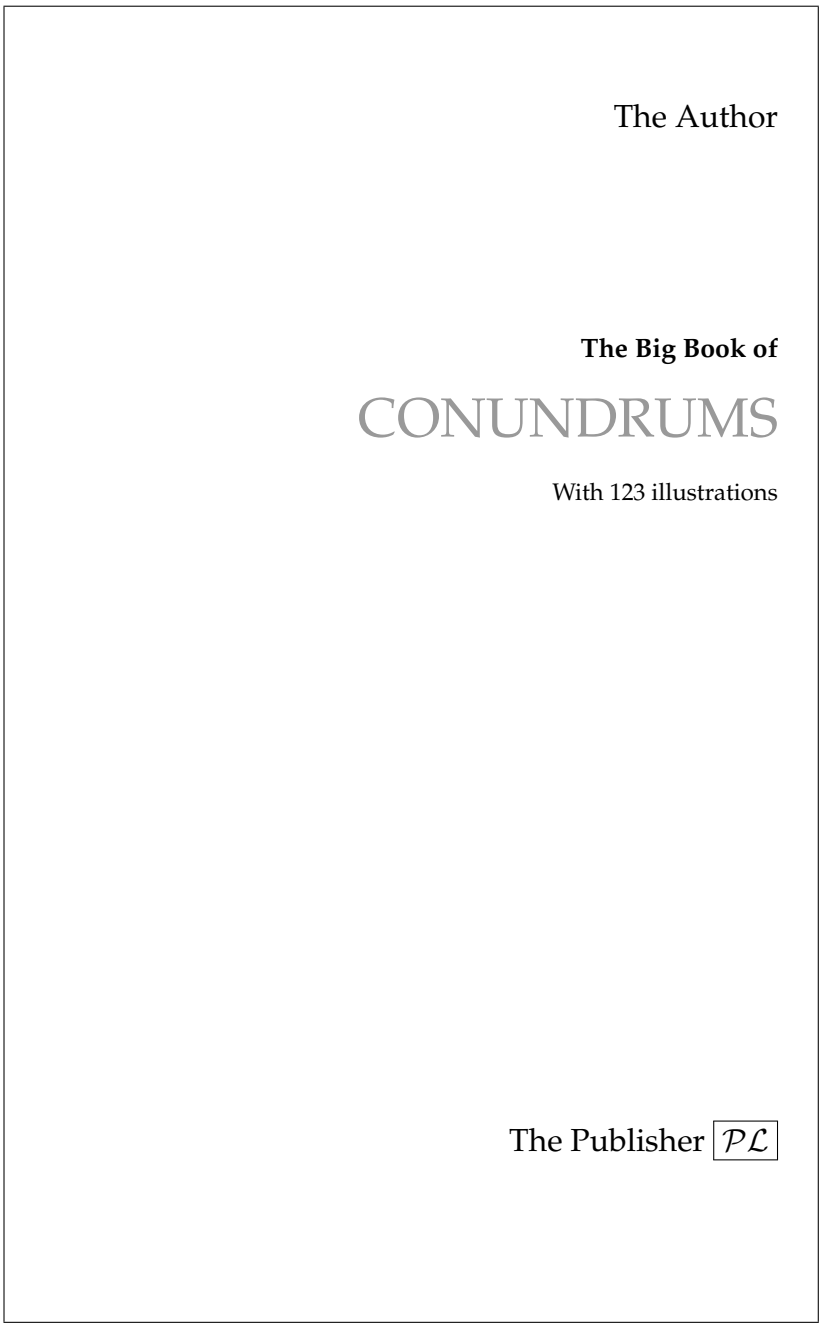


Figure 4.1: Layout of a title page for a book on typography

CONUNDRUMS CONSIDERED AS PUZZLES FOR THE MIND

By

The Candidate

A Thesis Submitted to the Graduate

Faculty of The University

in Partial Fulfillment of the

Requirements for the Degree of

DEGREE

Major Subject: Logic

Approved by the
Examining Committee:

_____
A Professor, Thesis Advisor

_____
Another Professor, Thesis Advisor

_____
A Faculty, Member

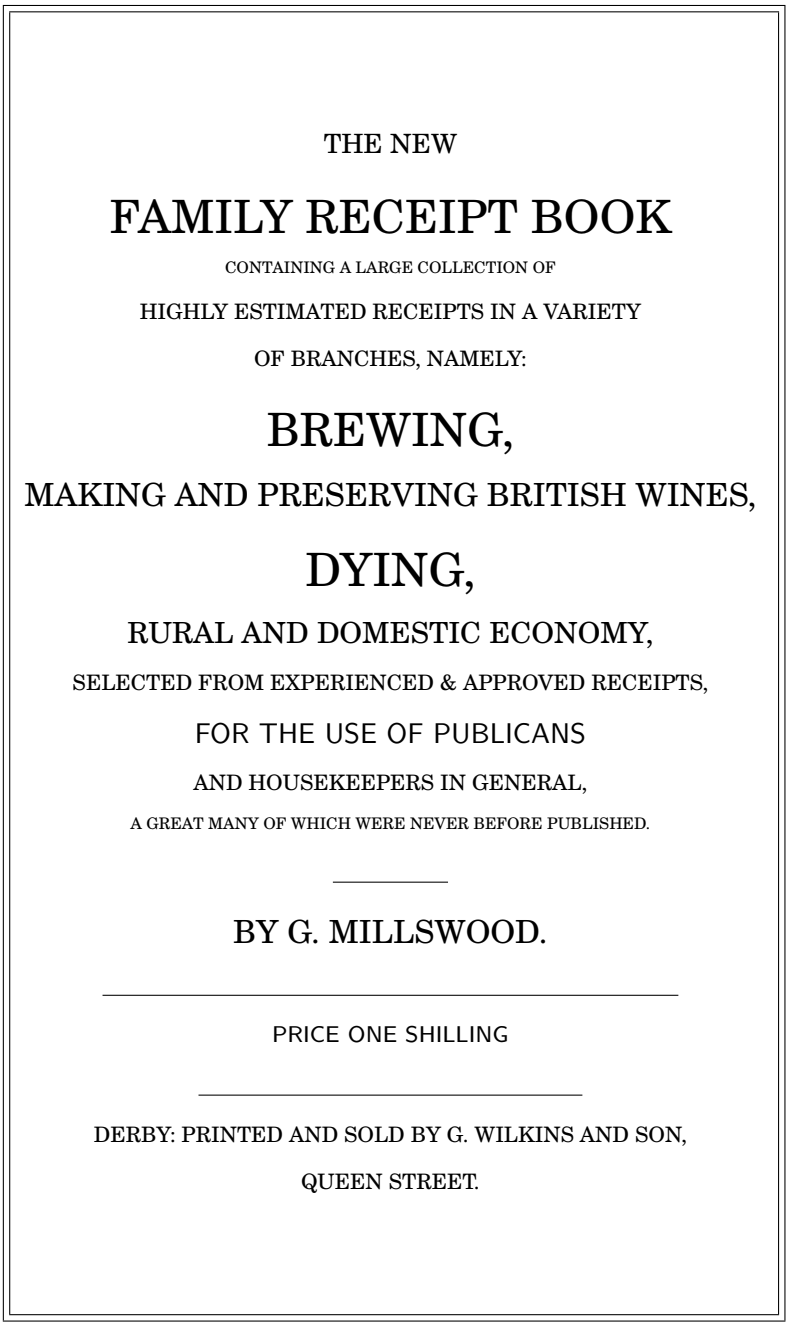
_____
Another Faculty, Member

_____
A Third Faculty, Member

The University
The Address

The Date

Figure 4.2: Example of a mandated title page style for a doctoral thesis



THE NEW

FAMILY RECEIPT BOOK

CONTAINING A LARGE COLLECTION OF

HIGHLY ESTIMATED RECEIPTS IN A VARIETY

OF BRANCHES, NAMELY:

BREWING,

MAKING AND PRESERVING BRITISH WINES,

DYING,

RURAL AND DOMESTIC ECONOMY,

SELECTED FROM EXPERIENCED & APPROVED RECEIPTS,

FOR THE USE OF PUBLICANS

AND HOUSEKEEPERS IN GENERAL,

A GREAT MANY OF WHICH WERE NEVER BEFORE PUBLISHED.

BY G. MILLSWOOD.

PRICE ONE SHILLING

DERBY: PRINTED AND SOLD BY G. WILKINS AND SON,

QUEEN STREET.

Figure 4.3: Example of a Victorian title page

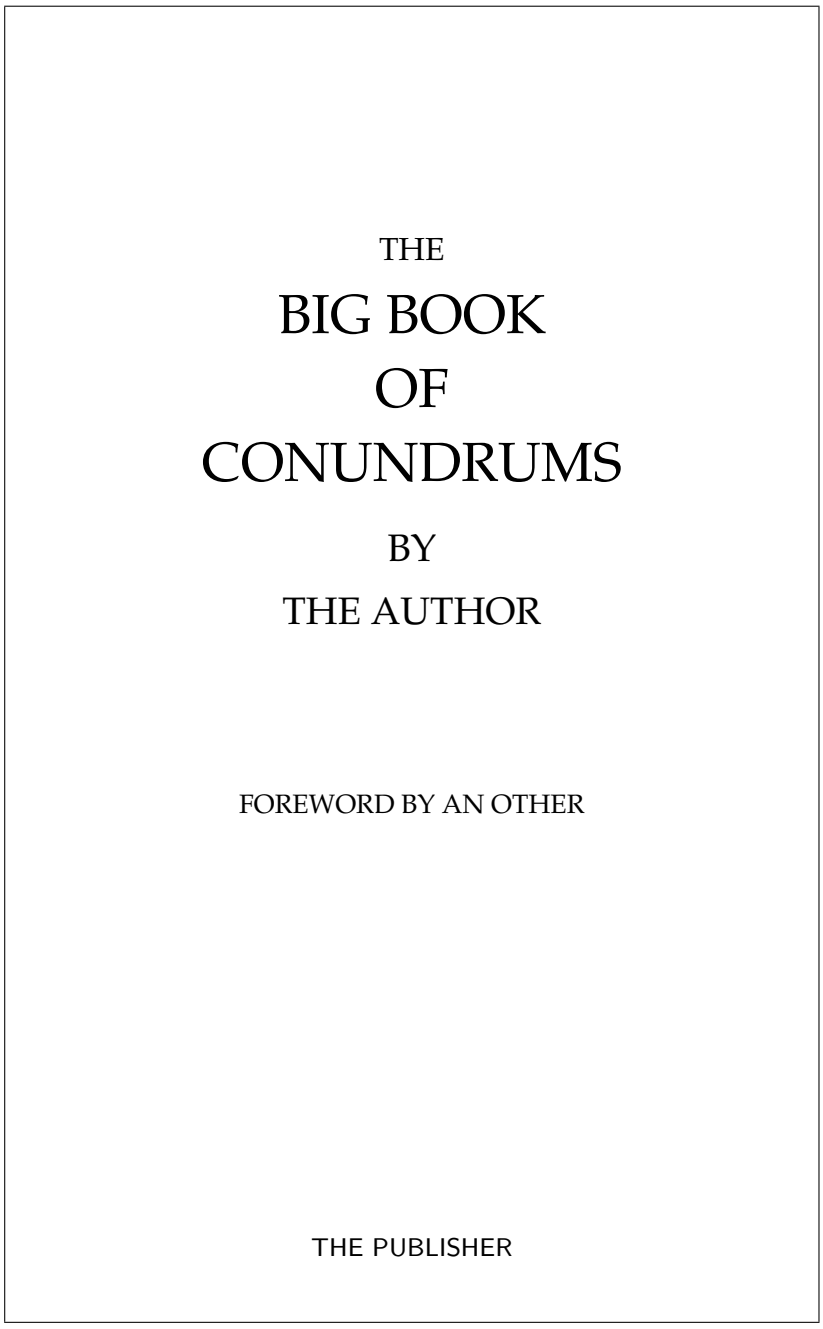


Figure 4.4: Layout of a title page for a book on book design

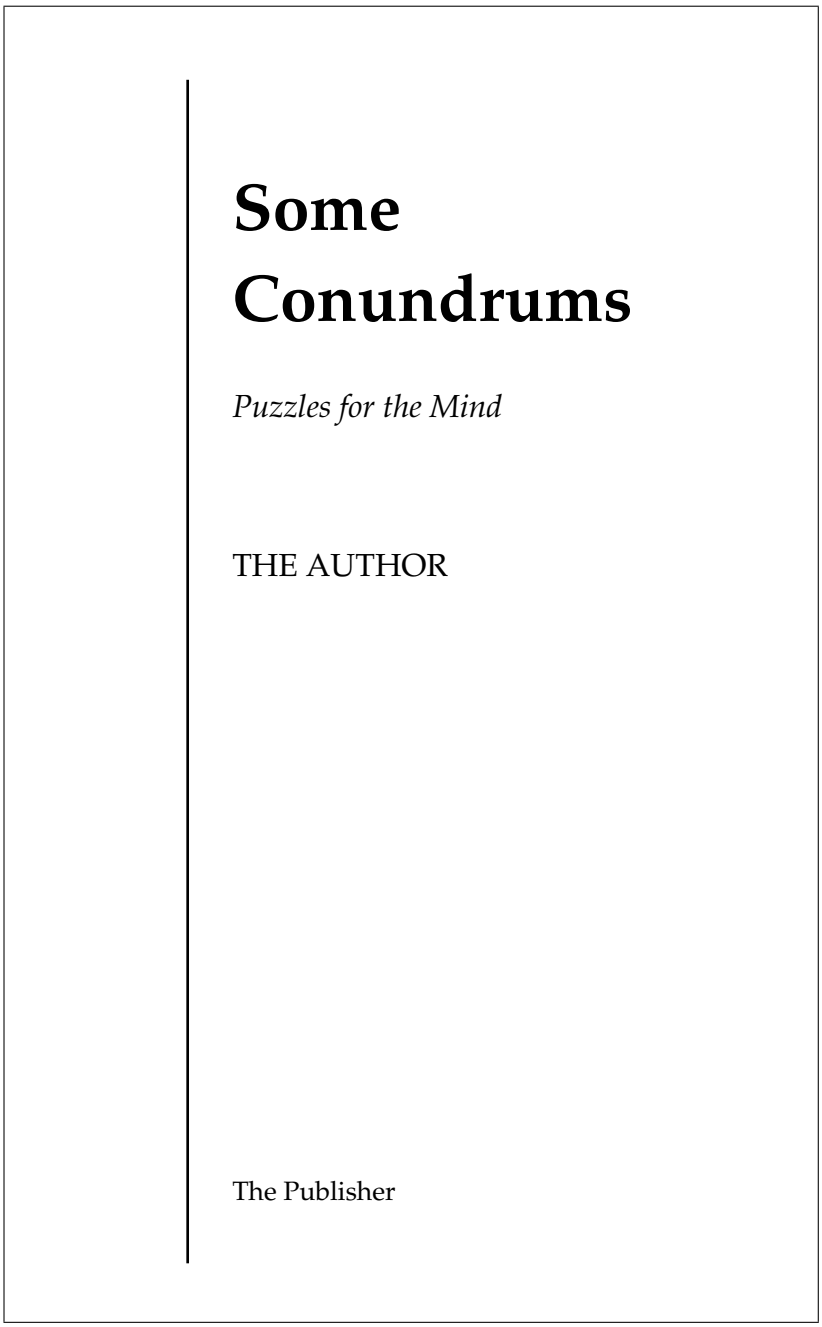


Figure 4.5: Layout of a title page for a book about books

4.1 Styling the titling

The facilities provided for typesetting titles are limited, essentially catering for the kind of titles of articles published in technical journals. They can also be used as a quick and dirty method for typesetting titles on reports, but for serious work, such as a title page for a book or thesis, each title page should be handcrafted. For instance, a student of mine, Donald Sanderson used LaTeX to typeset his doctoral thesis, and Figure 4.2 shows the title page style mandated by Rensselear Polytechnic Institute as of 1994. Many other examples of title pages, together with the code to create them, are in [Wil07a].

Another handcrafted title page from [Wil07a] is shown in Figure 4.3. This one is based on an old booklet I found that was published towards the end of the 19th century and exhibits the love of Victorian printers in displaying a variety of types; the rules are an integral part of the title page. For the purposes of this manual I have used New Century Schoolbook, which is part of the regular LaTeX distribution, rather than my original choice of Century Old Style which is one of the commercial FontSite fonts licensed from the SoftMaker/ATF library, supported for LaTeX through Christopher League's estimable work [Lea03].

The title page in Figure 4.4 follows the style of of *The Design of Books* [Wil93] and a page similar to Nicholas Basbanes *A Gentle Madness: Bibliophiles, Bibliomanes, and the Eternal Passion for Books* is illustrated in Figure 4.5. These are all from [Wil07a] and handcrafted.

In contrast the following code produces the standard `\maketitle` layout.

Source for example 4.1

```
\title{MEANDERINGS}
\author{T. H. E. River \and
        A. Wanderer\thanks{Supported by a grant from the
        R. Ambler's Fund}\\
        Dun Roamin Institute, NY}
\date{1 April 1993\thanks{First drafted on 29 February 1992}}
...
\maketitle
```

This part of the class is a reimplementation of the titling package [Wil01g].

The class provides a configurable `\maketitle` command. The `\maketitle` command as defined by the class is essentially

```
\newcommand{\maketitle}{%
   \vspace*{\droptitle}
   \maketitlehooka
   {\pretitle \title \posttitle}
   \maketitlehookb
   {\preauthor \author \postauthor}
   \maketitlehookc
   {\predate \date \postdate}
   \maketitlehookd
   \thispagestyle{title}
```

Typeset example 4.1: Example \maketitle title

MEANDERINGS

T. H. E. River and A. Wanderer*
Dun Roamin Institute, NY

1 April 1993†

⋮

* Supported by a grant from the R. Ambler's Fund
† First drafted on 29 February 1992

```
}
```

where the *title* pagestyle is initially the same as the *plain* pagestyle. The various macros used within \maketitle are described below.

\pretitle{⟨*text*⟩} \posttitle{⟨*text*⟩}
\preauthor{⟨*text*⟩} \postauthor{⟨*text*⟩}
\predate{⟨*text*⟩} \postdate{⟨*text*⟩}

These six commands each have a single argument, ⟨*text*⟩, which controls the typesetting of the standard elements of the document's \maketitle command. The \title is effectively processed between the \pretitle and \posttitle commands; that is, like:

```
{\pretitle \title \posttitle}
```

and similarly for the \author and \date commands. The commands are initialised to mimic the normal result of \maketitle typesetting in the report class. That is, the default definitions of the commands are:

```
\pretitle{\begin{center}\LARGE}
\posttitle{\par\end{center}\vskip 0.5em}
\preauthor{\begin{center}
           \large \lineskip 0.5em%
           \begin{tabular}[t]{c}}
\postauthor{\end{tabular}\par\end{center}}
\predate{\begin{center}\large}
\postdate{\par\end{center}}
```

They can be changed to obtain different effects. For example to get a right justified sans-serif title and a left justifed small caps date:

```
\pretitle{\begin{flushright}\LARGE\sffamily}
\posttitle{\par\end{flushright}\vskip 0.5em}
\predate{\begin{flushleft}\large\scshape}
```

```
\postdate{\par\end{flushleft}}
```

\droptitle

The \maketitle command puts the title at a particular height on the page. You can change the vertical position of the title via the length \droptitle. Giving this a positive value will lower the title and a negative value will raise it. The default definition is:

```
\setlength{\droptitle}{0pt}
```

\maketitlehooka \maketitlehookb \maketitlehookc \maketitlehookd

These four hook commands are provided so that additional elements may be added to \maketitle. These are initially defined to do nothing but can be renewed. For example, some publications want a statement about where an article is published immediately before the actual titling text. The following defines a command \published that can be used to hold the publishing information which will then be automatically printed by \maketitle.

```
\newcommand{\published}[1]{%
   \gdef\puB{#1}}
\newcommand{\puB}{}
\renewcommand{\maketitlehooka}{%
   \par\noindent \puB}
```

You can then say:

```
\published{Originally published in
          \textit{The Journal of ...}\thanks{Reprinted with permission}}
...
\maketitle
```

to print both the published and the normal titling information. Note that nothing extra had to be done in order to use the \thanks command in the argument to the new \published command.

\begin{titlingpage} text \end{titlingpage} \begin{titlingpage*} text \end{titlingpage*} \titlingpageend{⟨*twoside code*⟩}{⟨*oneside code*⟩}

When one of the standard classes is used with the titlepage option, \maketitle puts the title elements on an unnumbered page and then starts a new page numbered page 1. The standard classes also provide a titlepage environment which starts a new unnumbered page and at the end starts a new page numbered 1. You are entirely responsible for specifying exactly what and where is to go on this title page. If \maketitle is used within the titlepage environment it will start yet another page.

This class provides neither a titlepage option nor a titlepage environment; instead it provides the titlingpage environment which falls between the titlepage option and the titlepage environment. Within the titlingpage environment you can use the \maketitle command, and any others you wish. The *titlingpage* pagestyle is used, and at the end it starts another ordinary page numbered one (\begin{titlingpage*} does

note reset the page number). The *titlingpage* pagestyle is initially defined to be the same as the *empty* pagestyle.

At the end of a `titlingpage` clearing code is issued, which can send you to the next page or the next right handed page. Using `\titlingpageend{`⟨*twoside code*⟩`}{`⟨*oneside code*⟩`}`, you can specify what this clearing code should be. The default is `\cleardoublepage` and `\clearpage` respectively.[2] However a better choice might be to just let it follow `\clearforchapter`:

```
\titlingpageend{\clearforchapter}{\clearforchapter}
```

– using this value, `titlingpage` will work as expected with `openany`.

For example, to put both the title and an abstract on a title page, with a *plain* pagestyle:

```
\begin{document}
\begin{titlingpage}
\aliaspagestyle{titlingpage}{plain}
\setlength{\droptitle}{30pt} lower the title
\maketitle
\begin{abstract}...\end{abstract}
\end{titlingpage}
```

However, it is not required to use `\begin{titlingpage}` to create a title page, you can use regular LaTeX typesetting without any special environment. That is like:

```
\pagestyle{empty}
%%% Title, author, publisher, etc.,  here
\cleardoublepage
...
```

By default, titling information is centered with respect to the width of the typeblock. Occasionally someone asks on the `comp.text.tex` newsgroup how to center the titling information on a title page with respect to the width of the physical page. If the typeblock is centered with respect to the physical page, then the default centering suffices. If the typeblock is not physically centered, then the titling information either has to be shifted horizontally or `\maketitle` has to be made to think that the typeblock has been shifted horizontally. The simplest solution is to use the `\calccentering` and `adjustwidth*` command and environment. For example:

```
\begin{titlingpage}
  \calccentering{\unitlength}
  \begin{adjustwidth*}{\unitlength}{-\unitlength}
    \maketitle
  \end{adjustwidth*}
\end{titlingpage}
```

`\title{`⟨*text*⟩`}` `\thetitle`
`\author{`⟨*text*⟩`}` `\theauthor`
`\date{`⟨*text*⟩`}` `\thedate`

[2] Thus this refactorization will not change existing documents, LM, 2018/03/06.

In the usual document classes, the contents (⟨*text*⟩) of the `\title`, `\author` and `\date` macros used for `\maketitle` are unavailable once `\maketitle` has been issued. The class provides the `\thetitle`, `\theauthor` and `\thedate` commands that can be used for printing these elements of the title later in the document, if desired.

`\and` `\andnext`

The macro `\and` is used within the argument to the `\author` command to add some extra space between the author's names. The class `\andnext` macro inserts a newline instead of a space. Within the `\theauthor` macro both `\and` and `\andnext` are replaced by a comma.

The class does not follow the standard classes' habit of automatically killing the titling commands after `\maketitle` has been issued. You can have multiple `\title`, `\author`, `\date` and `\maketitle` commands in your document if you wish. For example, some reports are issued with a title page, followed by an executive summary, and then they have another, possibly modified, title at the start of the main body of the report. This can be accomplished like this:

```
\title{Cover title}
...
\begin{titlingpage}
\maketitle
\end{titlingpage}
...
\title{Body title}
\maketitle
...
```

`\killtitle` `\keepthetitle`
`\emptythanks`

The `\killtitle` macro makes all aspects of titling, including `\thetitle` etc., unavailable from the point that it is issued (using this command will save some macro space if the `\thetitle`, etc., commands are not required). Using this command is the class's manual version of the automatic killing performed by the standard classes. The `\keepthetitle` command performs a similar function, except that it keeps the `\thetitle`, `\theauthor` and `\thedate` commands, while killing everything else.

The `\emptythanks` command discards any text from prior use of `\thanks`. This command is useful when `\maketitle` is used multiple times — the `\thanks` commands in each use just stack up the texts for output at each use, so each subsequent use of `\maketitle` will output all previous `\thanks` texts together with any new ones. To avoid this, put `\emptythanks` before each `\maketitle` after the first.

4.2 Styling the thanks

The class provides a configurable `\thanks` command.

`\thanksmarkseries{`⟨*format*⟩`}`
`\symbolthanksmark`

Any \thanks are marked with symbols in the titling and footnotes. The command \thanksmarkseries can be used to change the marking style. The ⟨*format*⟩ argument is the name of one of the formats for printing a counter. The name is the same as that of a counter format but without the backslash. To have the \thanks marks as lowercase letters instead of symbols do:

```
\thanksmarkseries{alph}
```

Just for convenience the \symbolthanksmark command sets the series to be footnote symbols. Using this class the potential names for ⟨*format*⟩ are: arabic, roman, Roman, alph, Alph, and fnsymbol.

\continuousmarks

The \thanks command uses the footnote counter, and normally the counter is zeroed after the titling so that the footnote marks start from 1. If the counter should not be zeroed, then just specify \continuousmarks. This might be required if numerals are used as the thanks markers.

\thanksheadextra{⟨*pre*⟩}{⟨*post*⟩}

The \thanksheadextra command will insert ⟨*pre*⟩ and ⟨*post*⟩ before and after the thanks markers in the titling block. By default ⟨*pre*⟩ and ⟨*post*⟩ are empty. For example, to put parentheses round the titling markers do:

```
\thanksheadextra{(}{)}
```

\thanksmark{⟨*n*⟩}

It is sometimes desireable to have the same thanks text be applied to, say, four out of six authors, these being the first 3 and the last one. The command \thanksmark{⟨*n*⟩} is similar to \footnotemark[⟨*n*⟩] in that it prints a thanks mark identical to that of the ⟨*n*⟩'th \thanks command. No changes are made to any thanks in the footnotes. For instance, in the following the authors Alpha and Omega will have the same mark:

```
\title{The work\thanks{Draft}}
\author{Alpha\thanks{ABC},
        Beta\thanks{XYZ} and
        Omega\thanksmark{2}}
\maketitle
```

\thanksmarkstyle{⟨*defn*⟩}

By default the thanks mark at the foot is typeset as a superscript. In the class this is specifed via

```
\thanksmarkstyle{\textsuperscript{#1}}
```

where #1 will be replaced by the thanks mark symbol. You can change the mark styling if you wish. For example, to put parentheses round the mark and typeset it at normal size on the baseline:

```
\thanksmarkstyle{(#1)}
```

\thanksmarkwidth

The thanks mark in the footnote is typeset right justified in a box of width `\thanksmarkwidth`. The first line of the thanks text starts after this box. The initialisation is

```
\setlength{\thanksmarkwidth}{1.8em}
```

giving the default position.

`\thanksmarksep`

The value of the length `\thanksmarksep` controls the indentation the second and subsequent lines of the thanks text, with respect to the end of the mark box. As examples:

```
\setlength{\thanksmarksep}{0em}
```

will align the left hand ends of of a multiline thanks text, while:

```
\setlength{\thanksmarksep}{-\thanksmarkwidth}
```

will left justify any second and subsequent lines of the thanks text. This last setting is the initialised value, giving the default appearance.

`\thanksfootmark`

A thanks mark in the footnote region is typeset by `\thanksfootmark`. The code for this is roughly:

```
\newcommand{\thanksfootmark}{%
  \hbox to\thanksmarkwidth{\hfil\normalfont%
    \thanksscript{\thefootnote}}}
```

You should not need to change the definition of `\thanksfootmark` but you may want to change the default definitions of one or more of the macros it uses.

`\thanksscript{`⟨*text*⟩`}`

This is initially defined as:

```
\newcommand{\thanksscript}[1]{\textsuperscript{#1}}
```

so that `\thanksscript` typesets its argument as a superscript, which is the default for thanks notes.

`\makethanksmark`
`\makethanksmarkhook`

The macro `\makethanksmark` typesets both the thanks marker (via `\thanksfootmark`) and the thanks text. You probably will not need to change its default definition. Just in case, though, `\makethanksmark` calls the macro `\makethanksmarkhook` before it does any typesetting. The default definition of this is:

```
\newcommand{\makethanksmarkhook}{}
```

which does nothing.

You can redefine `\makethanksmarkhook` to do something useful. For example, if you wanted a slightly bigger baseline skip you could do:

```
\renewcommand{\makethanksmarkhook}{\fontsize{8}{11}\selectfont}
```

where the numbers 8 and 11 specify the point size of the font and the baseline skip respectively. In this example 8pt is the normal `\footnotesize` in a 10pt document, and 11pt is the baselineskip for `\footnotesize` text in an 11pt document (the normal baseline skip for `\footnotesize` in a 10pt document is 9.5pt); adjust these numbers to suit.

```
\thanksrule
\usethanksrule
\cancelthanksrule
```

By default, there is no rule above `\thanks` text that appears in a `titlingpage` environment. If you want a rule in that environment, put `\usethanksrule` before the `\maketitle` command, which will then print a rule according to the current definition of `\thanksrule`. `\thanksrule` is initialised to be a copy of `\footnoterule` as it is defined at the end of the preamble. The definition of `\thanksrule` can be changed after `\begin{document}`. If the definition of `\thanksrule` is modified and a `\usethanksrule` command has been issued, then the redefined rule may also be used for footnotes. Issuing the command `\cancelthanksrule` will cause the normal `\footnoterule` definition to be used from thereon; another `\usethanksrule` command can be issued later if you want to swap back again.

The parameters for the vertical positioning of footnotes and thanks notes, and the default `\footnoterule` are the same (see Figure 12.1 on page 241). You will have to change one or more of these if the vertical spacings of footnotes and thanks notes are meant to be different.

Five

Abstracts

Abstracts do not normally appear in books but they are often an essential part of an article in a technical journal. Reports may or may not include an abstract, but if so it will often be called a 'Summary'. There may be an even shorter abstract as well, often called an 'Executive Summary', for those who feel that details are irrelevant.

In the standard classes appearance of the `abstract` environment is fixed. The class provides a set of controls adjusting the appearance of an `abstract` environment.

Questions about how to have a one-column abstract in a two-column paper seem to pop up fairly regularly on the `comp.text.tex` newsgroup. While an answer based on responses on CTT is provided in the FAQ, the class provides a more author-friendly means of accomplishing this.

5.1 Styling

Much of this part of the class is a reimplementation of the abstract package [Wil01a].

The typeset format of an `abstract` in a report or article class document depends on the class options.[1] The formats are:

- titlepage class option: The abstract heading (i.e., value of `\abstractname`) is typeset centered in a bold font; the text is set in the normal font and to the normal width.
- twocolumn class option: The abstract heading is typeset like an unnumbered section; the text is set in the normal font and to the normal width (of a single column).
- Default (neither of the above class options): The abstract heading is typeset centered in a small bold font; the text is set in a small font and indented like the `quotation` environment.

This class provides an `abstract` environment and handles to modify the typesetting of an `abstract`.

```
\begin{abstract} text \end{abstract}
```

There is nothing special about using the `abstract` environment. Formatting is controlled by the macros described below.

```
\abstractcol
\abstractintoc
\abstractnum
\abstractrunin
```

[1] The `abstract` environment is not available for the book class.

The normal format for an abstract is with a centered, bold title and the text in a small font, inset from the margins.

The `\abstractcol` declaration specifies that an abstract in a twocolumn class option document should be typeset like a normal, unnumbered chapter. The `\abstractintoc` specifies that the abstract title should be added to the ToC. The declaration `\abstractnum` specifies that the abstract should be typeset like a numbered chapter and `\abstractrunin` specifies that the title of the abstract should look like a run-in heading; these two declarations are mutually exclusive. Note that the `\abstractnum` declaration has no effect if the abstract is in the `\frontmatter`.

```
\abstractname
\abstractnamefont
\abstracttextfont
```

`\abstractname` (default 'Abstract') is used as the title for the `abstract` environment and is set using the `\abstractnamefont`. The body of the abstract is typeset using the `\abstracttextfont`. These two commands can be redefined to change the fonts if you wish. The default definitions are

```
\newcommand{\abstractnamefont}{\normalfont\small\bfseries}
\newcommand{\abstracttextfont}{\normalfont\small}
```

```
\absleftindent \absrightindent
\absparindent \absparsep
```

This version of `abstract` uses a `list` environment for typesetting the text. These four lengths can be changed (via `\setlength` or `\addtolength`) to adjust the left and right margins, the paragraph indentation, and the vertical skip between paragraphs in this environment. The default values depend on whether or not the twocolumn class option is used. The general layout parameters for lists are illustrated in Figure 8.2.

> ***Caveat.*** The default values for `\absparindent` and `\absparsep` are `\parindent` and `\parskip`. But, since users may change these values in the preamble, we delay setting `\absparindent` and `\absparsep` to the beginning of the document. Thus if you want to change these values remember to use
>
> ```
> \AtBeginDocument{
>   \setlength{\absparindent}{...}
>   \setlength{\absparsep}{...}
> }
> ```

```
\abslabeldelim{⟨text⟩}
```

If the `\abstractrunin` declaration has been given, the heading is typeset as a run-in heading. That is, it is the first piece of text on the first line of the abstract text. The ⟨*text*⟩ argument of `\abslabeldelim` is typeset immediately after the heading. By default it is defined to do nothing, but if you wanted, for example, the `\abstractname` to be followed by a colon and some extra space you could specify

```
\abslabeldelim{:\quad}
```

```
\absnamepos
```

If the \abstractrunin declaration is not used then the heading is typeset in its own environment, specified by \absnamepos. The default definition is

```
\newcommand{\absnamepos}{center}
```

It can be defined to be one of flushleft, center, or flushright to give a left, centered or right aligned heading; or to any other appropriate environment which is supported by a used package.

\abstitleskip

With the \abstractrunin declaration a horizontal space of length \abstitleskip is typeset before the heading. For example, if \absparindent is non-zero, then:

```
\AtBeginDocument{
  \setlength{\abstitleskip}{-\absparindent}
}
```

will typeset the heading flush left (we have to delay as the value of \absparindent is zero in the preamble and is set at the start of the doc).

Without the \abstractrunin declaration, \abstitleskip is aditional vertical space (either positive or negative) that is inserted between the abstract name and the text of the abstract.

5.2 One column abstracts

The usual advice [FAQ] about creating a one-column abstract in a twocolumn document is to write code like this:

```
\documentclass[twocolumn...]{...}
...
\twocolumn[
   \begin{@twocolumnfalse}
     \maketitle               need full-width title
     \begin{abstract}
        abstract text...
     \end{abstract}
   \end{@twocolumnfalse}
]
... hand make footnotes for any \thanks commands
...
```

\begin{onecolabstract} text \end{onecolabstract} \saythanks

The class provides a onecolabstract environment that you can use for a one column abstract in a twocolumn document, and it is used like this:

```
\documentclass[twocolumn...]{memoir}
...
\twocolumn[
   \maketitle               need full-width title
   \begin{onecolabstract}
```

```
      abstract text...
    \end{onecolabstract}
]
\saythanks % typesets any \thanks commands
...
```

The command `\saythanks` ensures that any `\thanks` texts from an earlier `\maketitle` are printed out as normal.

If you want, you can use the `onecolabstract` environment in place of the `abstract` environment — it doesn't have to be within the optional argument of the `\twocolumn` command. In fact, `onecolabstract` internally uses `abstract` for the typesetting.

6 Document divisions

For this chapter the *pedersen* chapterstyle has been used in order to demonstrate how it appears.

In this chapter I first discuss the various kinds of divisions within a book and the commands for typesetting these.

After that I describe the class methods for modifying the appearance of the chapter and other sectional titles (subheads). The facilities described here provide roughly the same as you would get if you used the titlesec [Bez99] and sectsty [McD98] packages together; the commands are different, though.

6.1 Logical divisions

As described earlier there are three main logical divisions to a book; the front matter, main matter and back matter. There are three LaTeX commands that correspond to these, namely `\frontmatter`, `\mainmatter` and `\backmatter`.

`\frontmatter` `\frontmatter*`

The `\frontmatter` declaration sets the folios to be printed in lowercase roman numerals, starts the page numbering from i, and prohibits any numbering of sectional divisions. Caption, equations, etc., will be numbered continuously. The starred version of the command, `\frontmatter*`, is similar to the unstarred version except that it makes no changes to the page numbering or the print style for the folios. Even though `\chapter` and other divisions will not be numbered their titles will be added to the ToC.

If it is to be used at all, the `\frontmatter` declaration should come before any text is set, otherwise the pagination scheme will be in disarray (in books pagination starts on the first page).

`\mainmatter` `\mainmatter*`

The `\mainmatter` declaration, which is the default at the start of a document, sets the folios to be printed in arabic numerals, starts the page numbering from 1, and sections and above will be numbered. Float captions, equations, etc., will be numbered per chapter. The starred version of the command, `\mainmatter*`, is similar to the unstarred version except that it makes no changes to the page numbering or the print style for the folios.

Please note that `\mainmatter` will not only change the folio numbers to arabic and restart it at 1, it will also make sure it starts at the next coming recto page. (Even when running under the `openany` option).

`\backmatter`

The `\backmatter` declaration makes no change to the pagination or folios but does prohibit sectional division numbering, and captions, etc., will be numbered continuously.

If you have other types of floats that might be used in the front- main- or backmatter, then you can change some internals to add these to be numbered in the same manner as we do with figures and tables. They are defined as

```
\newcommand\@memfront@floats{%
  \counterwithout{figure}{chapter}
```

```
   \counterwithout{table}{chapter}}
\newcommand\@memmain@floats{%
   \counterwithin{figure}{chapter}
   \counterwithin{table}{chapter}}
\newcommand\@memback@floats{%
    \counterwithout{figure}{chapter}
    \counterwithout{table}{chapter}
    \setcounter{figure}{0}
    \setcounter{table}{0}}
```

The macros can also be changed in case you want to have consecutive figure numbering throughout, i.e.,

```
\makeatletter
\counterwithout{figure}{chapter}
\counterwithout{table}{chapter}
\renewcommand\@memfront@floats{}
\renewcommand\@memmain@floats{}
\newcommand\@memback@floats{}
\makeatother
```

in the preamble.

6.2 SECTIONAL DIVISIONS

The memoir class lets you divide a document up into eight levels of named divisions. They range from book, part through chapter and down to sub-paragraph. A particular sectional division is specified by one of the commands `\book`, `\part`, `\chapter`, `\section`, `\subsection`, which is probably as deep as you want to go. If you really need finer divisions, they are `\subsubsection`, `\paragraph` and lastly `\subparagraph`. The sectional commands, except for `\book` and `\part`, have the same form, so rather than describing each one in turn I will use `\section` as model for all but the two exceptions.

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

There are two forms of the command; the starred version is simpler, so I'll describe its effects first — it just typesets ⟨*title*⟩ in the document in the format for that particular sectional division. Like the starred version, the plain version also typesets ⟨*title*⟩ in the document, but it may be numbered. Diferent forms of the division title are available for the Table of Contents (ToC) and a running header, as follows:

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

A `\section` command restarts the numbering of any `\subsections` from one. For most of the divisions the ⟨*title*⟩ is put on the page where the command was issued. The `\book`, `\part` and `\chapter` commands behave a little differently.

The \book and \part commands are simpler and both behave in the same way.

```
\book{⟨title⟩}
\part{⟨title⟩}
```

The \book{⟨*title*⟩} command puts the book name (default Book), number and ⟨*title*⟩ on a page by itself. The numbering of books has no effect on the numbering of \parts or \chapters. Similarly the \part{⟨*title*⟩} command puts the part name (default Part), number and ⟨*title*⟩ on a page by itself. The numbering of parts has no effect on the numbering of \chapters.

Later I'll give a list of LaTeX's default names, like Part.

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

The \chapter command starts a new page and puts the chapter name (default Chapter), number and ⟨*title*⟩ at the top of the page. It restarts the numbering of any \sections from one. If no optional arguments are specified, ⟨*title*⟩ is used as the ToC entry and for any page headings. If one optional argument is specified this is ⟨*toc-title*⟩ and is used for the ToC entry and for page headings. If both optional arguments are specified the ⟨*head-title*⟩ is used for page headings.

The \chapter* command starts a new page and puts ⟨*title*⟩ at the top of the page. It makes no ToC entry, changes no numbers and by default changes no page headings. If the optional ⟨*head-title*⟩ argument is given, this is used for page headings. Use of the optional argument has the side-effect that the secnumdepth counter is set to maxsecnumdepth (see below for an explanation of these).

When the article option is in effect, however, things are slightly different. New chapters do not necessarily start on a new page. The \mainmatter command just turns on sectional numbering and starts arabic page numbering; the \backmatter command just turns off sectional numbering. The \tableofcontents command and friends, as well as any other commands created via \newlistof, *always*[1] call thispagestyle{chapter}. If you are using the article option you will probably want to ensure that the *chapter* pagestyle is the same as you normally use for the document.

Unlike the standard classes the ⟨*title*⟩ is typeset ragged right. This means that if you need to force a linebreak in the ⟨*title*⟩ you have to use \newline instead of the more usual \\. For instance

```
\section{A broken\newline title}
```

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

```
\raggedbottomsection
\normalbottomsection
\bottomsectionskip
\bottomsectionpenalty
```

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

The \raggedbottomsection declaration will typeset any pages that are short because of a moved subhead as though \raggedbottom was in effect for the short page; other pages are not affected. The length \bottomsectionskip controls the amount of stretch on the short page. Setting it to zero allows the last line to be flush with the bottom of the typeblock. The default setting of 10mm appears to remove any stretch. \bottomsectionpenalty control the penalty it costs to make a page break at this point. The detault is zero as the stretch is usually enough, by setting it to a negative integer one can be a bit more incouraging regarding a possible page break.

The declaration \normalbottomsection, which is the default, cancels any previous \raggedbottomsection declaration.

6.2.1 Appendices

Appendices normally come after the main text and are often considered to be part of the \mainmatter as they are normally numbered (the \backmatter declaration turns off all sectional numbering).

```
\appendix
\appendixname
```

The \appendix declaration changes the numbering of chapters to an alphabetic form and also changes the names of chapters from \chaptername (default Chapter) to the value of \appendixname (default Appendix). Thus, the first and any subsequent \chapters after the \appendix command will be 'Appendix A ...', 'Appendix B ...', and so on. That is as far as the standard classes go but this class provides more ways of dealing with appendices.

```
\appendixpage
\appendixpage*
\appendixpagename
```

The \appendixpage command generates a part-like page (but no name or number) with the title given by the value of \appendixpagename (default Appendices). It also makes an entry in the ToC using \addappheadtotoc (see below). The starred version generates the appendix page but makes no ToC entry.

```
\addappheadtotoc
\appendixtocname
```

The command \addappheadtotoc adds an entry to the ToC. The title is given by the value of \appendixtocname (default Appendices).

```
\begin{appendices} text \end{appendices}
```

The appendices environment acts like the \appendix command in that it resets the numbering and naming of chapters. However, at the end of the environment, chapters are restored to their original condition and any chapter numbers continue in sequence as though the appendices environment had never been there.

```
\begin{subappendices} text \end{subappendices}
\namedsubappendices \unnamedsubappendices
```

The subappendices environment can be used to put appendices at the end of a chapter. Within the environment \section starts a new sub-appendix. You may put

Table 6.1: Division levels

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

\addappheadtotoc at the start of the environment if you want a heading entry in the ToC. If you put the declaration \namedsubappendices *before* the subappendices environment, the sub-appendix number in the body of the document will be preceded by the value of \appendixname. The \unnamedsubappendices declaration, which is the default, may be used to switch off this behaviour.

Caveat: The implementation of the named subappendices make use of \setsecnumformat, thus if you have used this command to change the formating of the section number you will need to re-do this in a special manner inside the subappendices environemt. Something like this (where a user wanted to use old style numerials for sectioning numbers)

```
\begin{subappendices}
\setsecnumformat{\sectionname\
  \oldstylenums{\csname the#1\endcsname\quad}}
```

The macro sectionname is a special macro that only lives inside the subappendices environment and is only available when \namedsubappendices is applied.

6.3 Numbering

Each type of sectional division has an associated *level* as shown in Table 6.1. Divisions are numbered if the value of the secnumdepth counter is equal to or greater than their level. For example, with

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

then subsections up to book will be numbered.

\setsecnumdepth{⟨*secname*⟩}
\maxsecnumdepth{⟨*secname*⟩}

Instead of having to remember the levels if you want to change what gets numbered you can use the \setsecnumdepth command. It sets secnumdepth so that divisions ⟨*secname*⟩ and above will be numbered. The argument ⟨*secname*⟩ is the name of a sectional division without the backslash. For example, to have subsections and above numbered:

```
\setsecnumdepth{subsection}
```

You can also use `all` or `none` for ⟨*secname*⟩ which will either turn on numbering for all levels, or turn off numbering altogether.

When used in the preamble `\setsecnumdepth` also calls `\maxsecnumdepth`, which is the numbering level used once `\mainmatter` is called. You can use `\setsecnumdepth` anywhere in the `\mainmatter` to (temporarily) change the numbering level.

By default, the class sets:

```
\setsecnumdepth{section}
\maxsecnumdepth{section}
```

The `\frontmatter` commands sets the numbering level to `none`. The commands `\mainmatter` and `\mainmatter*` set the numbering level to the value specified by `\maxsecnumdepth`.

Hint. *One will often set* `\setsecnumdepth` *in the preamble and then wonder why it didn't do anything. That is probably because of* `\frontmatter` *and* `mainmatter`*, so make it a habit of setting them both like above.*

The number setting commands come from the tocvsec2 package [Wil99b].

6.4 BOOK AND PART HEADINGS

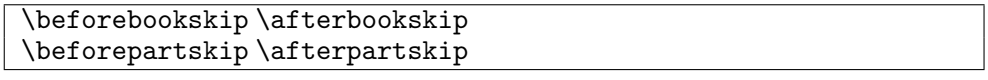
Book and part headings *always* start on a new page with the *book* and *part* pagestyles, respectively. The typical book and part heading consists of the name (e.g., 'Book' or 'Part') followed by a number represented as an uppercase Roman numeral. There is a vertical space after which the title is printed. Finally a new page is started.

Several aspects of the typesetting of the `\book` and `\part` title are configurable. Ignoring details, such as the optional argument, the code for printing `\part` headings looks like this:

```
\newcommand{\part}[1]{%    % THIS IS A VERY SIMPLIFIED VERSION
  \cleardoublepage         % start a new recto page
  \thispagestyle{part}     % set the page style
  \beforepartskip          % space before Name and Number
  \printpartname\partnamenum\printpartnum
  \midpartskip             % space after Name and Number
  \printparttitle{#1}      % print the title
  \partpageend}            % finish off
\newcommand{\partpageend}{%  THIS IS SIMPLIFIED
  \afterpartskip
  % ifblankpage then blank next page and restore twocolumn if necessary
  }
```

The code for `\book` headings is similar.

The general layout for `\book`, `\part` and `\chapter` headings is similar and you may wish to refer to Figure 6.1 which, although it shows the vertical layout for a `chapter` head, is also applicable to `\book` and `\part` heads with appropriate changes in the names of the commands.



These commands effectively control the spacing before and after the book and part titles. Their default definitions are:

```
\newcommand*{\beforebookskip}{\null\vfil}
\newcommand*{\afterbookskip}{\vfil\newpage}
\newcommand*{\beforepartskip}{\null\vfil}
\newcommand*{\afterpartskip}{\vfil\newpage}
```

Together, these vertically center any typesetting on the page, and then start a new page. To move the \part title upwards on the page, for example, you could do:

```
\renewcommand*{\beforepartskip}{\null\vskip 0pt plus 0.3fil}
\renewcommand*{\afterpartskip}{\vskip 0pt plus 0.7fil \newpage}
```

\midbookskip \midpartskip

The macros \midbookskip and \midpartskip are the spacings between the number lines and the titles. The default definitions are:

```
\newcommand{\midbookskip}{\par\vspace 2\onelineskip}
\newcommand{\midpartskip}{\par\vspace 2\onelineskip}
```

and they can be changed.

\printbookname \booknamefont \booknamenum \printbooknum \booknumfont \printpartname \partnamefont \partnamenum \printpartnum \partnumfont

The macro \printbookname typesets the book name (the value of \bookname) using the font specified by \booknamefont. The default is the \bfseries font in the \huge size. Likewise the book number is typeset by \printbooknum using the font specified by \booknumfont, which has the same default as \booknamefont. The macro \booknamenum, which is defined to be a space, is called between printing the book name and the number. All these can be changed to obtain different effects.

Similarly, the macro \printpartname typesets the part name (the value of \partname) using the font specified by \partnamefont. The default is the \bfseries font in the \huge size. Likewise the part number is typeset by \printpartnum using the font specified by \partnumfont, which has the same default as \partnamefont. The macro \partnamenum, which is defined to be a space, is called between printing the part name and the number.

For example, to set a \part in a large sans font with the part name flush left:

```
\renewcommand{\partnamefont}{\normalfont\huge\sffamily\raggedright}
\renewcommand{\partnumfont}{\normalfont\huge\sffamily}
```

or to only print the part number in the default font:

```
\renewcommand{\printpartname}{}
\renewcommand{\partnamenum}{}
```

\printbooktitle{⟨*title*⟩} \booktitlefont \printparttitle{⟨*title*⟩} \parttitlefont

A book's title is typeset by \printbooktitle using the font specified by \booktitlefont. By default this is a \bfseries font in the \Huge size. This can be changed to have, say, the title set raggedleft in a small caps font by

```
\renewcommand{\booktitlefont}{\normalfont\Huge\scshape\raggedleft}
```

Similarly a part's title is typeset by \printparttitle using the font specified by \parttitlefont. By default this is a \bfseries font in the \Huge size.

The \parttitlefont font is also used by \appendixpage, or its starred version, when typesetting an appendix page.

\bookpagemark{⟨*title*⟩} \partmark{⟨*title*⟩}

The \book code includes \bookpagemark{⟨*title*⟩} for capturing the ⟨*title*⟩ of the book division if it is going to be used, for example, in page headers. Its definition is simply:

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

There is the corresponding \partmark for the title of \part divisions.

\bookpageend \bookblankpage \nobookblankpage \partpageend \partblankpage \nopartblankpage

The macro \bookpageend finishes off a book title page. It first calls \afterbookskip. If the \nobookblankpage is in effect it does nothing more. If the declaration \bookblankpage (the default) is in effect then it finishes the current page, outputs a blank page and then, if twocolumn typesetting was in effect before \book then it restores twocolumn typesetting. The macro \partpageend performs similar functions for \part pages.

So to add something on the back side of a \part page (assuming twoside) use something similar to

```
...
\nopartblankpage
\part{Title of the Part}
\thispagestyle{simple}
Text on the following (normally blank page)
\clearpage
...
```

Alternatively you can redefine \partpageend.

If you use the declaration \nopartblankpage (or \nobookblankpage) then you are responsible for setting everything correctly to end off the \part (or \book) page. This is the default definition of \partpageend (that for \bookpageend is similar):

```
\newcommand{\partpageend}{%
  \afterpartskip
  \ifm@mnopartnewpage%   set by \(no)partblankpage
  \else%                 default finish off
    \if@twoside
      \if@openright%     output blank page
```

```
        \null
        \thispagestyle{afterpart}%
        \newpage
      \fi
    \fi
  \fi
  \if@tempswa%   true if twocolumn was being used
    \twocolumn
  \fi}
```

Here with the default definitions, `\afterpartskip` ends off the `\part` page, and then the rest of the code in `\partpageend` takes care of typesetting the blank back side of the `\part` page (or send us back to twocolumn mode).

If on the other hand we actually want to write something below the part title on the `\part` page, then we need a different route. The 'air' above and below the part title is by default defined as

```
\newcommand*{\beforepartskip}{\null\vfil}
\newcommand*{\afterpartskip}{\vfil\newpage}
```

Thus we need to redefined this such that it does not change the page and such that it add useful spacing above and below the part titling. Something like this may do the trick

```
\makeatletter
  \newcommand*{\beforepartskip}{\null\vskip4cm}
  \newcommand*{\afterpartskip}{\par\vskip1cm%
    \@afterindentfalse\@afterheading}
\makeatother
```

6.4.1 Leadpage

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

The `\newleadpage`[2] command defines a macro `\cmdname` that when called will typeset an Appendixpage-like page (see §6.2.1) with a title ⟨*title*⟩ using the ⟨*page-style*⟩ as the pagestyle for the page. The default is the *empty* pagestyle. The macro `\renewleadpage` redefines an existing leadpage command. `\cmdname` will add an entry to the TOC, the similarly defined `\cmdname*` will not.

As an example:

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

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

`\leadpagetoclevel`

When `\(re)newleadpage` is used the resulting command adds ⟨*title*⟩ to the ToC as though it was an unnumbered `\leadpagetoclevel` entry, whose definition is

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

[2]The suggestions for this came from Danie Els and Lars Madsen.

If you wished them to be entered like a \part header then simply:

```
\renewcommand*{\leadpagetoclevel}{part}
```

The layout of the page matches that for unnumbered \part pages, and internally the resulting commands use \partmark in case you wish to capture the ⟨*title*⟩ to use in running headers.

6.5 CHAPTER HEADINGS

The chapter headings are configurable in much the same way as book or part headings, but in addition there are some built in chapter styles that you may wish to try, or define your own.

Chapters, except when the article class option is used, *always* start on a new page with the *chapter* pagestyle. The particular page, recto or verso, that they start on is mainly controlled by the class options. If the oneside option is used they start on the next new page, but if the twoside option is used the starting page may differ, as follows.

openright The chapter heading is typeset on the next recto page, which may leave a blank verso leaf.

openleft The chapter heading is typeset on the next verso page, which may leave a blank recto leaf.

openany The chapter heading is typeset on the next page and there will be no blank leaf.

\openright \openleft \openany

These three declarations have the same effect as the options of the same name. They can be used anywhere in the document to change the chapter opening page.

Ignoring many details, like the optional arguments, the code for printing a \chapter heading is similar to that for \book and \part (the \chapterhead command below is *not* part of the class).

```
\newcommand{\chapterhead}[1]{ % THIS IS A SIMPLIFIED VERSION
  \clearforchapter        % move to correct page
  \thispagestyle{chapter} % set the page style
  \insertchapterspace     % Inserts space into LoF and LoT
  \chapterheadstart       % \beforechapskip space before heading
  \printchaptername\chapternamenum\printchapternum
  \afterchapternum        % \midchapskip space between number and title
  \printchaptertitle{#1}  % title
  \afterchaptertitle}     % \afterchapskip space after title
```

The general layout is shown in Figure 6.1.

\clearforchapter

The actual macro that sets the opening page for a chapter is \clearforchapter. The class options openright, openleft and openany (or their macro equivalents \openright, \openleft and \openany) define \clearforchapter to be respectively (see §18.13) \cleartorecto, \cleartoverso and \clearpage. You can obviously change \clearforchapter to do more than just start a new page.

\memendofchapterhook

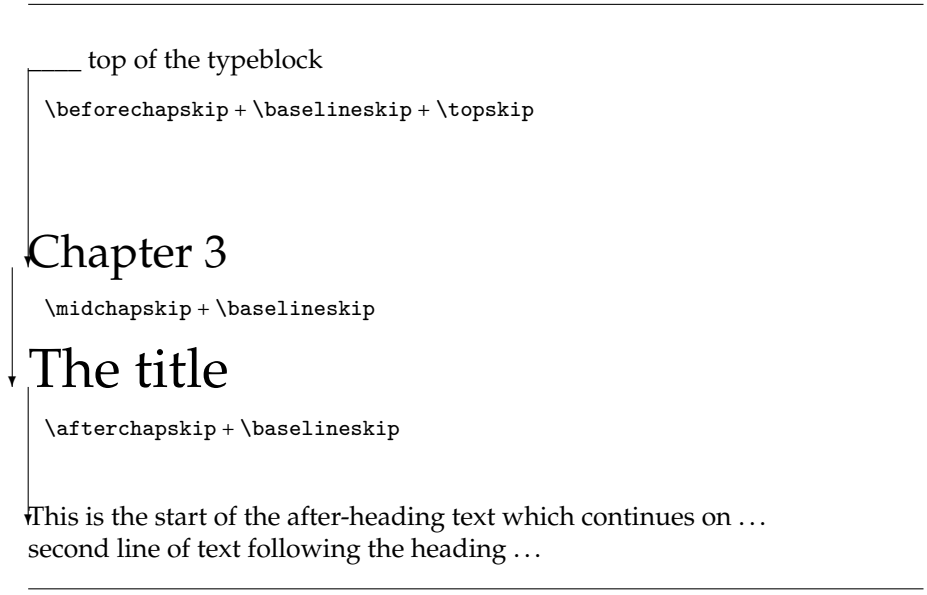


Figure 6.1: Class layout parameters for chapter titles. Working with `\beforechapskip` need a little thought, see the text.

Where `\clearforchapter` comes at the very beginning, `\memendofchapterhook` comes at the very end of the `\chapter` command. It does nothing by default, but could be redefined to insert, say, `\clearpage`:

```
\makeatletter
\renewcommand\memendofchapterhook{%
  \clearpage\m@mindentafterchapter\@afterheading}
\makeatother
```

Some books have the chapter headings on a verso page, with perhaps an illustration or epigraph, and then the text starts on the opposite recto page. The effect can be achieved like:

```
\openleft               % chapter title on verso page
\chapter{The title}     % chapter title
\begin{centering}       % include a centered illustration
\includegraphics{...}
\end{centering}
\clearpage              % go to recto page
Start of the text       % chapter body
```

`\chapterheadstart` `\beforechapskip`
`\afterchapternum` `\midchapskip`
`\afterchaptertitle` `\afterchapskip`

The macro \chapterheadstart is called just before printing a chapter name and number. By default it inserts \beforechapskip space (default 50pt).

The macro \afterchapternum is called just after printing a chapter number. By default it inserts \midchapskip space (default 20pt).

The macro \afterchaptertitle is called just after printing a chapter title. By default it inserts \afterchapskip space (default 40pt).

The lengths \beforechapskip, \midchapskip and \afterchapskip can all be changed by \setlength or \addtolength. Though as mentioned in Figure 6.1 they need some explanation:

\beforechapskip
: See Figure 6.1. The actual distance between the first baseline of the chapter stuff to the top of the text block is \beforechapskip + \topskip + \baselineskip. But because the implementation of \chapter (via \chapterheadstart) make use of \vspace*, getting rid of \beforechapskip a strange endeavour. If you want to avoid any space before the first text in the chapter heading, use

```
\setlength\beforechapskip{-\baselineskip}
```

or redefine \chapterheadstart to do nothing.

\midchapskip

\afterchapskip
: for both, one has to add \baselineskip to get the distance baseline to baseline.

\printchaptername \chapnamefont \chapternamenum \printchapternum \chapnumfont

The macro \printchaptername typesets the chapter name (default Chapter or Appendix) using the font specified by \chapnamefont. The default is the \bfseries font in the \huge size. Likewise the chapter number is typeset by \printchapternum using the font specified by \chapnumfont, which has the same default as \chapnamefont. The macro \chapternamenum, which is defined to be a space, is called between printing the chapter name and the number.

\printchaptertitle{⟨*title*⟩} \chaptitlefont

The title is typeset by \printchaptertitle using the font specified by \chaptitlefont. By default this is a \bfseries font in the \Huge size.

\printchapternonum

If a chapter is unnumbered, perhaps because it is in the \frontmatter or because \chapter* is used, then when printing the command \printchapternonum is called instead of printing the name and number, as illustrated below:

```
\newcommand{\chapterhead}[1]{ % THIS IS A SIMPLIFIED VERSION
  \clearforchapter        % move to correct page
  \thispagestyle{chapter} % set the page style
  \insertchapterspace     % Inserts space into LoF and LoT
  \chapterheadstart       % \beforechapskip space before heading
  \printchaptername\chapternamenum\printchapternum
```

```
\afterchapternum        % \midchapskip space between number and title
\printchaptertitle{#1}  % title
\afterchaptertitle}     % \afterchapskip space after title
```

By default the first paragraph following a `\chapter` is *not* indented, this can be controlled by

```
\indentafterchapter
\noindentafterchapter
```

The default is not to indent the first paragraph following a `\chapter`.

```
\insertchapterspace
```

By default a `\chapter` inserts a small amount of vertical space into the List of Figures and List of Tables. It calls `\insertchapterspace` to do this. The default definition is:

```
\newcommand{\insertchapterspace}{%
  \addtocontents{lof}{\protect\addvspace{10pt}}%
  \addtocontents{lot}{\protect\addvspace{10pt}}%
}
```

If you would prefer no inserted spaces then

```
\renewcommand{\insertchapterspace}{}
```

will do the job. Different spacing can be inserted by changing the value of the length arguments to `\addvspace`.

By making suitable changes to the above macros you can make some simple modifications to the layout.

6.5.1 Defining a chapter style

The class provides many ways in which you can implement your designs for chapter headings.

```
\chapterstyle{⟨style⟩}
```

The macro `\chapterstyle` is rather like the `\pagestyle` command in that it sets the style of any subsequent chapter headings to be ⟨*style*⟩.

The class provides some predefined chapter styles, including the *default* style which is the familiar LaTeX book class chapter headings style. To use the chapterstyle *fred* just issue the command

```
\chapterstyle{fred}
```

Different styles can be used in the same document.

The simpler predefined styles include:

default The normal LaTeX book class chapter styling; shown in Figure B.1.

section The heading is typeset like a section; that is, there is just the number and the title on one line. This is illustrated in Figure B.2.

hangnum Like the *section* style except that the chapter number is put in the margin, as shown in Figure B.3.

companion This produces chapter headings like those of the *LaTeX Companion* series of books. An example is in Figure B.4.

article The heading is typeset like a `\section` heading in the article class. This is similar to the *section* style but different fonts and spacings are used, as shown in Figure B.5.

reparticle When the article class option is used the default chapter and section styles are close, but not identical, to the corresponding division heads in the article class. The *reparticle* chapterstyle makes `\chapter` replicate the appearance of `\section` in the article class.

If you use only the predefined chapterstyles there is no need to plough through the rest of this section, except to look at the illustrations of the remaining predefined chapterstyles shown a little later.

The various macros shown in the `\chapterhead` code above are initially set up as (the code is known as the *default* style):

```
\newcommand{\chapterheadstart}{\vspace*{\beforechapskip}}
\newcommand{\printchaptername}{\chapnamefont \@chapapp}
\newcommand{\chapternamenum}{\space}
\newcommand{\printchapternum}{\chapnumfont \thechapter}
\newcommand{\afterchapternum}{\par\nobreak\vskip \midchapskip}
\newcommand{\printchapternonum}{}
\newcommand{\printchaptertitle}[1]{\chaptitlefont #1}
\newcommand{\afterchaptertitle}{\par\nobreak\vskip \afterchapskip}
\newcommand{\chapnamefont}{\normalfont\huge\bfseries}
\newcommand{\chapnumfont}{\normalfont\huge\bfseries}
\newcommand{\chaptitlefont}{\normalfont\Huge\bfseries}
\setlength{\beforechapskip}{50pt}
\setlength{\midchapskip}{20pt}
\setlength{\afterchapskip}{40pt}
```

(The mysterious `\@chapapp` is the internal macro that LaTeX uses to store normally the chapter name.[3] It will normally have different values, set automatically, when typesetting a chapter in the main body (e.g., `Chapter`) or in the appendices where it would usually be set to `Appendix`, but you can specify these names yourself.)

A new style is specified by changing the definitions of this last set of macros and/or the various font and skip specifications.

`\makechapterstyle{`⟨*style*⟩`}{`⟨*text*⟩`}`

Chapter styles are defined via the `\makechapterstyle` command, where ⟨*style*⟩ is the style being defined and ⟨*text*⟩ is the LaTeX code defining the style. Please note that the ⟨*text*⟩ *always* start by resetting macros and lengths to the values shown above. That is what you actually get is is the same as

`\makechapterstyle{`⟨*style*⟩`}{`
`\chapterstyle{default}`
⟨*text*⟩
}

To start things off, the *default* chapter style, which mimics the chapter heads in the standard book and report classes, as it appears in `memoir.cls`:

[3] Remember, if you use a macro that has an `@` in its name it must be in a place where `@` is treated as a letter.

```
\makechapterstyle{default}{}
\chapterstyle{default}
```

– since the *default* style *is* the initial value of `\makechapterstyle`. The actual code is seen above.

As an example of setting up a simple chapterstyle, here is the code for defining the *section* chapterstyle. In this case it is principally a question of eliminating most of the printing and zeroing some spacing.

```
\makechapterstyle{section}{%
  \renewcommand*{\printchaptername}{}
  \renewcommand*{\chapternamenum}{}
  \renewcommand*{\chapnumfont}{\chaptitlefont}
  \renewcommand*{\printchapternum}{\chapnumfont \thechapter\space}
  \renewcommand*{\afterchapternum}{}
}
```

In this style, `\printchaptername` is vacuous, so the normal 'Chapter' is never typeset. The same font is used for the number and the title, and the number is typeset with a space after it. The macro `\afterchapternum` is vacuous, so the chapter title will be typeset immediately after the number.

In the standard classes the title of an unnumbered chapter is typeset at the same position on the page as the word 'Chapter' for numbered chapters. The macro `\printchapternonum` is called just before an unnumbered chapter title text is typeset. By default this does nothing but you can use `\renewcommand` to change this. For example, if you wished the title text for both numbered and unnumbered chapters to be at the same height on the page then you could redefine `\printchapternonum` to insert the amount of vertical space taken by any 'Chapter N' line. For example, as `\printchapternonum` is vaucuous in the *default* chapterstyle the vertical position of a title depends on whether or not it is numbered.

The *hangnum* style, which is like *section* except that it puts the number in the margin, is defined as follows:

```
\makechapterstyle{hangnum}{%
  \renewcommand*{\chapnumfont}{\chaptitlefont}
  % allow for 99 chapters!
  \settowidth{\chapindent}{\chapnumfont 999}
  \renewcommand*{\printchaptername}{}
  \renewcommand*{\chapternamenum}{}
  \renewcommand*{\chapnumfont}{\chaptitlefont}
  \renewcommand*{\printchapternum}{%
    \noindent\llap{\makebox[\chapindent][l]{%
    \chapnumfont \thechapter}}}
  \renewcommand*{\afterchapternum}{}
}
```

The chapter number is put at the left of a box wide enough for three digits. The box is put into the margin, via `\llap`, for typesetting. The chapter title is then typeset, starting at the left margin.

`\chapindent`

The length `\chapindent` is provided for use in specifying chapterstyles, but you could use it for any other purposes.

The definition of the *companion* chapterstyle is more complicated.

```
\makechapterstyle{companion}{%
  \renewcommand*{\chapnamefont}{\normalfont\LARGE\scshape}
  \renewcommand*{\chapnumfont}{\normalfont\Huge}
  \renewcommand*{\printchaptername}{%
    \raggedleft\chapnamefont \@chapapp}
  \setlength{\chapindent}{\marginparsep}
  \addtolength{\chapindent}{\marginparwidth}
  \renewcommand{\printchaptertitle}[1]{%
    \begin{adjustwidth}{}{-\chapindent}
      \raggedleft \chaptitlefont ##1\par\nobreak
    \end{adjustwidth}}
}
```

As shown in Figure B.4 the chapter name is in small caps and set flushright. The title is also set flushright aligned with the outermost part of the marginal notes. This is achieved by use of the `adjustwidth` environment[4] to make LaTeX think that the typeblock is locally wider than it actually is.

6.5.2 Further chapterstyles

The class provides more chapterstyles, which are listed here. Some are mine and others are from postings to CTT by memoir users. I have modified some of the posted ones to cater for things like appendices, multiline titles, and unnumbered chapters which were not considered in the originals. The code for some of them is given later to help you see how they are done. Separately, Lars Madsen has collected a wide variety of styles [Mad06] and shows how they were created.

If you want to try several chapterstyles in one document, request the *default* style before each of the others to ensure that a previous style's changes are not passed on to a following one.

bianchi This style was created by Stefano Bianchi[5] and is a two line centered arrangement with rules above and below the large bold sanserif title line. The chapter number line is in a smaller italic font. An example is in Figure B.6.

bringhurst The *bringhurst* chapterstyle described in the manual and illustrated in Figure B.7.

brotherton A very simple style designed by William Adams[6] for the science fiction novel *Star Dragon* by Mike Brotherton. The novel is freely downloadable from Brotherton's web site. The style is the same as the *default* except that the number is spelt out in words. It is demonstrated in Figure B.8. In the novel the chapters are actually untitled i.e., via `\chapter{}`.

chappell The name and number are centered above a rule and the title in italics is below, again centered. It is illustrated in Figure B.9.

[4] See §8.5.

[5] CTT, *New chapter style: chapter vs chapter**, 2003/12/09

[6] CTT, *An example of a novel?*, 2006/12/09

crosshead The number and title are centered and set with a large bold font. It is illustrated in Figure B.10.

culver A chapter style I created for Christopher Culver[7] based on the format of 'ancient' texts. It is one line, centered, bold and with the number printed as Roman numerals, as shown in Figure B.11.

He also wanted sections to just start with the number and the text to immediately follow on the same line. That can be accomplished like this:

```
\renewcommand*{\thesection}{\arabic{section}}
\renewcommand*{\section}[1]{%
  \refstepcounter{section}%
  \par\noindent
  \textbf{\thesection.}%
  \space\nolinebreak}
```

dash A simple two line centered chapterstyle. There is a short dash on either side of the number and a slightly larger version of the regular font is used for both the number and the title. This style is shown in Figure B.12.

demo2 A two line chapterstyle with a large sanserif title; the number is above, centered and written (e.g., Six instead of 6) in a bold font. There are rules above and below the title. An example is shown in Figure B.13.

demo3 The chapterstyle used in this document. It is a modified version of the *demo2* chapterstyle, using an italic rather than bold font for the number.

dowding A centered style where the name and number are set in a bold font, with the number spelled out. The title is below in a large italic font. The style is based on the design used in Dowding's *Finer Points* [Dow96]. It is illustrated in Figure B.14.

ell A raggedleft sanserif chapterstyle. The number line is separated from the title by rules like an 'L' on its side and the number is placed in the margin, as shown in Figure B.15. I will probably use this in my next book.

ger This style was created by Gerardo Garcia[8] and is a two line, raggedright, large bold style with rules above and below. It is demonstrated in Figure B.16.

komalike A section-like style set with a sans serif type. It is like that in the scrbook class (part of the KOMA bundle). It is illustrated in Figure B.17.

lyhne A style created by Anders Lyhne[9] and shown in Figure B.18 where the raggedleft sanserif title is between two rules, with the name and number above. I modified the original to cater for unnumbered chapters.

Caveat: The *lyhne* style requires the graphicx package.

madsen This was created by Lars Madsen[10] and is shown in Figure B.19. It is a large sanserif raggedleft style with the number in the margin and a rule between the number and title lines.

Caveat: The *madsen* style requires the graphicx package.

[7] CTT, *"Biblical" formatting, how?*, 2004/03/29
[8] CTT, *Fancy Headings, Chapter Headings*, 2002/04/12
[9] CTT, *Glossary*, 2006/02/09
[10] CTT, *New chapter style: chapter vs chapter**, 2003/12/09

ntglike A smaller version of the standard chapterstyle; it is like that in the NTG classes (boek class) developed by the Dutch TeX User Group. It is illustrated in Figure B.20.

pedersen This was created by Troels Pedersen[11] and requires the graphicx package, and, to get the full effect, the color package as well. The title is raggedright in large italics while the number is much larger and placed in the righthand margin (I changed the means of placing the number). The head of this chapter is set with the *pedersen* style, because it cannot be adequately demonstrated in an illustration.

Caveat: The *pedersen* style requires the graphicx package.

southall This style was created by Thomas Dye. It is a simple numbered heading with the title set as a block paragraph, and with a horizontal rule underneath. It is illustrated in Figure B.21.

tandh A simple section-like style in a bold font. It is based on the design used in the Thames & Hudson *Manual of Typography* [McL80] and is illustrated in Figure B.22.

thatcher A style created by Scott Thatcher[12] which has the chapter name and number centered with the title below, also centered, and all set in small caps. There is a short rule between the number line and the title, as shown in Figure B.23. I have modified the original to cater for multiline titles, unnumbered chapters, and appendices.

veelo This style created by Bastiaan Veelo is shown in Figure B.24 and is raggedleft, large, bold and with a black square in the margin by the number line.

Caveat: The *veelo* style requires the graphicx package.

verville A chapterstyle I created for Guy Verville[13]. It is a single line, large centered style with rules above and below, as in Figure B.25. Unlike my posted version, this one properly caters for unnumbered chapters.

wilsondob A one line flushright (raggedleft) section-like style in a large italic font. It is based on the design used in Adrian Wilson's *The Design of Books* [Wil93] and is illustrated in Figure B.26.

The code for some of these styles is given in section B.1 within the Showcase Appendix. For details of how the other chapter styles are defined, look at the documented class code. This should give you ideas if you want to define your own style.

Note that it is not necessary to define a new chapterstyle if you want to change the chapter headings — you can just change the individual macros without putting them into a style.

6.5.3 Chapter precis

Some old style novels, and even some modern text books,[14] include a short synopsis of the contents of the chapter either immediately after the chapter heading or in the ToC, or in both places.

`\chapterprecis{`⟨*text*⟩`}`

[11] CTT, *Chapter style*, 2006/01/31
[12] CTT, *memoir: chapter headings capitalize math symbols*, 2006/01/18
[13] CTT, *Headers and special formatting of sections*, 2005/01/18
[14] For example, Robert Sedgewick, *Algorithms*, Addison-Wesley, 1983.

The command \chapterprecis prints its argument both at the point in the document where it is called, and also adds it to the .toc file. For example:

```
...
\chapter{}%  first chapter
\chapterprecis{Our hero is introduced; family tree; early days.}
...
```

Now for the details.

\prechapterprecisshift

The length \prechapterprecisshift controls the vertical spacing before a \chapterprecis. If the precis immediately follows a \chapter then a different space is required depending on whether or not the article class option is used. The class sets:

```
\ifartopt
  \setlength{\prechapterprecisshift}{0pt}
\else
  \setlength{\prechapterprecisshift}{-2\baselineskip}
\fi
```

\chapterprecishere{⟨*text*⟩}
\chapterprecistoc{⟨*text*⟩}

The \chapterprecis command calls these two commands to print the ⟨*text*⟩ in the document (the \chapterprecishere command) and to put it into the ToC (the \chapterprecistoc command). These can be used individually if required.

\precisfont
\prechapterprecis \postchapterprecis

The \chapterprecishere macro is intended for use immediately after a \chapter. The ⟨*text*⟩ argument is typeset in the \precisfont font in a quote environment. The macro's definition is:

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

where \prechapterprecis, \postchapterprecis and \precisfont are defined as:

```
\newcommand{\prechapterprecis}{%
  \vspace*{\prechapterprecisshift}%
  \begin{quote}\precisfont}
\newcommand{\postchapterprecis}{\end{quote}}
\newcommand*{\precisfont}{\normalfont\itshape}
```

Any or all of these can be changed if another style of typesetting is required.

Next the following macros control the formatting of the precis text in the ToC.

\precistoctext{⟨*text*⟩} \precistocfont \precistocformat

The \chapterprecistoc macro puts \precistoctext{⟨*text*⟩} into the toc file. The default definition similar to (but not exactly[15])

[15]Internally we use a different name for \leftskip and \rightskip to make it easier to do right to left documents with the class and the bidi package.

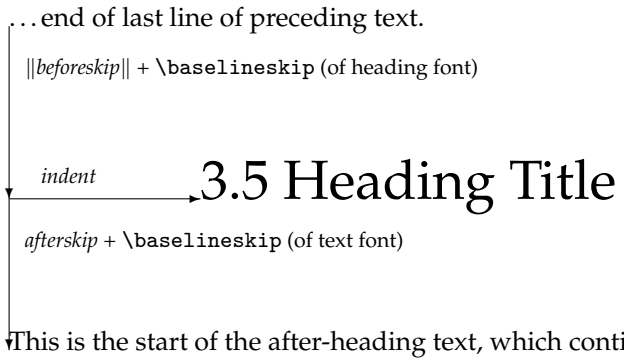


Figure 6.2: Displayed sectional headings

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

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

6.6 LOWER LEVEL HEADINGS

The lower level heads — sections down to subparagraphs — are also configurable, but there is nothing corresponding to chapter styles.

There are essentially three things that may be adjusted for these heads: (a) the vertical distance between the baseline of the text above the heading to the baseline of the title text, (b) the indentation of the heading from the left hand margin, and (c) the style (font) used for the heading title. Additionally, a heading may be run-in to the text or as a display before the following text; in the latter case the vertical distance between the heading and the following text may also be adjusted. Figure 6.2 shows the parameters controlling a displayed sectional heading and Figure 6.3 shows the parameters for a run-in heading. The default values of the parameters for the different heads are in Table 6.2 for the display heads and Table 6.3 for the run-in heads.

In the following I will use `S` to stand for one of `sec`, `subsec`, `subsubsec`, `para` or `subpara`, which are in turn shorthand for `section` through to `subparagraph`, as summarised in Table 6.4.

`\setbeforeSskip{`⟨*skip*⟩`}`

The absolute value of the ⟨*skip*⟩ length argument is the space to leave above the heading. If the actual value is negative then the first line after the heading will not be indented. The

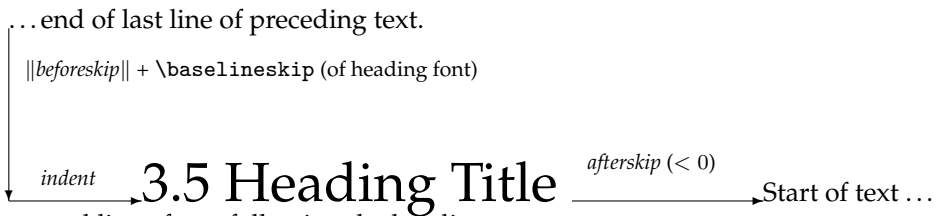


Figure 6.3: Run-in sectional headings

Table 6.2: Default display sectioning layout parameter values

	section	subsection	subsubsection
beforeskip (-ex)	3.5+1-.2	3.25+1-.2	3.25+1-.2
indent	0	0	0
afterskip (ex)	2.3+.2	1.5+.2	1.5+.2
font	`\Large\bfseries`	`\large\bfseries`	`\bfseries`

Table 6.3: Default run-in sectioning layout parameter values

	paragraph	subparagraph
beforeskip (+ex)	3.25+1-.2	3.25+1-.2
indent	0	`\parindent`
afterskip	-1em	-1em
font	`\bfseries`	`\bfseries`

Table 6.4: Values for `S` in section styling macro names.

`S`	`sec`	`subsec`	`subsubsec`	`para`	`subpara`
	`section`	`subsection`	`subsubsection`	`paragraph`	`subparagraph`

default ⟨*skip*⟩ depends on the particular level of heading, but for a `\section` (i.e., when `S = sec`) it is

```
-3.5ex plus -1ex minus -.2ex
```

where the plus and minus values are the allowable stretch and shrink; note that all the values are negative so that there is no indentation of the following text. If you wanted indentation then you could do

```
\setbeforesecskip{3.5ex plus 1ex minus .2ex}
```

`\setSindent{`⟨*length*⟩`}`

The value of the ⟨*length*⟩ length argument is the indentation of the heading (number and title) from the lefthand margin. This is normally 0pt.

`\setSheadstyle{`⟨*font*⟩`}`

This macro specifies the style (font) for the sectional number and title. As before, the default value of the ⟨*font*⟩ argument depends on the level of the heading. For a `\subsection` (i.e., `S=subsec`) it is `\large\bfseries\raggedright`, to typeset in the `\bfseries` font in the `\large` size; the title will also be set ragged right (i.e., there will be no hyphenation in a multiline title).

Note that the very last element in the ⟨*font*⟩ argument may be a macro that takes one argument (the number and title of the heading). So, if for some reason you wanted `\subsubsection` titles to be all uppercase, centered, and in the normal font, you can do

```
\setsubsubsecheadstyle{\normalfont\centering\MakeUppercase}
```

* * *

Under very special circumstances, one might want to get completely rid of the sectional header (for example, if you are using a very small paper size, and you are placing the section information in the header instead), by creating a macro that takes on argument and ignores it

```
\newcommand\ignoreMe[1]{}
\setsecheadstyle{\ignoreMe}
```

* * *

As another example, although I don't recommend this, you can draw a horizontal line under section titles via:

```
\newcommand{\ruledsec}[1]{%
  \Large\bfseries\raggedright #1 \rule{\textwidth}{0.4pt}}
\setsecheadstyle{\ruledsec}
```

`\setafterSskip{`⟨*skip*⟩`}`

If the value of the ⟨*skip*⟩ length argument is positive it is the space to leave between the display heading and the following text. If it is negative, then the heading will be run-in and the value is the horizontal space between the end of the heading and the following text. The default ⟨*skip*⟩ depends on the particular level of heading, but for a `\section` (i.e., when `S = sec`) it is `2.3ex plus .2ex`, and for a `\subparagraph` (i.e., `S = subpara`), which is a run-in heading, it is `-1em`.

*

\@hangfrom{⟨*code*⟩} \sethangfrom{⟨*code*⟩}

Internally all the titling macros use a macro called \@hangfrom which by default makes multiline titles look like a hanging paragraph. The default definition of \@hangfrom (in file ltsect.dtx) is effectively:

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

The argument is put into a box and its width is measured, then a hanging paragraph is started with the argument as the first thing and second and later lines indented by the argument's width.

The \sethangfrom macro redefines \@hangfrom to be ⟨*code*⟩. For example, to have the titles set as block paragraphs instead of hanging paragraphs, simply do:

```
\sethangfrom{\noindent #1}
```

Note that you have to use #1 at the position in the replacement code for \@hangfrom where the argument to \@hangfrom is to be located.

\@seccntformat{⟨*code*⟩} \setsecnumformat{⟨*code*⟩}

Internally all the titling macros like \section, \subsection etc. use a kernel macro called \@seccntformat which defines the formatting of sectional numbers in a title. Its default definition (in file ltsect.dtx) is effectively:[16]

```
\newcommand{\@seccntformat}[1]{\csname the#1\endcsname\quad}
```

which formats the sectional numbers as \thesec... with a space afterwards. The command \setsecnumformat redefines \@seccntformat to be ⟨*code*⟩. For example, to put a colon and space after the number

```
\setsecnumformat{\csname the#1\endcsname:\quad}
```

Note that you have to use #1 where you want the argument (sectional number) of \@seccntformat to go.

Note that \setsecnumformat applies to all \section, \subsection, etc. If you want to change it only for, say, \subsection, use the \setsubsechook described below.

*

\hangsecnum \defaultsecnum

The macro \hangsecnum is a declaration that makes sectional numbers hang in the margin. The macro \defaultsecnum is a declaration that reverses the effect of \hangsecnum, that is, sectional numbers will be typeset in their familiar places.

\Shook \setShook{⟨*text*⟩}

[16] Because they all internally use the \@startsection macro.

The macro \Shook is called immediately before the typesetting of the title; its default definition does nothing. The macro \setShook redefines \Shook to be ⟨*text*⟩. You can use this hook, for example, to insert a \sethangfrom or \setsecnumformat command into the definition of a particular section division command. In that case, remember that if you want to refer to the #1 argument, in the argument for \setsecnumformat, then you have to double the #, i.e. use ##1, see the example below.

Here are some example lower level heads and the code used to produce them.

Source for example 6.1

```
\setsubsubsecheadstyle{\bfseries\raggedright}
    \subsubsection*{Bold raggedright}
\setsubsubsecheadstyle{\scshape\raggedright}
    \subsubsection*{Small caps raggedright}
\setsubsubsecheadstyle{\itshape\raggedright}
    \subsubsection*{Italic raggedright}
\setsubsubsecheadstyle{\Large\centering}
    \subsubsection*{Large centered}
\setsubsubsecheadstyle{\large\centering\MakeUppercase}
    \subsubsection*{large centered uppercase}
\setsubsubsecheadstyle{\bfseries\centering}
    \subsubsection*{Bold centered}
\setsubsubsecheadstyle{\scshape\centering}
    \subsubsection*{Small caps centered}
\setsubsubsecindent{2\parindent}
\setsubsubsecheadstyle{\scshape\raggedright}
    \subsubsection*{Small caps indented}
\setsubsubsecindent{0pt}
\setsubsubsecheadstyle{\itshape\raggedleft}
    \subsubsection*{Italic flushright}
\newcommand*{\shortcenter}[1]{%
  \sethangfrom{\noindent ##1}%
  \normalfont\boldmath\bfseries
  \centering
  \parbox{3in}{\centering #1}\par}
\setsubsubsecheadstyle{\shortcenter}
\subsubsection*{Bold centered but taking up no more than 3 inches
                if a long title}
```

Hang the whole heading in the margin

A less traditional style is to put the whole heading into the margin. I have done this here for a \paragraph heading (which is not otherwise used in this manual). The code is:

```
\newcommand{\marginbox}[1]{%
  \parbox[t][0pt]{6em}{\itshape\raggedleft\leavevmode #1}}
\newcommand{\marginhead}[1]{%
  {\llap{\marginbox{#1}\kern0.5em}}}
\setparaindent{0em}
```

Typeset example 6.1: A variety of subhead styles

Bold raggedright

SMALL CAPS RAGGEDRIGHT

Italic raggedright

Large centered

LARGE CENTERED UPPERCASE

Bold centered

SMALL CAPS CENTERED

SMALL CAPS INDENTED

Italic flushright

Bold centered but taking up no more than 3 inches if a long title

```
\setafterparaskip{0em}
\setparaheadstyle{\marginhead}
\setparahook{\setsecnumformat{\csname the##1\endcsname\ }}
\paragraph{Hang the whole heading in the margin}%
```

The macro `\marginbox` puts its argument, raggedleft, into a zero height `\parbox` of width 6em, aligned at the top. The `\marginhead` macro puts its argument into a `\marginbox` and puts the `\marginbox` 0.5em to the left. The `\paragraph` head style is then set to use `\marginhead` to typeset the heading. The format for the number is reset via `\setparahook` and `\setsecnumformat`.

*

A different approach is to create new macros, each named by the type of sectional macro it formats, and then make the number format call these macros. In this example we will provide separate formatting for `\section` and `\subsection`.

```
\setsecnumformat{\csname #1secnumformat\endcsname}
\newcommand\sectionsecnumformat{\thesection:\quad}
\newcommand\subsectionsecnumformat{\fbox{\enspace\thesubsection\enspace}\enspace}
```

Since the macro is only called in the proper context, we can use `\thesection` directly in the code for `\section`. However, this requires you to define formatting macros for all sectional headings. Here is how to reverse it, that is see if a format exists, use it, otherwise use a default:

```
\makeatletter
\setsecnumformat{%
  \@ifundefined{#1secnumformat}{% nope, use a default
    \csname the#1\endcsname\quad%
```

```
    }{%
      \csname #1secnumformat\endcsname%
    }%
  }
\makeatother
```

6.7 FANCY ANONYMOUS BREAKS

Often, in novels, there is a need to break up the text to indicate that there is a major break in the story, but not enough to warrant starting a new chapter. I have called these *anonymous divisions* as there is neither number nor title associated with them.

`\plainbreak{`⟨*num*⟩`}` `\plainbreak*{`⟨*num*⟩`}`
`\fancybreak{`⟨*text*⟩`}` `\fancybreak*{`⟨*text*⟩`}`

The `\plainbreak` is an anonymous division. It puts ⟨*num*⟩ blank lines into the typescript and the first line of the following paragraph is not indented. Another anonymous division is `\fancybreak` which puts ⟨*text*⟩ centered into the typescript and the initial line of the following paragraph is not indented. For example:

```
\fancybreak{{*}\\{* * *}\\{*}}
```

typesets a little diamond made of asterisks.

The starred versions of the commands indent the first line of the following paragraph.

`\plainfancybreak{`⟨*space*⟩`}{`⟨*num*⟩`}{`⟨*text*⟩`}`
`\plainfancybreak*{`⟨*space*⟩`}{`⟨*num*⟩`}{`⟨*text*⟩`}`

If a plain break comes at the top or bottom of a page then it is very difficult for a reader to discern that there is a break at all. If there is text on the page and enough space left to put some text after a break the `\plainfancybreak` command will use a `\plainbreak` with ⟨*num*⟩ lines, otherwise (the break would come at the top or bottom of the page) it will use a `\fancybreak` with ⟨*text*⟩. The ⟨*space*⟩ argument is a length specifying the space needed for the ⟨*num*⟩ blank lines and some number of text lines for after the plain break. The starred version of the command uses the starred versions of the `\plainbreak` and `\fancybreak` commands.

Unfortunately there is an interaction between the requested, plain, and fancy break spaces. Let P be the space (in lines) required for the plain break, F the space (in lines) required for the fancy break, and S the ⟨*space*⟩ argument (in lines). From some experiments it appears that the condition for the plain break to avoid the top and bottom of the page is that $S - P > 1$. Also, the condition for the fancy break to avoid being put in the middle of a page (i.e., not at the top or bottom) is that $S - F < 3$. For example, if the plain and fancy breaks take the same vertical space then $S = P + 2$ is the only value that matches the conditions. In general, if $F = P + n$ then the condition is $1 < S - P < 3 + n$, which means that for the `\plainfancybreak` command the fancy break must always take at least as much space as the plain break.

* * *

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

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

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

```
\newcommand*{\pfbreakdisplay}{*\quad*\quad*}
```

typesets three asterisks, as shown a few lines before this.

♣ ♢ ♣

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

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

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

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

❧ ❧ ❧

The fancy break using fleurons just before this paragraph was produced by:

```
\renewcommand{\pfbreakdisplay}{%
  \ding{167}\quad\ding{167}\quad\ding{167}}
\fancybreak{\pfbreakdisplay}
```

where the \ding command is from the pifont package.

◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇

The fancy break made with fleurons was simple to specify. There are many other symbols that you can use in LaTeX and these can be combined in potentially attractive ways to produce a fancy break like the one just above.

The following idea was originally suggested by Christina Thiele [Thi98], and can be used to string together mathematical symbols. It works following the same principles as the dot leaders in the Table of Contents.

Define a macro called with the syntax \motif{⟨*shape*⟩}, where ⟨*shape*⟩ is a symbol or other shape to be repeated in a chain,

```
\newcommand{\motif}[1]{\cleaders\hbox{#1}\hfil}
```

The definition of \motif is basically taken from TeX, and is part of the code for making things like dot leaders. \hbox{⟨*stuff*⟩} puts ⟨*stuff*⟩ into a horizontal box, and \cleaders⟨*box*⟩ fills a specified amount of space using whatever number of copies of the ⟨*box*⟩ as is needed; if there is too much space to be filled by a whole number of boxes, the spare space is spread around equally. \hfil is stretchy space. The \motif macro essentially says, fill up a space with with copies of ⟨*shape*⟩.

We also need another macro, \chain{⟨*shape*⟩}{⟨*length*⟩}, where {⟨*shape*⟩} is a shape to be repeated as many times as it takes to fill up a distance ⟨*length*⟩.

```
\newcommand{\chain}[2]{\leavevmode\hbox to #2{\motif{#1}}}
```

The `\leavevmode` makes sure that we are typesetting horizontally, and `\hbox to <length>{stuff}` puts ⟨*stuff*⟩ into a horizontal box with the fixed length of ⟨*length*⟩. Roughly, what `\chain` and `\motif` do together is typeset enough copies of ⟨*shape*⟩ to make up a distance ⟨*length*⟩.

That is what we have been aiming for. All that remains is to decide on what shape we might want to use. Here is one consisting of diamonds.

```
\makeatletter
\newcommand{\diamonds}{\m@th$\mkern-.6mu \diamond \mkern-.6mu$}
\makeatother
```

The `\diamond` symbol can only be used in math mode, hence it is surrounded by the shorthand `$...$`. TeX usually leaves a little space around maths but the `\m@th` command stops that. `\mkern` adjusts space in math mode, and in this case we are eliminating the spaces[17] that would normally be on either side of the diamond symbol. The whole effect gives us a diamond symbol with zero space around it.

The fancy break at the start of this discussion was typeset by

```
% define \motif, \chain, \diamonds and then
\fancybreak{\chain{\diamonds}{0.25\textwidth}}
```

The code is not part of the memoir class; I defined it just as indicated in the body of the book. It would more naturally go into the preamble or a package. You might like to try specifying your own pattern, say `\clubs`, using the `\club` math symbol but leaving some space between them.

6.8 FOOTNOTES IN DIVISION HEADINGS

With the sectioning commands the text of the required argument ⟨*title*⟩ is used as the text for the section title in the body of the document.

When the optional argument ⟨*toc-title*⟩ is used in a sectioning command it is moving and any fragile commands must be `\protected`, while the ⟨*title*⟩ argument is fixed. The ⟨*toc-title*⟩ also serves double duty:

1. It is used as the text of the title in the ToC;
2. It is used as the text in page headers.

If the optional argument is not present, then the ⟨*title*⟩ is moving and serves the triple duty of providing the text for the body and ToC titles and for page headers.

Some folk feel an urge to add a footnote to a sectioning title, which should be resisted. If their flesh is weak, then the optional argument must be used and the `\footnote` attached to the required argument only. If the optional argument is not used then the footnote mark and text is likely to be scattered all over the place, on the section page, in the ToC, on any page that includes ⟨*title*⟩ in its headers. This is unacceptable to any reader. So, a footnoted title should look like this:

```
\chapter[Title]{Title\footnote{Do you really have to do this?}}
```

[17] It is usually a matter for experiment to find the right values for the kerning.

Table 6.5: Default fonts for sectional headings

\booknamefont	\huge\bfseries	**huge**
\booknumfont	\huge\bfseries	**huge**
\booktitlefont	\Huge\bfseries	**Huge**
\partnamefont	\huge\bfseries	**huge**
\partnumfont	\huge\bfseries	**huge**
\parttitlefont	\Huge\bfseries	**Huge**
\chapnamefont	\huge\bfseries	**huge**
\chapnumfont	\huge\bfseries	**huge**
\chaptitlefont	\Huge\bfseries	**Huge**
\secheadstyle	\Large\bfseries	**Large**
\subsecheadstyle	\large\bfseries	**Large**
\subsubsecheadstyle	\normalsize\bfseries	**normal**
\paraheadstyle	\normalsize\bfseries	**normal**
\subparaheadstyle	\normalsize\bfseries	**normal**

6.9 Predefined heading styles

All LaTeX classes for typesetting books and reports provide a particular style for sectional headings. The memoir class is unusual in that it provides several sets of heading styles. Each set has different spacing around the division heads, and different fonts in different sizes. As a reference, Table 6.5 lists the default fonts used for the sectional headings. These fonts are all bold but in different sizes depending on the division level.

```
\makeheadstyles{⟨name⟩}{⟨code⟩}
\headstyles{⟨name⟩}
```

The default sectional division head styles provided by memoir form the *default* headstyles and give the same appearance as the standard book and report classes. The set is created via the `\makeheadstyles` macro and called for via the `headstyles` declaration.

```
\makeheadstyles{default}{%
  \renewcommand*{\booknamefont}{\normalfont\huge\bfseries}
  %% and so on down to subparagraph specification
  \renewcommand*{\subparaheadstyle}{\normalfont\normalsize\bfseries}
}
\headstyles{default}
```

A somewhat different set of headstyles is used for this manual. When using `\makeheadstyles` you only need to specify things that differ from the *default*. Within the class the *memman* set of headstyles is defined as:

```
\newcommand*{\addperiod}[1]{#1.}
\makeheadstyles{memman}{%
```

```
% book changes
  \renewcommand*{\booknamefont}{\normalfont\huge\sffamily}
  \renewcommand*{\booknumfont}{\normalfont\huge\sffamily}
  \renewcommand*{\booktitlefont}{\normalfont\Huge\sffamily}
  \renewcommand*{\midbookskip}{\par\vskip 2\onelineskip}%
% part changes
  \renewcommand*{\partnamefont}{\normalfont\huge\sffamily}
  \renewcommand*{\partnumfont}{\normalfont\huge\sffamily}
  \renewcommand*{\parttitlefont}{\normalfont\Huge\sffamily}
  \renewcommand*{\midpartskip}{\par\vskip 2\onelineskip}%
% chapter
  \chapterstyle{demo3}
% section
  \setbeforesecskip{-1.333\onelineskip
                    \@plus -0.5\onelineskip \@minus -.5\onelineskip}%
  \setaftersecskip{0.667\onelineskip \@plus 0.1\onelineskip}%
  \setsecheadstyle{\normalfont\scshape\raggedright}%
% subsection
  \setbeforesubsecskip{-0.667\onelineskip
                       \@plus -0.25\onelineskip \@minus -0.25\onelineskip}%
  \setaftersubsecskip{0.333\onelineskip \@plus 0.1\onelineskip}%
  \setsubsecheadstyle{\normalfont\bfseries\raggedright}%
% subsubsection
  \setbeforesubsubsecskip{-0.667\onelineskip
                          \@plus -0.25\onelineskip \@minus -0.25\onelineskip}%
  \setaftersubsubsecskip{0.333\onelineskip \@plus 0.1\onelineskip}%
  \setsubsubsecheadstyle{\normalfont\normalsize\itshape\raggedright}%
% paragraph
  \setbeforeparaskip{1.0\onelineskip
                     \@plus 0.5\onelineskip \@minus 0.2\onelineskip}%
  \setafterparaskip{-1em}%
  \setparaheadstyle{\normalfont\normalsize\itshape\addperiod}%
% subparagraph
  \setsubparaindent{\parindent}%
  \setbeforesubparaskip{1.0\onelineskip
                        \@plus 0.5\onelineskip \@minus 0.2\onelineskip}%
  \setaftersubparaskip{-1em}%
  \setsubparaheadstyle{\normalfont\normalsize\itshape\addperiod}}
```

You can see the effect throughout this document. This chapter is slightly different in that I have used the *pedersen* chapterstyle instead of the *demo3* chapterstyle that I have normally used.

Several other sets of headstyles are provided as well and the full list is below. The different fonts used are given in Table 6.6 and generally speaking they start off being large for chapter heads but are normal size by the time subsubsection heads are reached, or before.

bringhurst A set based on Bringhurst's *Elements of Typographic Style* [Bri99]. It uses the

Table 6.6: Fonts used by different headstyles

Headstyles	chapter	section	subsec	subsubsec	para	subpara
bringhurst	CAPS	S. CAPS	*Italic*	S. CAPS	*Italic*	*Italic*
crosshead	**Bold**	CAPS	**Bold**	S. CAPS	*Italic*	S. CAPS
default	**Bold**	**Bold**	**Bold**	**Bold**	**Bold**	**Bold**
dowding	*Italic*	CAPS	S. CAPS	*Italic*	*Italic*	*Italic*
komalike	Sans	Sans	Sans	Sans	Sans	Sans
memman	Sans	S. CAPS	**Bold**	*Italic*	*Italic*	*Italic*
ntglike	**Bold**	**Bold**	**Bold**	*Slanted*	*Slanted*	*Slanted*
tandh	**Bold**	CAPS	*Italic*	**Bold**	*Italic*	*Italic*
wilsondob	*Italic*	CAPS	*Italic*	S. CAPS	*Italic*	*Italic*

bringhurst chapterstyle (Figure B.7).

crosshead This set uses the *crosshead* chapterstyle and the lower level division titles are set as crossheads.

default The default set corresponding the LaTeX book class.

dowding A set based on Dowding's *Finer Points* [Dow96]. It uses the *dowding* chapterstyle (Figure B.14).

komalike A set based on the kind of headings used in the KOMA scrbook class, where there are all in a bold sans serif font. It uses the *komalike* chapterstyle (Figure B.17).

memman The set used in this document, including the *demo3* chapterstyle.

ntglike A set based on the kind of headings used in the NTG (Dutch TUG) boek class. It uses the *ntglike* chapterstyle (Figure B.20) and the headings are quiter than the default.

tandh A set based the heads used in Thames & Hudson *Manual of Typography* [McL80]. It uses the *tandh* chapterstyle (Figure B.22)

wilsondob A set based on those used in Adrian Wilson's *Design of Books* [Wil93]. It uses the *wilsondob* chapterstyle (Figure B.26).

Seven

Pagination and headers

The focus of this chapter is on marking the pages with signposts so that the reader can more readily navigate through the document.

7.1 Pagination and folios

Every page in a LaTeX document is included in the pagination. That is, there is a number associated with every page and this is the value of the `page` counter. This value can be changed at any time via either `\setcounter` or `\addtocounter`.

```
\pagenumbering{⟨rep⟩}
\pagenumbering*{⟨rep⟩}
```

The macros `\pagenumbering` and `\pagenumbering*` cause the folios to be printed using the counter representation ⟨*rep*⟩ for the page number, where ⟨*rep*⟩ can be one of: `Alph`, `alph`, `arabic`, `Roman` or `roman` for uppercase and lowercase letters, arabic numerals, and uppercase and lowercase Roman numerals, respectively. As there are only 26 letters, `Alph` or `alph` can only be used for a limited number of pages. Effectively, the macros redefine `\thepage` to be `\rep{page}`.

Additionally, the `\pagenumbering` command resets the `page` counter to one; the starred version does not change the counter. It is usual to reset the page number back to one each time the style is changed, but sometimes it may be desirable to have a continuous sequence of numbers irrespective of their displayed form, which is where the starred version comes in handy.

Hint. *If you are using* hyperref *and include a title page, the following setup might be useful*

```
\pagenumbering{Alpha}
\begin{titlingpage}
  ...
\end{titlingpage}
\frontmatter
... % this has roman numbering
\mainmatter
... % this has arabic numbering
```

The code has the advantage that hyperref *does not warn that the page anchors for pages 1 and 2 are already defined (the* `titlingpage` *env does not show the A, B page numbers).*

```
\savepagenumber
\restorepagenumber
```

The macro \savepagenumber saves the current page number, and the macro \restorepagenumber sets the page number to the saved value. This pair of commands may be used to apparently interrupt the pagination. For example, perhaps some full page illustrations will be electronically tipped in to the document and pagination is not required for these. This could be done along the lines of:

```
\clearpage          % get onto next page
\savepagenumber     % save the page number
\pagestyle{empty}   % no headers or footers
%% insert the illustrations
\clearpage
\pagestyle{...}
\restorepagenumber
...
```

If you try this sort of thing, you may have to adjust the restored page number by one.

```
\restorepagenumber
% perhaps \addtocounter{page}{1} or \addtocounter{page}{-1}
```

Depending on the timing of the \...pagenumber commands and TeX's decisions on page breaking, this may or may not be necessary.

7.2 PAGE STYLES

The class provides a selection of pagestyles that you can use and if they don't suit, then there are means to define your own.

These facilities were inspired by the fancyhdr package [Oos96], although the command set is different.

The standard classes provide for a footer and header for odd and even pages. Thus there are four elements to be specified for a pagestyle. This class partitions the headers and footers into left, center and right portions, so that overall there is a total of 12 elements that have to be specified for a pagestyle. You may find, though, that one of the built in pagestyles meets your needs so you don't have to worry about all these specifications.

\pagestyle{⟨*style*⟩}
\thispagestyle{⟨*style*⟩}

\pagestyle sets the current pagestyle to ⟨*style*⟩, where ⟨*style*⟩ is a word containing only letters. On a particular page \thispagestyle can be used to override the current pagestyle for the one page.

Some of the class' commands automatically call \thispagestyle. For example:

- the titlingpage environment calls

  ```
  \thispagestyle{titlingpagestyle}
  ```

- if \cleardoublepage will result in an empty verso page it calls

  ```
  \thispagestyle{cleared}
  ```

  for the empty page.

For reference, the full list is given in Table 7.1.

The page styles provided by the class are:

empty The headers and footers are empty.

plain The header is empty and the folio (page number) is centered at the bottom of the page.

headings The footer is empty. The header contains the folio at the outer side of the page; on verso pages the chapter name, number and title, in slanted uppercase is set at the spine margin and on recto pages the section number and uppercase title is set by the spine margin.

myheadings Like the *headings* style the footer is empty. You have to specify what is to go in the headers.

simple The footer is empty and the header contains the folio (page number) at the outer side of the page. It is like the *headings* style but without any title texts.

ruled The footer contains the folio at the outside. The header on verso pages contains the chapter number and title in small caps at the outside; on recto pges the section title is typeset at the outside using the normal font. A line is drawn underneath the header.

Ruled This is like the *ruled* style except that the headers and footers extend into the fore-edge margin.

companion This is a copy of the pagestyle in the *Companion* series (e.g., see [MG+04]). It is similar to the *Ruled* style in that the header has a rule which extends to the outer edge of the marginal notes. The folios are set in bold at the outer ends of the header. The chapter title is set in a bold font flushright in the verso headers, and the section number and title, again in bold, flushleft in the recto headers. There are no footers.

book This is the same as the *plain* pagestyle.

chapter This is the same as the *plain* pagestyle.

cleared This is the same as the *empty* pagestyle.

part This is the same as the *plain* pagestyle.

title This is the same as the *plain* pagestyle.

titlingpage This is the same as the *empty* pagestyle.

```
\uppercaseheads \nouppercaseheads
```

Following the declaration `\nouppercaseheads` the titles in the *headings* pagestyle will not be automatically uppercased. The default is `\uppercaseheads` which specifies that the titles are to be automatically uppercased.

Change 2012. *The upper casing macro used by* `\uppercaseheads` *has been changed into* `\MakeTextUppercase` *(from the* textcase *package) such that the upper casing does not touch math, references or citations.*

For the *myheadings* pagestyle above, you have to define your own titles to go into the header. Each sectioning command, say `\sec`, calls a macro called `\secmark`. A pagestyle usually defines this command so that it picks up the title, and perhaps the number, of the `\sec`. The pagestyle can then use the information for its own purposes.

```
\markboth{⟨left⟩}{⟨right⟩}
\markright{⟨right⟩}
```

Table 7.1: The use of \thispagestyle

Called from	Style
\book	*book*
\chapter	*chapter*
\cleardoublepage	*cleared*
\cleartorecto	*cleared*
\cleartoverso	*cleared*
\epigraphhead	*epigraph*
\listoffigures	*chapter*
\listoftables	*chapter*
\maketitle	*title*
\part	*part*
\tableofcontents	*chapter*
thebibliography	*chapter*
theindex	*chapter*
titlingpage	*titlingpage*

\markboth sets the values of two *markers* to ⟨*left*⟩ and ⟨*right*⟩ respectively, at the point in the text where it is called. Similarly, \markright sets the value of a marker to ⟨*right*⟩.

```
\leftmark \rightmark
```

The macro \leftmark contains the value of the ⟨*left*⟩ argument of the *last* \markboth on the page. The macro \rightmark contains the value of the ⟨*right*⟩ argument of the *first* \markboth or \markright on the page, or if there is not one it contains the value of the most recent ⟨*right*⟩ argument.

A pagestyle can define the \secmark commands in terms of \markboth or \markright, and then use \leftmark and/or \rightmark in the headers or footers. I'll show examples of how this works later, and this is often how the *myheadings* style gets implemented.

All the division commands include a macro that you can define to set marks related to that heading. Other commands also include macros that you can redefine for setting marks.

The \...mark commands are listed in Table 7.2. When they are called by the relevant main macro, those that take an argument are called with the 'title' as the argument's value. For example, the \chapter macro calls \chaptermark with the value of the title specified as being for the header.

Please remember that the macros listed in Table 7.2 are 'provider' macros, i.e. they provide information for \leftmark and \rightmark for you to use later on. To gain access to the section title, you do *not* use \sectionmark in the header or footer. It is a macro that provides information, but you need to use \leftmark or \rightmark to access depending on how you have defined \sectionmark.

Table 7.2: Mark macros for page headers

Main macro	default mark definition
\book(*)	\newcommand*{\bookpagemark}[1]{}
\part(*)	\newcommand*{\partmark}[1]{}
\chapter(*)	\newcommand*{\chaptermark}[1]{}
\section(*)	\newcommand*{\sectionmark}[1]{}
\subsection(*)	\newcommand*{\subsectionmark}[1]{}
\subsubsection(*)	\newcommand*{\subsubsectionmark}[1]{}
\paragraph(*)	\newcommand*{\paragraphmark}[1]{}
\subparagraph(*)	\newcommand*{\subparagraphmark}[1]{}
\tableofcontents(*)	\newcommand*{\tocmark}[1]{}
\listoffigures(*)	\newcommand*{\lofmark}[1]{}
\listoftables(*)	\newcommand*{\lotmark}[1]{}
\thebibliography	\newcommand*{\bibmark}{}
\theindex	\newcommand*{\indexmark}{}
\theglossary	\newcommand*{\glossarymark}{}
\PoemTitle	\newcommand*{\poemtitlemark}[1]{}
\PoemTitle*	\newcommand*{\poemtitlestarmark}[1]{}

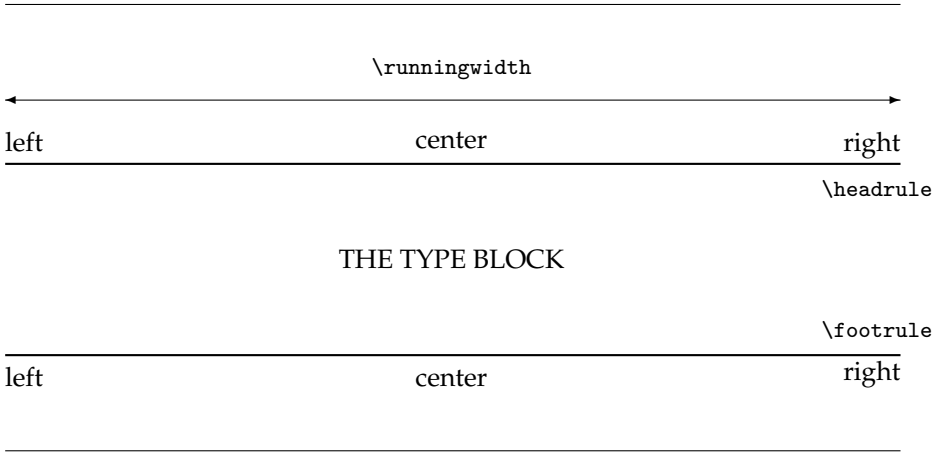


Figure 7.1: Header and footer slots

7.3 Making headers and footers

As mentioned, the class provides for left, center, and right slots in even and odd headers and footers. This section describes how you can make your own pagestyle using these 12 slots. The 6 slots for a page are diagrammed in Figure 7.1.

The class itself uses the commands from this section. For example, the *plain* pagestyle is defined as

```
\makepagestyle{plain}
\makeevenfoot{plain}{}{\thepage}{}
\makeoddfoot{plain}{}{\thepage}{}
```

which centers the page number at the bottom of the page.

`\makepagestyle{`⟨*style*⟩`}`
`\aliaspagestyle{`⟨*alias*⟩`}{`⟨*original*⟩`}`
`\copypagestyle{`⟨*copy*⟩`}{`⟨*original*⟩`}`

The command `\makepagestyle` specifies a pagestyle ⟨*style*⟩ which is initially equivalent to the *empty* pagestyle. On the other hand, `\aliaspagestyle` defines the ⟨*alias*⟩ pagestyle to be the same as the ⟨*original*⟩ pagestyle. As an example of the latter, the class includes the code

```
\aliaspagestyle{part}{plain}
\aliaspagestyle{chapter}{plain}
\aliaspagestyle{cleared}{empty}
```

The `\copypagestyle` command creates a new pagestyle called ⟨*copy*⟩ using the ⟨*original*⟩ pagestyle specification.

If an alias and a copy pagestyle are created based on the same ⟨*original*⟩ and later the ⟨*original*⟩ is modified, the alias and copy behave differently. The appearance of the alias pagestyle will continue to match the modified ⟨*original*⟩ but the copy pagestyle is unaffected by any change to the ⟨*original*⟩. You cannot modify an alias pagestyle but you can modify a copy pagestyle.

`\makeevenhead{`⟨*style*⟩`}{`⟨*left*⟩`}{`⟨*center*⟩`}{`⟨*right*⟩`}`
`\makeoddhead{`⟨*style*⟩`}{`⟨*left*⟩`}{`⟨*center*⟩`}{`⟨*right*⟩`}`
`\makeevenfoot{`⟨*style*⟩`}{`⟨*left*⟩`}{`⟨*center*⟩`}{`⟨*right*⟩`}`
`\makeoddfoot{`⟨*style*⟩`}{`⟨*left*⟩`}{`⟨*center*⟩`}{`⟨*right*⟩`}`

The macro `\makeevenhead` defines the ⟨*left*⟩, ⟨*center*⟩, and ⟨*right*⟩ portions of the ⟨*style*⟩ pagestyle header for even numbered (verso) pages. Similarly `\makeoddhead`, `\makeevenfoot`, and `\makeoddfoot` define the ⟨*left*⟩, ⟨*center*⟩ and ⟨*right*⟩ portions of the ⟨*style*⟩ header for odd numbered (recto) pages, and the footers for verso and recto pages. These commands for ⟨*style*⟩ should be used after the corresponding `\makepagestyle` for ⟨*style*⟩.

`\makerunningwidth{`⟨*style*⟩`}[`⟨*footwidth*⟩`]{`⟨*headwidth*⟩`}`
`\headwidth`

The macro `\makerunningwidth` sets the widths of the ⟨*style*⟩ pagestyle headers and footers. The header width is set to ⟨*headwidth*⟩. If the optional ⟨*footwidth*⟩ is present, then the footer width is set to that, otherwise to ⟨*headwidth*⟩. The header width is stored as the length `\`⟨*style*⟩`headrunwidth` and the footer width as `\`⟨*style*⟩`footrunwidth`.

The `\makepagestyle` initialises the widths to be the textwidth, so the macro need only be used if some other width is desired. The length `\headwidth` is provided as a (scratch) length that may be used for headers or footers, or any other purpose.

`\makeheadrule{`⟨*style*⟩`}{`⟨*width*⟩`}{`⟨*thickness*⟩`}`
`\makefootrule{`⟨*style*⟩`}{`⟨*width*⟩`}{`⟨*thickness*⟩`}{`⟨*skip*⟩`}`
`\makeheadfootruleprefix{`⟨*style*⟩`}{`⟨*for headrule*⟩`}{`⟨*for footrule*⟩`}`

A header may have a rule drawn between it and the top of the typeblock, and similarly a rule may be drawn between the bottom of the typeblock and the footer. The `\makeheadrule` macro specifies the ⟨*width*⟩ and ⟨*thickness*⟩ of the rule below the ⟨*style*⟩ pagestyle header, and the `\makefootrule` does the same for the rule above the footer; the additional ⟨*skip*⟩ argument is a distance that specifies the vertical positioning of the foot rule (see `\footruleskip`). The `\makepagestyle` macro initialises the ⟨*width*⟩ to the `\textwidth` and the ⟨*thickness*⟩ to 0pt, so by default no rules are visible. The macro `\makeheadfootruleprefix` is intended for adding alternative colors to the head/foot rules, e.g.

```
\makeheadfootruleprefix{mystyle}{\color{red}}{\color{blue}}
```

`\normalrulethickness`

`\normalrulethickness` is the normal thickness of a visible rule, by default 0.4pt. It can be changed using `\setlength`, although I suggest that you do not unless perhaps when using at least the 14pt class option.

`\footruleheight`
`\footruleskip`

The macro `\footruleheight` is the height of a normal rule above a footer (default zero). `\footruleskip` is a distance sufficient to ensure that a foot rule will be placed between the bottom of the typeblock and the footer. Despite appearing to be lengths, if you really need to change the values use `\renewcommand`, not `\setlength`.

`\makeheadposition{`⟨*style*⟩`}`
`{`⟨*eheadpos*⟩`}{`⟨*oheadpos*⟩`}{`⟨*efootpos*⟩`}{`⟨*ofootpos*⟩`}`

The `\makeheadposition` macro specifies the horizontal positioning of the even and odd headers and footers, respectively, for the ⟨*style*⟩ pagestyle. Each of the ⟨*...pos*⟩ arguments may be `flushleft`, `center`, or `flushright`, with the obvious meanings. An empty, or unrecognised, argument is equivalent to `center`. This macro is really only of use if the header/footer width is not the same as the `\textwidth`.

`\makepsmarks{`⟨*style*⟩`}{`⟨*code*⟩`}`

The last thing that the `\pagestyle{`⟨*style*⟩`}` does is call the ⟨*code*⟩ argument of the `\makepsmarks` macro for ⟨*style*⟩. If specified, this code will often redefine the various macros that provide the automated data for the headers (or footers). For example how `\chapter` makes the title and chapter number avaialble for the header. If no ⟨*code*⟩ is specified (or `\makepsmarks` is not setup for this ⟨*style*⟩, the providing macros already active will be used.

`\makeheadfootstrut{`⟨*style*⟩`}{`⟨*head strut*⟩`}{`⟨*foot strut*⟩`}`

The headers and footers are each made up of three separate entities. At the front and end of these a special ⟨*style*⟩ related strut is inserted. By default `\makepagestyle` will initialize them to `\strut` (except the *empty* style where the struts are empty). One can use the macro above to change these struts to something different.

* * *

Writing the ⟨*code*⟩ part for `\makepsmarks` can be very daunting, as you will be needing to define most of the macros listed in Table 7.2

However, most uses are very similar, so we have provided some helper macros:

```
\createmark{⟨sec⟩}{⟨marks⟩}{⟨show⟩}{⟨prefix⟩}{⟨postfix⟩}
\createplainmark{⟨type⟩}{⟨marks⟩}{⟨text⟩}
\memUChead{⟨text⟩}
\uppercaseheads \nouppercaseheads
```

The macro `\createmark{`⟨*sec*⟩`}{`⟨*marks*⟩`}{`⟨*show*⟩`}{`⟨*prefix*⟩`}{`⟨*postfix*⟩`}` defines a new macro `\<sec>mark` that will take one argument, where

⟨***marks***⟩ is either `left`, `right` or `both`,

⟨***sec***⟩ is a sectional division such as `part`, `chapter`, `section`, etc.,

⟨***show***⟩ is one of the terms `shownumber`, `nonumber` or `notitle`. They control what is available in the provided header text. The terms `shownumber`, `nonumber` control whether the number is displayed within `\mainmatter` (under `\frontmatter` no numbers are displayed.

The term `notitle` (which implies `shownumber`) will only show the number part (including the ⟨*postfix*⟩ and ⟨*prefix*⟩ parts described below) and *no* title.

⟨***pre-/postfix***⟩ are affixed before and after the division number.

For example:

```
\createmark{section}{left}{nonumber}{}{}
\createmark{section}{both}{nonumber}{}{}
\createmark{section}{right}{nonumber}{}{}
```

is equivalent to, respectively

```
\def\sectionmark#1{\markboth{\memUChead{#1}}{}}
\def\sectionmark#1{\markboth{\memUChead{#1}}{\memUChead{#1}}}
\def\sectionmark#1{\markright{\memUChead{#1}}}
```

whereas

```
\createmark{section}{left}{shownumber}{}{}
```

is equivalent to the following pseudo code

```
\def\sectionmark#1{\markboth{
  \memUChead{
    if secnumdepth > number:
      if mainmatter:
        prefix \thesection postfix
      fi
    fi
    #1% aka the section title
  }
}{}}
```

and

```
\createmark{section}{left}{notitle}{}{}
```

is equivalent to the following pseudo code

```
\def\sectionmark#1{\markboth{
  \memUChead{
```

```
    if secnumdepth > number:
      if mainmatter:
        prefix \thesection postfix
      fi
    fi
  }
}{}}
```

The macro `\createplainmark` defines the `\<type>mark`, where ⟨*type*⟩ is an unnumbered division-like head, such as `toc`, `lof`, `index`, using ⟨*text*⟩ as the mark value, and ⟨*marks*⟩ is `left`, `both` or `right`. For example:

```
\createplainmark{toc}{left}{\contentsname}
\createplainmark{lot}{right}{\listtablename}
\createplainmark{bib}{both}{\bibname}
```

is equivalent to

```
\def\tocmark{\markboth{\memUChead{\contentsname}}{}}
\def\lotmark{\markright{\memUChead{\listtablename}}}
\def\lofmark{\markboth{\memUChead{\bibname}}{\memUChead{\bibname}}}
```

Lastly the macro `\memUChead` is an internal macro handling the upper- or non-uppercasing. Its definition is usually controlled using `\uppercaseheads` which sets `\memUChead` to `\MakeTextUppercase` (from textcase) whereas `\nouppercaseheads` defines it to output the argument unchanged. The class default is `\uppercaseheads`.

The class' default page style is *headings*, which has the following setup for `\makepsmarks` (slight pseudocode):

```
if twopage setup:
  \makepsmarks{headings}{%
    \createmark{chapter}{left}{shownumber}{\@chapapp\ }{. \ }
    \createmark{section}{right}{shownumber}{}{. \ }
    \createplainmark{toc}{both}{\contentsname}
    \createplainmark{lof}{both}{\listfigurename}
    \createplainmark{lot}{both}{\listtablename}
    \createplainmark{bib}{both}{\bibname}
    \createplainmark{index}{both}{\indexname}
    \createplainmark{glossary}{both}{\glossaryname}
  }
else: % aka oneside
  \makepsmarks{headings}{%
    \createmark{chapter}{right}{shownumber}{\@chapapp\ }{. \ }
    \createplainmark{toc}{right}{\contentsname}
    \createplainmark{lof}{right}{\listfigurename}
    \createplainmark{lot}{right}{\listtablename}
    \createplainmark{bib}{right}{\bibname}
    \createplainmark{index}{right}{\indexname}
```

```
    \createplainmark{glossary}{right}{\glossaryname}
  }
fi
```

Where `\@chapapp` is a special macro what equals `\chaptername` unless `\appendix` is active, where it is then equal to `\appendixname`.

Note how the twosided setup only include marks for `\chapter` and `\section` and onesided only for `\chapter`.

So what if we wanted to add support for `\subsection` to an already defined page style? Use this:

`\addtopsmarks{`⟨*pagestyle*⟩`}{`⟨*prepend*⟩`}{`⟨*append*⟩`}`

Here ⟨*prepend*⟩ and ⟨*append*⟩ is inserted *before* and *after* the current definition of `\makepsmarks` for ⟨*pagestyle*⟩. So to answer our question we could use

```
\addtopsmarks{headings}{}{%
  \createmark{subsection}{right}{shownumber}{}{. \space}}
\pagestyle{headings}
```

– always remember to activate the page style afterwards.

When the class runs the marks part of page style, it does not zero out old marks, i.e. if an old `\sectionmark` exist, it still exist even if we do not change it. This is both a good and a bad thing. To help users redefine these marks to doing nothing we provide

`\clearplainmark{`⟨*type*⟩`}`
`\clearmark{`⟨*type*⟩`}`

The used types are the same as for `\createplainmark` and `\createmark`.

7.3.1 Example pagestyles

Perhaps when preparing drafts you want to note on each page that it is a draft document. Assuming that you are using the *headings* page style and that the default *plain* page style is used on chapter openings, then you could define the following in the preamble (`\ifdraftdoc` is provided by the class and is set `true` when the draft option is used).

```
\ifdraftdoc
  \makeevenfoot{plain}{}{\thepage}{\textit{Draft: \today}}
  \makeoddfoot{plain}{\textit{Draft: \today}}{\thepage}{}
  \makeevenfoot{headings}{}{}{\textit{Draft: \today}}
  \makeoddfoot{headings}{\textit{Draft: \today}}{}{}
\fi
```

Now when the draft option is used the word 'Draft:' and the current date will be typeset in italics at the bottom of each page by the spine margin. If any *empty* pages should be marked as well, specify similar footers for that style as well.

Here is part of the standard definition of the *headings* pagestyle for the book class which uses many internal LaTeX commands; but note that the class does not use this!

```
\def\ps@headings{%
  \let\@oddfoot\@empty\let\@evenfoot\@empty
  \def\@evenhead{\thepage\hfil\slshape\leftmark}%
  \def\@oddhead{{\slshape\rightmark}\hfil\thepage}%
  \def\chaptermark##1{%
    \markboth{\MakeUppercase{%
      \ifnum\c@secnumdepth > \m@ne
        \if@mainmatter
          \@chapapp\ \thechapter. \ %
        \fi
      \fi
      ##1}}{}}%
  \def\sectionmark##1{%
    \markright{\MakeUppercase{%
      \ifnum\c@secnumdepth > \z@
        \thesection. \ %
      \fi
      ##1}}}}
```

You don't need to understand this but in outline the first three lines specify the contents of the footers and headers, and the remainder of the code sets the marks that will be used in the headers. The `\leftmark` is specified to be the word 'chapter', followed by the number if it is in the `\mainmatter` and the `secnumdepth` is such that chapters are numbered, followed by the chapter's title; all this is made to be in upper case (via the `\MakeUppercase` macro). Similarly the other mark, `\rightmark`, is the section number, if there is one, and the section's title, again all in upper case.

A transliteration of this code into memoir's original coding style is:

```
\makepagestyle{headings}
\makeevenhead{headings}{\thepage}{}{\slshape\leftmark}
\makeoddhead{headings}{\slshape\rightmark}{}{\thepage}
\makepsmarks{headings}{%
  \def\chaptermark##1{%
    \markboth{\MakeTextUppercase{%
      \ifnum\c@secnumdepth > \m@ne
        \if@mainmatter
          \@chapapp\ \thechapter. \ %
        \fi
      \fi
      ##1}}{}}%
  \def\sectionmark##1{%
    \markright{\MakeTextUppercase{%
      \ifnum\c@secnumdepth > \z@
        \thesection. \ %
      \fi
      ##1}}}
  \def\tocmark{\markboth{\MakeTextUppercase{\contentsname}}{}}
  \def\lofmark{\markboth{\MakeTextUppercase{\listfigurename}}{}}
```

```
\def\lotmark{\markboth{\MakeTextUppercase{\listtablename}}{}}
\def\bibmark{\markboth{\MakeTextUppercase{\bibname}}{}}
\def\indexmark{\markboth{\MakeTextUppercase{\indexname}}{}}
\def\glossarymark{\markboth{\MakeTextUppercase{\glossaryname}}{}}}
```

As you can see, defining the marks for a pagestyle is not necessarily the simplest thing in the world. With the extra tools described above, this code can be reduced to

```
\makepagestyle{headings}
\makeevenhead{headings}{\thepage}{}{\slshape\leftmark}
\makeoddhead{headings}{\slshape\rightmark}{}{\thepage}
\makepsmarks{headings}{%
  \createmark{chapter}{left}{shownumber}{\@chapapp\ }{. \ }
  \createmark{section}{right}{shownumber}{}{. \ }
  \createplainmark{toc}{both}{\contentsname}
  \createplainmark{lof}{both}{\listfigurename}
  \createplainmark{lot}{both}{\listtablename}
  \createplainmark{bib}{both}{\bibname}
  \createplainmark{index}{both}{\indexname}
  \createplainmark{glossary}{both}{\glossaryname}
}
```

Note that if you want to use a predefined page style, but would like to not use automatic uppercasing, then issue `\nouppercaseheads` and reload the page style, for example with the default page style in memoir

```
\nouppercaseheads
\pagestyle{headings}
```

Header with the document title

As mentioned before, some publishers like the title of the book to be in the header. A simple header is probably all that is needed as it is unlikely to be a technical publication. Here is a use for *myheadings*.

```
\makevenhead{myheadings}{\thepage}{}{DOCUMENT TITLE}
\makeoddhead{myheadings}{Chapter~\thechapter}{}{\thepage}
```

Part and chapter in the header

Some documents have both part and chapter divisions and in such cases it may be useful for the reader to have the current part and chapter titles in the header. The *headings* pagestyle can be easily modified to accomplish this by simply resetting the marks for part and chapter:

```
\makepsmarks{headings}{%
  \createmark{part}{left}{shownumber}{\partname\ }{. \ }
  \createmark{chapter}{right}{shownumber}{\@chapapp\ }{. \ }
  \createplainmark{toc}{both}{\contentsname}
  \createplainmark{lof}{both}{\listfigurename}
  \createplainmark{lot}{both}{\listtablename}
  \createplainmark{bib}{both}{\bibname}
```

```
\createplainmark{index}{both}{\indexname}
\createplainmark{glossary}{both}{\glossaryname}}
```

The Companion pagestyle

This example demonstrates most of the page styling commands. In the *LaTeX Companion* series of books [MG+04, GM+07, GR99] the header is wider than the typeblock, sticking out into the outer margin, and has a rule underneath it. The page number is in bold and at the outer end of the header. Chapter titles are in verso headers and section titles in recto headers, both in bold font and at the inner margin. The footers are empty.

The first thing to do in implementing this style is to calculate the width of the headers, which extend to cover any marginal notes.

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

Now we can set up an empty *companion* pagestyle and start to change it by specifying the new header and footer width:

```
\makepagestyle{companion}
\makerunningwidth{companion}{\headwidth}
```

and specify the width and thickness for the header rule, otherwise it will be invisible.

```
\makeheadrule{companion}{\headwidth}{\normalrulethickness}
```

In order to get the header to stick out into the fore-edge margin, verso headers have to be flushright (raggedleft) and recto headers to be flushleft (raggedright). As the footers are empty, their position is immaterial.

```
\makeheadposition{companion}{flushright}{flushleft}{}{}
```

The current chapter and section titles are obtained from the `\leftmark` and `\rightmark` macros which are defined via the `\chaptermark` and `\sectionmark` macros. Remember that `\leftmark` is the last ⟨*left*⟩ marker and `\rightmark` is the first ⟨*right*⟩ marker on the page.

Chapter numbers are not put into the header but the section number, if there is one, is put into the header. We have to make sure that the correct definitions are used for these as well as for the ToC[1] and other similar elements, and this is where the `\makepsmarks` macro comes into play.

```
\makepsmarks{companion}{%
 \nouppercaseheads
 \createmark{chapter}{both}{nonumber}{}{}
 \createmark{section}{right}{shownumber}{}{. \space}
 \createplainmark{toc}{both}{\contentsname}
 \createplainmark{lof}{both}{\listfigurename}
 \createplainmark{lot}{both}{\listtablename}
 \createplainmark{bib}{both}{\bibname}
 \createplainmark{index}{both}{\indexname}
 \createplainmark{glossary}{both}{\glossaryname}
```

[1] The ToC and friends are described in detail in Chapter 9.

The preliminaries have all been completed, and it just remains to specify what goes into each header and footer slot (but the footers are empty).

```
\makeevenhead{companion}%
  {\normalfont\bfseries\thepage}{}{%
   \normalfont\bfseries\leftmark}
\makeoddhead{companion}%
  {\normalfont\bfseries\rightmark}{}{%
   \normalfont\bfseries\thepage}
```

Now issuing the command `\pagestyle{companion}` will produce pages typeset with *companion* pagestyle headers. This pagestyle is part of the class.

For instance, if you wanted `\subsection` titles to appear in the page headers of the *companion* pagestyle then this would be a way of doing it:

```
\addtopsmarks{companion}{}{%
  \createmark{subsection}{right}{shownumber}{}{. \space}}
```

The ruled pagestyle

For practical reasons I prefer a page style with headings where the chapter title is at least in the center of the page, and for technical works is at the fore-edge. I also prefer the page number to be near the outside edge. When picking up a book and skimming through it, either to get an idea of what is in it or to find something more specific, I hold it in one hand at the spine and use the other for flicking the pages. The book is half closed while doing this and it's much easier to spot things at the fore-edge than those nearer the spine. The *ruled* page style is like this. The general plan is defined as:

```
\makepagestyle{ruled}
\makeevenfoot {ruled}{\thepage}{}{} % page numbers at the outside
\makeoddfoot  {ruled}{}{}{\thepage}
\makeheadrule {ruled}{\textwidth}{\normalrulethickness}
\makeevenhead {ruled}{\scshape\leftmark}{}{} % small caps
\makeoddhead  {ruled}{}{}{\rightmark}
```

The other part of the specification has to ensure that the `\chapter` and `\section` commands make the appropriate marks for the headers. I wanted the numbers to appear in the headers, but not those for sections. The following code sets these up, as well as the marks for the other document elements.

```
\makepsmarks{ruled}{%
  \nouppercaseheads
  \createmark{chapter}{left}{shownumber}{}{. \space}
  \createmark{section}{right}{nonumber}{}{}
  \createplainmark{toc}{both}{\contentsname}
  \createplainmark{lof}{both}{\listfigurename}
  \createplainmark{lot}{both}{\listtablename}
  \createplainmark{bib}{both}{\bibname}
  \createplainmark{index}{both}{\indexname}
  \createplainmark{glossary}{both}{\glossaryname}
}
```

7.3.2 Index headers

If you look at the Index you will see that the header shows the first and last entries on the page. A main entry in the index looks like:

```
\item \idxmark{entry}, page number(s)
```

and in the preamble to this book `\idxmark` is defined as

```
\newcommand{\idxmark}[1]{#1\markboth{#1}{#1}}
```

This typesets the entry and also uses the entry as markers so that the first entry on a page is held in `\rightmark` and the last is in `\leftmark`.

As index entries are usually very short, the Index is set in two columns. Unfortunately LaTeX's marking mechanism can be very fragile on twocolumn pages.[2]

The index itself is called by

```
\clearpage
\pagestyle{index}
\renewcommand{\preindexhook}{%
The first page number is usually, but not always,
the primary reference to
the indexed topic.\vskip\onelineskip}
\printindex
```

The *index* pagestyle, which is the crux of this example, is defined here as:

```
\makepagestyle{index}
  \makeheadrule{index}{\textwidth}{\normalrulethickness}
  \makeevenhead{index}{\rightmark}{}{\leftmark}
  \makeoddhead{index}{\rightmark}{}{\leftmark}
  \makeevenfoot{index}{\thepage}{}{}
  \makeoddfoot{index}{}{}{\thepage}
```

This, as you can hopefully see, puts the first and last index entries on the page into the header at the left and right, with the folios in the footers at the outer margin.

7.3.3 Float pages

`\ifonlyfloats{`⟨*yes*⟩`}{`⟨*no*⟩`}`

There are occasions when it is desirable to have different headers on pages that only contain figures or tables. If the command `\ifonlyfloats` is issued on a page that contains no text and only floats then the ⟨*yes*⟩ argument is processed, otherwise on a normal page the ⟨*no*⟩ argument is processed. The command is most useful when defining a pagestyle that should be different on a float-only page.

For example, assume that the *companion* pagestyle is to be generally used, but on float-only pages all that is required is a pagestyle similar to *plain*. Borrowing some code from the *companion* specification this can be accomplished like:

```
\makepagestyle{floatcomp}
% \headwidth has already been defined for the companion style
\makeheadrule{floatcomp}{\headwidth}%
```

[2] This was fixed in the LaTeX kernel, but including the functionality from the fix|tx2e package.

```
  {\ifonlyfloats{0pt}{\normalrulethickness}}
\makeheadposition{floatcomp}{flushright}{flushleft}{}{}
\makepsmarks{floatcomp}{\companionpshook}
\makeevenhead{floatcomp}%
             {\ifonlyfloats{}{\normalfont\bfseries\thepage}}%
             {}%
             {\ifonlyfloats{}{\normalfont\bfseries\leftmark}}
\makeoddhead{floatcomp}%
             {\ifonlyfloats{}{\normalfont\bfseries\rightmark}}%
             {}%
             {\ifonlyfloats{}{\normalfont\bfseries\thepage}}
\makeevenfoot{floatcomp}{}{\ifonlyfloats{\thepage}{}}{}
\makeoddfoot{floatcomp}{}{\ifonlyfloats{\thepage}{}}{}
```

The code above for the *floatcomp* style should be compared with that for the earlier *companion* style.

The headrule is invisible on float pages by giving it zero thickness, otherwise it has the `\normalrulethickness`. The head position is identical for both pagestyles. However, the headers are empty for *floatcomp* and the footers have centered page numbers on float pages; on ordinary pages the footers are empty while the headers are the same as the *companion* headers.

The code includes one 'trick'. The macro `\makepsmarks{X}{code}` is equivalent to

```
\newcommand{\Xpshook}{code}
```

I have used this knowledge in the line:

```
\makepsmarks{floatcomp}{\companionpshook}
```

which avoids retyping the code from `\makepsmarks{companion}{...}`, and ensures that the code is actually the same for the two pagestyles.

`\mergepagefloatstyle{`⟨*style*⟩`}{`⟨*textstyle*⟩`}{`⟨*floatstyle*⟩`}`

If you have two pre-existing pagestyles, one that will be used for text pages and the other that can be used for float pages, then the `\mergepagefloatstyle` command provides a simpler means of combining them than the above example code for *floatcomp*. The argument ⟨*style*⟩ is the name of the pagestyle being defined. The argument ⟨*textstyle*⟩ is the name of the pagestyle for text pages and ⟨*floatstyle*⟩ is the name of the pagestyle for float-only pages. Both of these must have been defined before calling `\mergepagefloatstyle`. So, instead of the long winded, and possibly tricky, code I could have simply said:

```
\mergepagefloatstyle{floatcomp}{companion}{plain}
```

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

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

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

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

\renewcommand*{\@addtobot}{%
  \@getfpsbit 4\relax
  \ifodd \@tempcnta
    \@flsetnum \@botnum
    \ifnum \@botnum>\z@
      \@tempswafalse
      \@flcheckspace \@botroom \@botlist
      \if@tempswa
        \global \maxdepth \z@
        \@flupdates \@botnum \@botroom \@botlist
        \@inserttrue
  \global\floatatbottrue
      \fi
    \fi
  \fi}

\let\p@wold@output\@outputpage
\renewcommand*{\@outputpage}{%
```

```
    \p@wold@output
    \global\floatattopfalse
    \global\floatatbotfalse}

\endinput
```

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

7.4 THE SHOWLOCS PAGESTYLE

The *showlocs* pagestyle is somewhat special as it is meant to be used as an aid when designing a page layout. Lines are drawn showing the vertical positions of the headers and footers and a box is drawn around the textblock. It is implemented using two zero-sized pictures.[3]

```
\framepichead
\framepictextfoot
\framepichook
\showheadfootlocoff
\showtextblockoff
```

The macro `\framepichead` creates a zero-sized picture that draws a line at the header location, and the macro `\framepictextfoot` creates a zero-sized picture that draws a line at the footer location and also draws a box around the typeblock. Following the declaration `\showheadfootlocoff` the macros `\framepichead` and `\framepictextfoot` do not draw lines showing the header and footer locations. The declaration `\showtextblockoff` prevents `\framepictextfoot` from drawing a box around the textblock.

In case you want to change the color of the *showlocs*, simply do

```
\renewcommnand\framepichook{\color{red}}
```

If you generally want a box around the textblock you may want to create your own pagestyle using `\framepictextfoot` and the *showlocs* code as a starting point, see `memoir.cls` for details.

7.5 OTHER THINGS TO DO WITH PAGE STYLES

Back on page 116 we presented a way of adding some draft information. Here is a more advanced example of this.

One interesting use for page styles is to provide extra information below the footer. This might be some kind of copyright information. Or if your document is under version control with a system like Subversion, and you have all your chapter laying in seperate files, then why not add information at the start of very chapter, specifying who did the last change to this chapter at which time. See the `svn-multi` package ([Sch09]) and the PracTEX Journal article [Sch07] by the same author. Then this information can be added to the start of every chapter using something like:

[3] A zero-sized picture starts off with `begin{picture}(0,0)...`.

```
\usepackage[filehooks]{svn-multi}
\makeatletter
% remember to define a darkgray color
\newcommand\addRevisionData{%
  \begin{picture}(0,0)%
    \put(0,-10){%
      \tiny%
      \expandafter\@ifmtarg\expandafter{\svnfiledate}{}{%
        \textcolor{darkgray}{Chapter last updated
          \svnfileyear/\svnfilemonth/\svnfileday
         \enspace \svnfilehour:\svnfileminute\ (revision \svnfilerev)}
     }%
    }%
  \end{picture}%
}
\makeatother
% chapter is normally an alias to the plain style, we want to change
% it, so make it a real pagestyle
\makepagestyle{chapter}
\makeoddfoot{chapter}{\addRevisionData}{\thepage}{}
\makeevenfoot{chapter}{\addRevisionData}{\thepage}{}
```

Eight

Paragraphs and lists

Within a sectional division the text is typically broken up into paragraphs. Sometimes there may be text that is set off from the normal paragraphing, like quotations or lists.

8.1 Paragraphs

There are basically two parameters that control the appearance of normal paragraphs.

`\parindent` `\parskip`

The length `\parindent` is the indentation of the first line of a paragraph and the length `\parskip` is the vertical spacing between paragraphs, as illustrated in Figure 8.1. The value of `\parskip` is usually 0pt, and `\parindent` is usually defined in terms of ems so that the actual indentation depends on the font being used. If `\parindent` is set to a negative length, then the first line of the paragraphs will be 'outdented' into the lefthand margin.

8.1.1 Block paragraph

A block paragraph is obtained by setting `\parindent` to 0em; `\parskip` should be set to some positive value so that there is some space between paragraphs to enable them to be identified. Most typographers heartily dislike block paragraphs, not only on aesthetical grounds but also on practical considerations. Consider what happens if the last line of a block paragraph is full and also is the last line on the page. The following block paragraph will start at the top of the next page but there will be no identifiable space to indicate an inter-paragraph break.

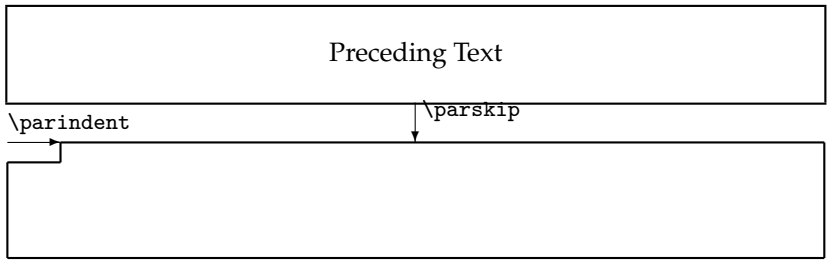


Figure 8.1: Paragraphing parameters

It is important to know that LaTeX typesets paragraph by paragraph. For example, the `\baselineskip` that is used for a paragraph is the value that is in effect at the end of the paragraph, and the font size used for a paragraph is according to the size declaration (e.g., `\large` or `\normalsize` or `\small`) at the end of the paragraph, and the raggedness or otherwise of the whole paragraph depends on the declaration (e.g., `\centering`) in effect at the end of the paragraph. If a pagebreak occurs in the middle of a paragraph TeX will not reset the part of the paragraph that goes onto the following page, even if the textwidths on the two pages are different.

8.1.2 Hanging paragraphs

A hanging paragraph is one where the length of the first few lines differs from the length of the remaining lines. (A normal indented paragraph may be considered to be a special case of a hanging paragraph where 'few = one').

`\hangpara{`⟨*indent*⟩`}{`⟨*num*⟩`}`

Using `\hangpara` at the start of a paragraph will cause the paragraph to be hung. If the length ⟨*indent*⟩ is positive the lefthand end of the lines will be indented but if it is negative the righthand ends will be indented by the specified amount. If the number ⟨*num*⟩, say N, is negative the first N lines of the paragraph will be indented while if N is positive the N+1 th lines onwards will be indented. This paragraph was set with `\hangpara{3em}{-3}`. There should be no space between the `\hangpara` command and the start of the paragraph.

`\begin{hangparas}{`⟨*indent*⟩`}{`⟨*num*⟩`}` text `\end{hangparas}`

The `hangparas` environment is like the `\hangpara` command except that every paragraph in the environment will be hung.

The code implementing the hanging paragraphs is the same as for the `hanging` package [Wil01f]. Examples of some uses can be found in [Thi99].

As noted eleswhere the sectioning commands use the internal macro `\@hangfrom` as part of the formatting of the titles.

`\hangfrom{`⟨*text*⟩`}`

The `\hangfrom` macro is provided as an author's version of the internal `\@hangfrom` macro used, among other things, in typesetting section titles.

Simple hung paragraphs (like this one) can be specified using the `\hangfrom` macro. The macro puts ⟨*text*⟩ in a box and then makes a hanging paragraph of the following material. This paragraph commenced with `\hangfrom{Simple hung paragraphs }(like ...` and you are now reading the result.

The commands for hanging paragraphs do not quite work as might be expected when they are used in a list environment, for example inside an `enumerate`. If you wish for a hanging paragraph inside such an environment you will have to define your own commands for this. If you feel capable of doing so then, with my congratulations, move on to the next section. If you are not so confident you could try using the following non-guaranteed code, which is based on an idea by Patrik Nyman which he posted on CTT in January 2004.

```
%\makeatletter
```

```
% A version of \hangpara for use in a list
%    \listhanging{indent}{num} text text text ...
\def\listhanging#1#2#3\par{%
  \@tempdima\textwidth \advance\@tempdima -\leftmargin
  \parbox{\@tempdima}{\hangpara{#1}{#2}#3}\par}
% A version of \hangfrom for use in a list
%    \listhangfrom{stuff} text text text ...
\def\listhangfrom#1#2\par{%
  \@tempdima\textwidth \advance\@tempdima -\leftmargin
  \parbox{\@tempdima}{\@hangfrom{#1}#2}\par}
%\makeatother
```

8.2 Flush and ragged

Flushleft text has the lefthand end of the lines aligned vertically at the lefthand margin and flushright text has the righthand end of the lines aligned vertically at the righthand margin. The opposites of these are raggedleft text where the lefthand ends are not aligned and raggedright where the righthand end of lines are not aligned. LaTeX normally typesets flushleft and flushright.

`\begin{flushleft}` text `\end{flushleft}`
`\begin{flushright}` text `\end{flushright}`
`\begin{center}` text `\end{center}`

Text in a `flushleft` environment is typeset flushleft and raggedright, while in a `flushright` environment is typeset raggedleft and flushright. In a `center` environment the text is set raggedleft and raggedright, and each line is centered. A small vertical space is put before and after each of these environments.

`\raggedleft` `\raggedright` `\centering`

The `\raggedleft` declaration can be used to have text typeset raggedleft and flushright, and similary the declaration `\raggedright` causes typesetting to be flushleft and raggedright. The declaration `\centering` typesets raggedleft and raggedright with each line centered. Unlike the environments, no additional space is added.

`\centerfloat`

The contents of floats like tables or figures are usually centered and `\centering` should be used for this, not the `center` environment which adds extra, usually undesired, vertical space. For example:

```
\begin{figure}
\centering
...
\caption{...}
\end{figure}
```

However, if the float is wider than the textblock then it is aligned with the left margin and extends into the right margin. The command `\centerfloat` is a special version of `\center` that when used in a wide float will center it with respect to the textblock, i.e., it will extend equally into both margins. Note that `\centerfloat` needs to be applied

Typeset example 8.1: Setting the source of a quotation

This quotation has a short last line so there there is enough space for the source to be set at the end of the line.
I. M. Short

The last line of this quotation turns out to be too long for the source to be set at the end, so it is automatically set flushright on the following line.
N. O. Space

where there is a known width; if applied to a regular text paragraph it will center the paragraph but put all the text on one line.

`\raggedyright[`⟨*space*⟩`]`
`\ragrparindent`

When using `\raggedright` in narrow columns the right hand edge tends to be too ragged, and paragraphs are not indented. Text set `\raggedyright` usually fills more of the available width and paragraphs are indented by `\ragrparindent`, which is initially set to `\parindent`. The optional ⟨*space*⟩ argument, whose default is 2em, can be used to adjust the amount of raggedness. As examples:

```
\raggedyright[0pt]   % typeset flushright
\raggedyright[1fil]  % same as \raggedright
\raggedyright[0.5em] % less ragged than \raggedright
```

Remember that LaTeX typesets on a per-paragraph basis, so that putting the sequence of `\centering`, `\raggedleft` declarations in the same paragraph will cause the entire paragraph to be typeset raggedleft and flushright — the `\centering` declaration is not the one in effect at the end of the paragraph.

8.3 QUOTATIONS

LaTeX provides two environments that are typically used for typesetting quotations.

`\begin{quote}` text `\end{quote}`
`\begin{quotation}` text `\end{quotation}`

In both of these environments the text is set flushleft and flushright in a measure that is smaller than the normal textwidth. The only difference between the two environments is that paragraphs are not indented in the `quote` environment but normal paragraphing is used in the `quotation` environment.

`\sourceatright[`⟨*length*⟩`]{`⟨*text*⟩`}`

Some quotations are completed by giving the source or author. Using `\sourceatright` at the end of the quotation will typeset ⟨*text*⟩ flushright at the end of the line if there is enough space, otherwise it typesets it flushright on the next line. A space ⟨*length*⟩ (default 2em) is left between the end of the quote and ⟨*text*⟩.

Source for example 8.1

```
\begin{quotation}
This quotation has a short last line so there there is enough space
for the source to be set at
the end of the line.\sourceatright{I. M. Short}
\end{quotation}

\begin{quotation}
The last line of this quotation turns out to be too long for
the source to be set at the end, so it is automatically
set flushright on the following line.\sourceatright{N. O. Space}
\end{quotation}
```

8.4 Some less common paragraph shapes

The paragraph shapes described in this section are based on a series that I presented in my *Glisterings* column [Wil07e, Wil08b]. Like the earlier `\centering`, etc., paragraph style declarations, the style that applies is the one in effect at the *end* of the paragraph. Thus the general usage is:

```
\bgroup%    a group to keep changes local % or could be { or \begin...
\paragraphstyle
.... text
\par%       ensure the end of a paragraph
\egroup%    end the group % or could be } or \end...
```

If you use one of these paragraph shapes then using `\\` to break a line may give a surprising result. If so, the following may help.

```
\atcentercr
\break
\memorigdbs
\memorigpar
```

You could try `\atcentcr`, which is user level version of an internal LaTeX command used in some paragraph settings for line breaking, or `\break`, which is a TeX command to end a line.

In some cases the paragraph shaping commands change the definitions of `\\` or `\par`. Just in case you need to restore them, copies of the original definitions are in `\memorigdbs` (for `\\`) and `\memorigpar` (for `\par`).

```
\flushleftright
```

If you use one of the shapes listed later in this section and things go wrong, the declaration `\flushleftright` returns all paragraphing parameters[1] to their normal values, thus producing paragraphs as normal — justified left and right with the last line flushleft and raggedright.

[1] Except for the `\parindent`, which it leaves at its current value.

Typeset example 8.2: Paragraph's line not too short

The last line of this paragraph will be no shorter than a particular length. a b c d e f g h i

The last line of this paragraph will be no shorter than a particular length. a b c d e f g h i j k

8.4.1 Last line not short

On occasion a paragraph may end with a single short word as the last line.

`\linenottooshort[`⟨*length*⟩`]`

Following the `\linenottooshort` declaration paragraphs will be set as normal, except that the last line will not be shorter than ⟨*length*⟩ (default 2em).

Source for example 8.2

```
\linenottooshort[4em]
The last line of this paragraph will be no shorter than a particular
length. a b c d e f g h i % j k l m n

The last line of this paragraph will be no shorter than a particular
length. a b c d e f g h i j k % l m n
```

8.4.2 Russian typography

Apparently in the Russian typographic tradition the last line of a multiline paragraph must either be at least as long as the `\parindent` and have at least `\parindent` at the end, or it must fill the whole line (i.e., flushleft and flushright).

`\russianpar`

Ending a paragraph with `\russianpar` causes it to be set following Russian typographic rules.

If you have many such paragraphs it may be more convenient to do it like:

```
\let\par\russianpar
... many paragraphs
\let\par\memorigpar
```

or as:

```
\begingroup% start a group
\let\par\russianpar
... many paragraphs
\endgroup% end the group
```

Typeset example 8.3: Rules for spaces

The last line of this paragraph will be be set by ending it with a rule to fill up any space.

8.4.3 Fill with rules

In some legal documents there must be no space at the end of the lines in order to prevent anyone inserting something at a later date. Typically it is only the last line in a paragraph that needs this treatment.

```
\lastlinerulefill
\lastlineparrule
```

Source for example 8.3

```
The last line of this paragraph will be be set by ending it with
a rule to fill up any space.\lastlinerulefill
```

Using `\lastlinerulefill` to end a paragraph will cause any spaces at the ends of the lines to be filled with the `\lastlineparrule` rule. If you have many paragraphs of this kind then try:

```
\let\par\lastlinerulefill
.... many paragraphs
\let\par\memorigpar
```

Remember that LaTeX treats many constructs (like section headings or captions) as paragraphs, so you may have to alternate between filled text paragraphs and regular paragraphing.

8.4.4 Some ragged paragraphs

A few paragraph shapes with unusual ragged lines are available.

```
\justlastraggedleft
\raggedrightthenleft
\leftcenterright
```

Following the `\justlastraggedleft` declaration paragraphs will be set justified except the last line will be set raggedleft.

Following the declaration `\raggedrightthenleft` paragraphs will be set with the first line raggedright and the remainder set raggedleft.

Following the declaration `\leftcenteright` paragraphs will be set with the first line flushleft (and raggedright) and the last line flushright (and raggedleft) and those in the middle will be centered. This declaration should be used within a group; also `\everypar{}` should be called at the end.

Typeset example 8.4: Ragged paragraphs

Paragraphs following the `\justlastraggedleft` declaration, as this one does, have their lines justified except for the last which is set raggedleft. The demonstration works best if there are three or more lines.

This paragraph is set following the `\raggedrightthenleft` declaration. The first line is set raggedright and all the remaining lines are set raggedleft. The demonstration is better if there are three or more lines.

This paragraph is set following the `\leftcenterright` declaration. We really need three,
or four may be better,
lines to show the effect of this.

Source for example 8.4

```
\justlastraggedleft
Paragraphs following the \verb?\justlastraggedleft? declaration, as
this one does, have their lines justified except for the last which
is set raggedleft. The demonstration works best if there are three
or more lines.

\raggedrightthenleft
This paragraph is set following the \verb?\raggedrightthenleft?
declaration. The first line is set raggedright and all the remaining
lines are set raggedleft. The demonstration is better if there are three or
more lines.

\leftcenterright
This paragraph is set following the \verb?\leftcenterright?
declaration. We really need three, \\ or four may be better, \\
lines to show the effect of this.
\everypar{}
```

8.4.5 Left spring right

Typically the lines of a paragraph are both flushleft and flushright and filled with text, but sometimes filling is not desired.

`\leftspringright{`⟨*lfrac*⟩`}{`⟨*rfrac*⟩`}{`⟨*ltext*⟩`}{`⟨*rtext*⟩`}`

The `\leftspringright` macro sets ⟨*ltext*⟩ flushleft and raggedright in a column whose width is ⟨*lfrac*⟩ of the textwidth and, in parallel, it also sets ⟨*rtext*⟩ raggedleft and flushright in a column that is ⟨*rfrac*⟩ of the textwidth; the effect is as though there are springs between the lines of the two texts. The sum of ⟨*lfrac*⟩ and ⟨*rfac*⟩ must be less than one.

Typeset example 8.5: A sprung paragraph

Text at the left is set flushleft and raggedright.

But the text at the right is set raggedleft and flushright. It's as though there was a spring pushing the lines apart.

Source for example 8.5

```
\leftspringright{0.3}{0.6}%
  {Text at the left is set flushleft and raggedright.}
  {But the text at the right is set raggedleft and flushright.
   It's as though there was a spring pushing the lines apart.}
```

8.5 Changing the textwidth

The `quote` and `quotation` environments both locally change the textwidth, or more precisely, they temporarily increase the left and right margins by equal amounts. Generally speaking it is not a good idea to change the textwidth but sometimes it may be called for.

The commands and environment described below are similar to those in the originally found in the chngpage package, but do differ in some respects.

`\begin{adjustwidth}{`⟨*left*⟩`}{`⟨*right*⟩`}` text `\end{adjustwidth}`
`\begin{adjustwidth*}{`⟨*left*⟩`}{`⟨*right*⟩`}` text `\end{adjustwidth*}`

The `adjustwidth` environment temporarily adds the length ⟨*left*⟩ to the lefthand margin and ⟨*right*⟩ to the righthand margin. That is, a positive length value increases the margin and hence reduces the textwidth, and a negative value reduces the margin and increases the textwidth. The `quotation` environment is roughly equivalent to

```
\begin{adjustwidth}{2.5em}{2.5em}
```

The starred version of the environment, `adjustwidth*`, is really only useful if the left and right margin adjustments are different. The starred version checks the page number and if it is odd then adjusts the left (spine) and right (outer) margins by ⟨*left*⟩ and ⟨*right*⟩ respectively; if the page number is even (a verso page) it adjusts the left (outer) and right (spine) margins by ⟨*right*⟩ and ⟨*left*⟩ respectively.

`\strictpagecheck` `\easypagecheck`

Odd/even page checking may be either strict (`\strictpagecheck`) or easy (or one might call it lazy) (`\easypagecheck`). Easy checking works most of the time but if it fails at any point then the strict checking should be used.

As an example, if a figure is wider than the textwidth it will stick out into the righthand margin. It may be desireable to have any wide figure stick out into the outer margin where there is usually more room than at the spine margin. This can be accomplished by

```
\begin{figure}
\centering
\strictpagecheck
\begin{adjustwidth*}{0em}{-3em}
% the illustration
\caption{...}
\end{adjustwidth*}
\end{figure}
```

A real example in this manual is Table 10.1 on page 185, which is wider than the typeblock. In that case I just centered it by using `adjustwidth` to decrease each margin equally. In brief, like

```
\begin{table}
\begin{adjustwidth}{-1cm}{-1cm}
\centering
...
\end{adjustwidth}
\end{table}
```

Note that the `adjustwidth` environment applies to complete paragraphs; you can't change the width of part of a paragraph except for hanging paragraphs or more esoterically via `\parshape`. Further, if the adjusted paragraph crosses a page boundary the margin changes are constant; a paragraph that is, say, wider at the right on the first page will also be wider at the right as it continues onto the following page.

The `center` environment horizontally centers its contents with respect to the typeblock. Sometimes you may wish to horizontally center some text with respect to the physical page, for example when typesetting a colophon which may look odd centered with respect to the (unseen) typeblock.

The calculation of the necessary changes to the spine and fore-edge margins is simple. Using the same symbols as earlier in §2.4 (P_w and B_w are the width of the trimmed page and the typeblock, respectively; S and E are the spine and fore-edge margins, respectively) then the amount M to be added to the spine margin and subtracted from the fore-edge margin is calculated as:

$$M = 1/2(P_w - B_w) - S$$

For example, assume that the `\textwidth` is 5 inches and the `\spinemargin` is 1 inch. On US letterpaper (`\paperwidth` is 8.5 inches) the fore-edge margin is then 2.5 inches, and 0.75 inches[2] must be added to the spine margin and subtracted from the fore-edge to center the typeblock. The `adjustwidth` environment can be used to make the (temporary) change.

```
\begin{adjustwidth*}{0.75in}{-0.75in} ...
```

`\calccentering{`⟨*length*⟩`}`

[2] On A4 paper the result would be different.

If you don't want to do the above calculations by hand, `\calccentering` will do it for you. The ⟨*length*⟩ argument must be the name of a pre-existing length command, including the backslash. After calling `\calccentering`, ⟨*length*⟩ is the amount to be added to the spine margin and subtracted from the foredge margin to center the typeblock. An example usage is

```
\calccentering{\mylength}
\begin{adjustwidth*}{\mylength}{-\mylength}
text horizontally centered on the physical page
\end{adjustwidth*}
```

You do not necessarily have to define a new length for the purposes of `\calccentering`. Any existing length will do, such as `\unitlength`, provided it will be otherwise unused between performing the calculation and changing the margins (and that you can, if necessary reset it to its original value — the default value for `\unitlength` is 1pt).

8.6 Lists

Standard LaTeX provides four kinds of lists. There is a general `list` environment which you can use to define your own particular kind of list, and the `description`, `itemize` and `enumerate` lists (which are internally defined in terms of the general `list` environment[3]).

This class provides the normal `description` list, plus a couple of others of the same kind, but the `itemize` and `enumerate` lists are extended versions of the normal ones.

`\begin{description}` `\item[`⟨*label*⟩`]` ... `\end{description}`
`\begin{blockdescription}` `\item[`⟨*label*⟩`]` ... `\end{blockdescription}`
`\descriptionlabel`⟨*label*⟩
`\blockdescriptionlabel`⟨*label*⟩

In a `description` list an `\item`'s ⟨*label*⟩ is typeset by `descriptionlabel`. The default definition is

```
\newcommand*{\descriptionlabel}[1]{\hspace\labelsep
                                   \normalfont\bfseries #1}
```

which gives a bold label. To have, for example, a sans label instead, do

```
\renewcommand*{\descriptionlabel}[1]{\hspace\labelsep
                                     \normalfont\sffamily #1}
```

The only noticeable difference between a `description` list and a `blockdescription` list is that the latter is set as indented block paragraphs; invisibly, it also has its own `\blockdescriptionlabel`.

`\begin{labelled}{`⟨*name*⟩`}` `\item[`⟨*label*⟩`]` ... `\end{labelled}`
`\begin{flexlabelled}{`⟨*name*⟩`}{`⟨*labelwidth*⟩`}{`⟨*labelsep*⟩`}{`⟨*itemindent*⟩`}%`
`{`⟨*leftmargin*⟩`}{`⟨*rightmargin*⟩`}`
`\item[`⟨*label*⟩`]` ... `\end{flexlabelled}`

[3]The `quote` and `quotation` environments are also defined in terms of the general `list` environment. You may be surprised where it crops up.

.

The `labelled` environment is like the `description` environment except that you can specify the label format via the ⟨*name*⟩ argument where `\name` is the formatting macro. For example, if you wanted the item labels set in italics, then

```
\newcommand*{\itlabel}[1]{\hspace\labelsep \normalfont\itshape #1}
\begin{labelled}{itlabel}
\item[First] ...
...
```

The `flexlabelled` environment adds additional controls to the `labelled` one. The ⟨*name*⟩ argument is the same as that for `labelled` and the remainder are lengths that correspond to the dimensions shown in Figure 8.2. If you want any of the dimensions to retain their current values, use `*` instead of a length as the value for that particular argument.

Source for example 8.6

```
This example shows how the \texttt{flexlabelled} list can be used to
change the formatting of a description-like list.
\newcommand*{\sclabel}[1]{\normalfont\scshape #1}
\begin{flexlabelled}{sclabel}{0pt}{0.5em}{0.5em}{*}{\leftmargin}
\item[First] The labels should be typeset using smallcaps and the first
             paragraph should be set as block paragraph.

             Further paragraphs completing an \cs{item}'s descriptive text
             will be set with the normal paragraph indent.
\item[Second] The list should be indented from each margin like the
              \texttt{quote} and \texttt{quotation} environments.
\end{flexlabelled}
More major changes to a description-like list will probably involve writing
the code for a new environment.
```

The `itemize` and `enumerate` environments below are based on the enumerate package [Car98c].

`\begin{itemize}[`⟨*marker*⟩`] \item ... \end{itemize}`

The normal markers for `\items` in an `itemize` list are:

1. bullet (•`\textbullet`),
2. bold en-dash (– `\bfseries\textendash`),
3. centered asterisk (∗`\textasteriskcentered`), and
4. centered dot (·`\textperiodcentered`).

The optional ⟨*marker*⟩ argument can be used to specify the marker for the list items in a particular list. If for some reason you wanted to use a pilcrow symbol as the item marker for a particular list you could do

```
\begin{itemize}[\P]
\item ...
```

Typeset example 8.6: Smallcap quote style description list

This example shows how the `flexlabelled` list can be used to change the formatting of a description-like list.

FIRST The labels should be typeset using smallcaps and the first paragraph should be set as block paragraph.

Further paragraphs completing an `\item`'s descriptive text will be set with the normal paragraph indent.

SECOND The list should be indented from each margin like the `quote` and `quotation` environments.

More major changes to a description-like list will probably involve writing the code for a new environment.

...

`\begin{enumerate}[`⟨*style*⟩`] \item` ... `\end{enumerate}`

The normal markers for, say, the third item in an `enumerate` list are: 3., c., iii., and C. The optional ⟨*style*⟩ argument can be used to specify the style used to typeset the item counter. An occurrence of one of the special characters `A`, `a`, `I`, `i` or `1` in ⟨*style*⟩ specifies that the counter will be typeset using uppercase letters (`A`), lowercase letters (`a`), uppercase Roman numerals (`I`), lowercase Roman numerals (`i`), or arabic numerals (`1`). These characters may be surrounded by any LaTeX commands or characters, but if so the special characters must be put inside braces (e.g., `{a}`) if they are to be considered as ordinary characters instead of as special styling characters. For example, to have the counter typeset as a lowercase Roman numeral followed by a single parenthesis

```
\begin{enumerate}[i)]
...
```

Recommended alternative

memoir does not provide high level interfaces to configure the appearance. We provide some simple tools to adjust vertical spacing, see below.

Users seeking more control can have a look at the excellent enumitem package by Javier Bezos. If loaded as

```
\usepackage[shortlabels]{enumitem}
```

then our

```
\begin{enumerate}[i)]
 \item \label{item:tst}  ...
```

syntax will work out of the box.

One key difference: In memoir `\ref{item:tst}` will give you 'i', whereas, if enumitem is loaded the full formatting is returned from the cross reference, i.e., 'i)'. This is fully configurable in enumitem.

Note that, `\tightlists`, `\defaultlists`, `\firmlists`, `\firmlists*` presented below, are not supported by enumitem, it provides a highlevel key based interface instead.

Another feature from enumitem that I (LM) uses a lot is to combine the `\setlist` with `\AtBeginEnvironment` from the etoolbox package to specifically adjust enumerate used inside certain theorem constructions. That wway we can control the apparence of the lists from the preamble and does not need to use say

```
\begin{enumerate}[(a)]
```

every single time.

`\tightlists` `\defaultlists`
`\firmlists` `\firmlists*`

The normal LaTeX `description`, `itemize` and `enumerate` lists have an open look about them when they are typeset as there is significant vertical space between the items in the lists. After the declaration `\tightlists` is issued, the extra vertical spacing between the list items is deleted. The open list appearance is used after the `\defaultlists` declaration is issued. These declarations, if used, must come *before* the relevant list environment(s). The class initially sets `\defaultlists`. This manual, though, uses `\tightlists`. The spacing following the `\firmlists` declaration is intermediate between `\defaultlists` and `\tightlists`. The starred version, `\firmlists*`, allows sligthly less space around the lists when they are preceded by a blank line than does the unstarred `\firmlists`.

Caveat. Due to the manner in which `\small` and `\footnotesize` are implemented, `\tightlists` and `\firmlists` will have *no effect* on lists typeset under `\small` or `\footnotesize`.

A comprehensible solution can be done via the enumitem package via

```
% \tightlists equivalent
\usepackage[shortlabels]{enumitem}
\setlist{ noitemsep }
```

`\firmlist` `\tightlist`

The command `\firmlist` or `\tightlist` can be used immediately after the start of a list environment to reduce the vertical space within that list. The `\tightlist` removes all the spaces while the `\firmlist` produces a list that still has some space but not as much as in an ordinary list.

`\begin{list}{`⟨*default-label*⟩`}{`⟨*code*⟩`}` items `\end{list}`

LaTeX's list environments are defined in terms of a general `list` environment; some other environments, such as the `quote`, `quotation` and `adjustwidth` are also defined in terms of a `list`. Figure 8.2 shows the parameters controlling the layout of the `list` environment.

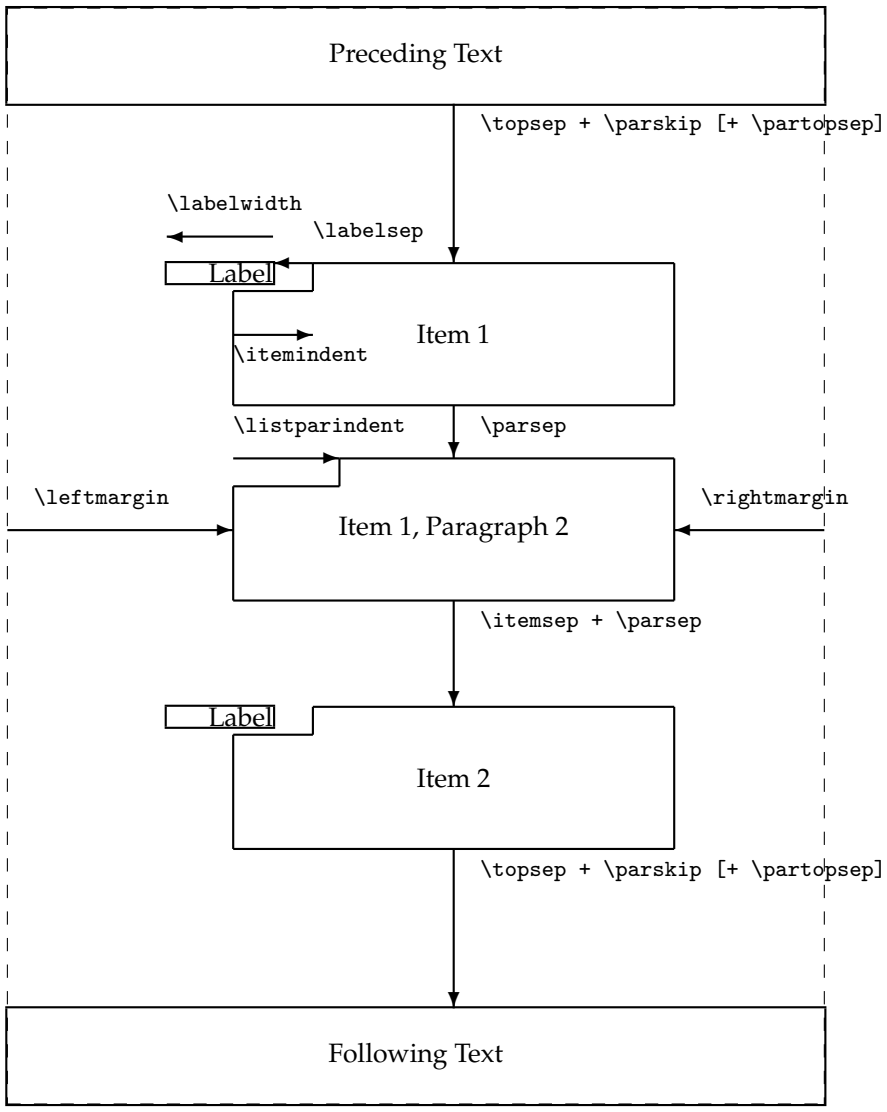


Figure 8.2: The layout parameters for general lists

The `list` environment takes two arguments. The ⟨*default-label*⟩ argument is the code that should be used when the `\item` macro is used without its optional ⟨*label*⟩ argument. For lists like `enumerate` this is specified but often it is left empty, such as for the `adjustwidth` environment.

The ⟨*code*⟩ argument is typically used for setting the particular values of the list layout parameters. When defining your own types of lists it is advisable to set each of the parameters unless you know that the default values are suitable for your purposes. These parameters can all be modified with either the `\setlength` or `\addtolength` commands.

As an example, here is the specification for a description-like list that uses an italic rather than bold font for the items, and is somewhat tighter than the normal `description` list.

```
%%%%% An italic and tighter description environment
\newcommand{\itlabel}[1]{\hspace\labelsep\normalfont\itshape #1}
\newenvironment{itdesc}{%
  \list{}{%
    \setlength{\labelsep}{0.5em}
    \setlength{\itemindent}{0pt}
    \setlength{\leftmargin}{\parindent}
    \setlength{\labelwidth}{\leftmargin}
    \addtolength{\labelwidth}{-\labelsep}
    \setlength{\listparindent}{\parindent}
    \setlength{\parsep}{\parskip}
    \setlength{\itemsep}{0.5\onelineskip}
    \let\makelabel\itlabel}}{\endlist}
```

This gets used like any other list:

```
\begin{itdesc}
\item[label] ....
\end{itdesc}
```

Here is another kind of list called `symbols` that might be used for a list of symbols or other similar kind of listing.

```
% Symbol list
\newenvironment{symbols}%
               {\list{}% empty label
                            {\setlength{\topsep}{\baselineskip}
                             \setlength{\partopsep}{0pt}
                             \setlength{\itemsep}{0.5\baselineskip}
                             \setlength{\parsep}{0pt}
                             \setlength{\leftmargin}{2em}
                             \setlength{\rightmargin}{0em}
                             \setlength{\listparindent}{1em}
                             \setlength{\itemindent}{0em}
                             \setlength{\labelwidth}{0em}
                             \setlength{\labelsep}{2em}}}%
               {\endlist}
\newcommand{\symb}[1]{\item[#1]\mbox{}\\\nopagebreak}
```

Typeset example 8.7: Changing space before and after lists

This example shows that the space around the

CENTER AND OTHER LIST ENVIRONMENTS

can be minimised by using the
\zerotrivseps declaration.
The normal spacing can be restored by using the

\restoretrivseps command.

An alternative is to use the \centering macro.

In this case it gets used like this

```
\begin{symbols}
\symb{SYMBOL 1} definition
\symb{SYMBOL 2} ...
\end{symbols}
```

In the code for the symbols list I used the command forms (i.e., \list and \endlist) for specifying the start and end of a list. It is a matter of taste whether you use the command or \begin{...} and \end{...} forms, but the latter does make it more obvious that an environment is being dealt with.

Several LaTeX environments are defined in terms of a very simple list, called a trivlist. Such a list has little internal structure but like the list environment the vertical space before and after a trivlist (or any list based on it) is set by \topsep and \partopsep, as shown in Figure 8.2.

\zerotrivseps \savetrivseps \restoretrivseps

The center environment is one of several that is based on a trivlist, and so has space before and after it. If you don't want this the \zerotrivseps declaration eliminates those spaces. You can think of it as being defined as:

```
\newcommand*{\zerotrivseps}{%
  \setlength{\topsep}{0pt}%
  \setlength{\partopsep}{0pt}}
```

Before doing this, though, you might consider calling \savetrivseps which stores the current values of \topsep and \partopsep; it is initially defined to store the default values. The command \restoretrivseps sets the current values of these lengths to the ones saved by \savetrivseps.

Source for example 8.7

```
This example shows that the space around the
```

```
\begin{center}
CENTER AND OTHER LIST ENVIRONMENTS
\end{center}
can be minimised by using the \zerotrivseps
\begin{center}
\verb?\zerotrivseps? declaration.
\end{center}
The normal spacing can be restored by using the \restoretrivseps
\begin{center}
\verb?\restoretrivseps? command.
\end{center}
An alternative is to use the \verb?\centering? macro.
```

Among the environments defined in terms of a `trivlist` are: `flushleft`, `center`, `flushright`, `verbatim`, and others. The example (8.7) shows how it is possible to change the spacing around the `center` environment, but it applies equally to the other environments.

Nine

Contents lists

This chapter describes how to change the appearance of the Table of Contents (ToC) and similar lists like the List of Figures (LoF). In the standard classes the typographic design of these is virtually fixed as it is buried within the class definitions.

As well as allowing these lists to appear multiple times in a document, the memoir class gives handles to easily manipulate the design elements. The class also provides means for you to define your own new kinds of "List of…".

The functionality described is equivalent to the combination of the tocloft and tocbibind packages [Wil01i, Wil01h].

`\tableofcontents` `\tableofcontents*`
`\listoffigures` `\listoffigures*`
`\listoftables` `\listoftables*`

The commands `\tableofcontents`, `\listoffigures` and `\listoftables` typeset, repectively, the Table of Contents (ToC), List of Figures (LoF) and List of Tables (LoT). In memoir, unlike the standard classes, the unstarred versions add their respective titles to the ToC. The starred versions act like the standard classes' unstarred versions as they don't add their titles to the ToC.

This chapter explains the inner workings behind the ToC and friends, how to change their appearance and the apperance of the entries, and how to create new 'List of…'. If you don't need any of these then you can skip the remainder of the chapter.

9.1 General ToC methods

In §9.2 we will provide the class configuration interface for the various parts of the ToC.

In order to understand how these macros are used, we start by providing some background information this is a general description of how the standard LaTeX classes process a Table of Contents (ToC). As the processing of List of Figures (LoF) and List of Tables (LoT) is similar I will just discuss the ToC. You may wish to skip this section on your first reading.

The basic process is that each sectioning command writes out information about itself — its number, title, and page — to the `toc` file. The `\tableofcontents` command reads this file and typesets the contents.

First of all, remember that each sectional division has an associated level as listed in in Table 6.1 on page 79. LaTeX will not typeset an entry in the ToC unless the value of the `tocdepth` counter is equal to or greater than the level of the entry. The value of the `tocdepth` counter can be changed by using `\setcounter` or `\settocdepth`.

`\addcontentsline{`⟨*file*⟩`}{`⟨*kind*⟩`}{`⟨*text*⟩`}`

LaTeX generates a `toc` file if the document contains a `\tableofcontents` command. The sectioning commands[1] put entries into the `toc` file by calling the `\addcontentsline` command, where ⟨*file*⟩ is the file extension (e.g., `toc`), ⟨*kind*⟩ is the kind of entry (e.g., `section` or `subsection`), and ⟨*text*⟩ is the (numbered) title text. In the cases where there is a number, the ⟨*text*⟩ argument is given in the form `{\numberline{number}title text}`.

`\contentsline{`⟨*kind*⟩`}{`⟨*text*⟩`}{`⟨*page*⟩`}`

The `\addcontentsline` command writes an entry to the given file in the form:
`\contentsline{`⟨*kind*⟩`}{`⟨*text*⟩`}{`⟨*page*⟩`}`
where ⟨*page*⟩ is the page number.

For example, if `\section{Head text}` was typeset as '**3.4 Head text**' on page 27, then there would be the following entry in the `toc` file:

```
\contentsline{section}{\numberline{3.4} Head text}{27}
```

Extracts from `toc`, `lof` and `lot` files are shown in Figure 9.1.

For each ⟨*kind*⟩ that might appear in a `toc` (`lof`, `lot`) file, LaTeX provides a command:
`\l@kind{`⟨*title*⟩`}{`⟨*page*⟩`}`
which performs the actual typesetting of the `\contentsline` entry.

`\@pnumwidth{`⟨*length*⟩`}`
`\@tocrmarg{`⟨*length*⟩`}`
`\@dotsep{`⟨*number*⟩`}`

The general layout of a typeset entry is illustrated in Figure 9.2. There are three internal LaTeX commands that are used in the typesetting. The page number is typeset flushright in a box of width `\@pnumwidth`, and the box is at the righthand margin. If the page number is too long to fit into the box it will stick out into the righthand margin. The title text is indented from the righthand margin by an amount given by `\@tocrmarg`. Note that `\@tocrmarg` should be greater than `\@pnumwidth`. Some entries are typeset with a dotted leader between the end of the title title text and the righthand margin indentation. The distance, in *math units*[2] between the dots in the leader is given by the value of `\@dotsep`. In the standard classes the same values are used for the ToC, LoF and the LoT.

The standard values for these internal commands are:

- `\@pnumwidth` = 1.55em
- `\@tocrmarg` = 2.55em
- `\@dotsep` = 4.5

The values can be changed by using `\renewcommand`, in spite of the fact that the first two appear to be lengths.

Dotted leaders are not available for Part and Chapter ToC entries.

`\numberline{`⟨*number*⟩`}`

Each `\l@kind` macro is responsible for setting the general *indent* from the lefthand margin, and the *numwidth*. The `\numberline` macro is responsible for typesetting the number flushleft in a box of width *numwidth*. If the number is too long for the box then it will protrude into the title text. The title text is indented by (*indent* + *numwidth*) from the lefthand

[1] For figures and tables it is the `\caption` command that populates the `lof` and `lot` files.
[2] There are 18mu to 1em.

Parts of a `toc` file:

```
...
\contentsline{section}{\numberline{10.1}The spread}{77}
\contentsline{section}{\numberline{10.2}Typeblock}{89}
\contentsline{subsection}{\numberline{10.2.1}Color}{77}
...
\contentsline{chapter}{Index}{226}
```

Part of a `lof` file:

```
...
\contentsline{figure}{\numberline{8.6}Measuring scales}{56}
\addvspace{10pt}
\addvspace{10pt}
\contentsline{figure}{\numberline{10.1}Two subfigures}{62}
\contentsline{subfigure}{\numberline{(a)}Subfigure 1}{62}
\contentsline{subfigure}{\numberline{(b)}Subfigure 2}{62}
...
```

Part of a `lot` file:

```
...
\contentsline{table}{\numberline{1.7}Font declarations}{11}
\contentsline{table}{\numberline{1.8}Font sizes}{13}
\addvspace
\contentsline{table}{\numberline{3.1}Division levels}{21}
...
```

Figure 9.1: Example extracts from `toc`, `lof` and `lot` files

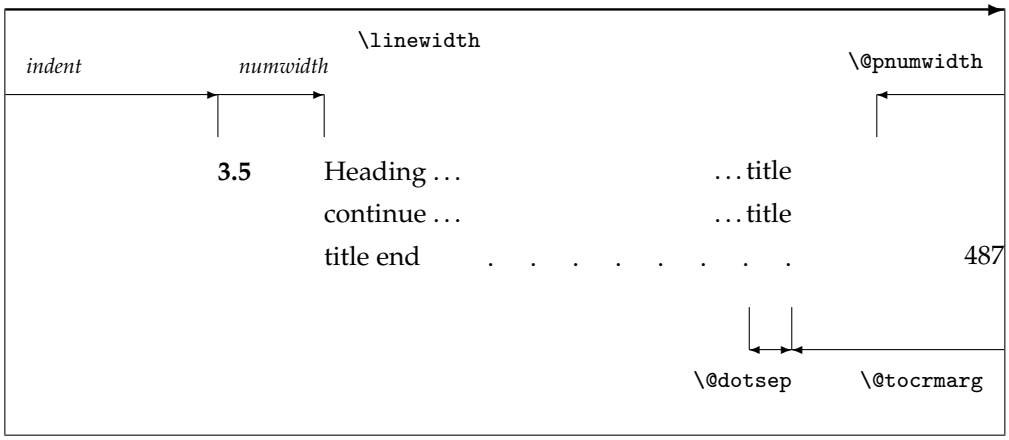


Figure 9.2: Layout of a ToC (LoF, LoT) entry

Table 9.1: Indents and Numwidths (in ems)

Entry	Level	Standard		memoir class	
		indent	numwidth	indent	numwidth
book	-2	—	—	0	—
part	-1	0	—	0	1.5
chapter	0	0	1.5	0	1.5
section	1	1.5	2.3	1.5	2.3
subsection	2	3.8	3.2	3.8	3.2
subsubsection	3	7.0	4.1	7.0	4.1
paragraph	4	10.0	5.0	10.0	5.0
subparagraph	5	12.0	6.0	12.0	6.0
figure/table	(1)	1.5	2.3	0	1.5
subfigure/table	(2)	—	—	1.5	2.3

margin. That is, the title text is typeset in a block of width
(`\linewidth` - *indent* - *numwidth* - `\@tocrmarg`).

Table 9.1 lists the standard values for the *indent* and *numwidth*. There is no explicit *numwidth* for a part; instead a gap of 1em is put between the number and the title text. Note that for a sectioning command the values depend on whether or not the document class provides the `\chapter` command; the listed values are for the book and report classes — in the article class a `\section` is treated like a `\chapter`, and so on. Also, which somewhat surprises me, the table and figure entries are all indented.

`\@dottedtocline{`⟨*level*⟩`}{`⟨*indent*⟩`}{`⟨*numwidth*⟩`}`

Most of the `\l@kind` commands are defined in terms of the `\@dottedtocline` command. This command takes three arguments: the ⟨*level*⟩ argument is the level as shown in Table 9.1, and ⟨*indent*⟩ and ⟨*numwidth*⟩ are the *indent* and *numwidth* as illustrated in Figure 9.2. For example, one definition of the `\l@section` command is:

```
\newcommand*{\l@section}{\@dottedtocline{1}{1.5em}{2.3em}}
```

If it is necessary to change the default typesetting of the entries, then it is usually necessary to change these definitions, but memoir gives you handles to easily alter things without having to know the LaTeX internals.

You can use the `\addcontentsline` command to add `\contentsline` commands to a file.

`\addtocontents{`⟨*file*⟩`}{`⟨*text*⟩`}`

LaTeX also provides the `\addtocontents` command that will insert ⟨*text*⟩ into ⟨*file*⟩. You can use this for adding extra text and/or macros into the file, for processing when the file is typeset by `\tableofcontents` (or whatever other command is used for ⟨*file*⟩ processing, such as `\listoftables` for a `lot` file).

As `\addcontentsline` and `\addtocontents` write their arguments to a file, any fragile commands used in their arguments must be `\protected`.

You can make certain adjustments to the ToC, etc., layout by modifying some of the above macros. Some examples are:

- If your page numbers stick out into the righthand margin

```
\renewcommand{\@pnumwidth}{3em}
\renewcommand{\@tocrmarg}{4em}
```

  but using lengths appropriate to your document.
- To have the (sectional) titles in the ToC, etc., typeset ragged right with no hyphenation

```
\renewcommand{\@tocrmarg}{2.55em plus1fil}
```

  where the value `2.55em` can be changed for whatever margin space you want.
- The dots in the leaders can be eliminated by increasing `\@dotsep` to a large value:

```
\renewcommand{\@dotsep}{10000}
```

- To have dotted leaders in your ToC and LoF but not in your LoT:

```
...
\tableofcontents
\makeatletter \renewcommand{\@dotsep}{10000} \makeatother
\listoftables
\makeatletter \renewcommand{\@dotsep}{4.5} \makeatother
\listoffigures
...
```

- To add a horizontal line across the whole width of the ToC below an entry for a Part:

```
\part{Part title}
\addtocontents{toc}{\protect\mbox{}\protect\hrulefill\par}
```

  As said earlier any fragile commands in the arguments to `\addtocontents` and `\addcontentsline` must be protected by preceding each fragile command with `\protect`. The result of the example above would be the following two lines in the `.toc` file (assuming that it is the second Part and is on page 34):

```
\contentsline {part}{II\hspace {1em}Part title}{34}
\mbox {}\hrulefill \par
```

  If the `\protect`s were not used, then the second line would instead be:

```
\unhbox \voidb@x \hbox {}\unhbox \voidb@x \leaders \hrule \hfill
        \kern \z@ \par
```

  which would cause LaTeX to stop and complain because of the commands that included the `@` (see §E.4). If you are modifying any command that includes an `@` sign then this must be done in either a `.sty` file or if in the document itself it must be surrounded by `\makeatletter` and `\makeatother`. For example, if you want to modify `\@dotsep` in the preamble to your document you have to do it like this:

```
\makeatletter
\renewcommand{\@dotsep}{9.0}
\makeatother
```

- To change the level of entries printed in the ToC (for example when normally subsections are listed in the ToC but for appendices only the main title is required)

```
\appendix
\addtocontents{toc}{\protect\setcounter{tocdepth}{0}}
\chapter{First appendix}
...
```

9.2 THE CLASS TOC METHODS

The class provides various means of changing the look of the ToC, etc., without having to go through some of the above.

\tableofcontents \tableofcontents*
\listoffigures \listoffigures*
\listoftables \listoftables*

The ToC, LoF, and LoT are printed at the point in the document where these commands are called, as per normal LaTeX. You can use `\tableofcontents`, `\listoffigures`, etc., more than once in a memoir class document.

However, there are two differences between the standard LaTeX behaviour and the behaviour with this class. In the standard LaTeX classes that have `\chapter` headings, the ToC, LoF and LoT each appear on a new page. With this class they do not necessarily start new pages; if you want them to be on new pages you may have to specifically issue an appropriate command beforehand. For example:

```
 ...
\clearpage
\tableofcontents
\clearpage
\listoftables
...
```

Also, the unstarred versions of the commands put their headings into the ToC, while the starred versions do not.

\begin{KeepFromToc} \listof... \end{KeepFromToc}

There is at least one package that uses `\tableofcontents` for its own 'List of...'. When used with the class this will put the package's 'List of...' title into the ToC, and the package doesn't seem to know about `\tableofcontents*`. The heading of any `\listof...` command that is in the `KeepFromToc` environment will not be added to the ToC. For example:

```
\begin{KeepFromToc}
\listoffigures
\end{KeepFromToc}
```

is equivalent to `\listoffigures*`.

\onecoltocetc
\twocoltocetc
\doccoltocetc

Table 9.2: Values for `X` in macros for styling the titles of 'List of...'

`toc`	`lof`	`lot`	`...`

In the standard classes the ToC, etc., are set in one column even if the document as a whole is set in two columns. This limitation is removed. Following the `\onecoltocetc` declaration, which is the default, the ToC and friends will be set in one column but after the `\twocoltocetc` declaration they will be set in two columns. Following the `\doccoltocetc` declaration they will be set in either one or two columns to match the document class onecolumn or twocolumn option.

`\maxtocdepth{`⟨*secname*⟩`}`
`\settocdepth{`⟨*secname*⟩`}`

The class `\maxtocdepth` command sets the `tocdepth` counter. It is currently not used in the memoir class.

The memoir class command `\settocdepth` is somewhat analagous to the `\setsecnumdepth` command described in §6.3. It sets the value of the `tocdepth` counter and puts it into the ToC to (temporarily) modify what will appear. The `\settocdepth` and `\maxtocdepth` macros are from the tocvsec2 package [Wil99b].

`\phantomsection`

NOTE: The hyperref package [Rahtz02] appears to dislike authors using `\addcontentsline`. To get it to work properly with hyperref you normally have to put `\phantomsection` (a macro defined within this class and the hyperref package) immediately before `\addcontentsline`.

9.2.1 Changing the titles

Commands are provided for controlling the appearance of the ToC, LoF and LoT titles.

`\contentsname` `\listfigurename` `\listtablename`

Following LaTeX custom, the title texts are the values of the `\contentsname`, `\listfigurename` and `\listtablename` commands.

The commands for controlling the typesetting of the ToC, LoF and LoT titles all follow a similar pattern, so for convenience (certainly mine, and hopefully yours) in the following descriptions I will use `X`, as listed in Table 9.2, to stand for the file extension for the appropriate 'List of...'. That is, any of the following:

- `toc` or
- `lof` or
- `lot`.

For example, `\Xmark` stands for `\tocmark` or `\lofmark` or `\lotmark`.

The code for typesetting the ToC title looks like:

```
\tocheadstart
\printtoctitle{\contentsname}
\tocmark
\thispagestyle{chapter}
```

```
\aftertoctitle
```

where the macros are described below.

`\Xheadstart`

This macro is called before the title is actually printed. Its default definition is

```
\newcommand{\Xheadstart}{\chapterheadstart}
```

`\printXtitle{⟨title⟩}`

The title is typeset via `\printXtitle`, which defaults to using `\printchaptertitle` for the actual typesetting.

`\Xmark`

These macros sets the marks for use by the running heads on the ToC, LoF, and LoT pages. The default definition is equivalent to:

```
\newcommand{\Xmark}{\markboth{\...name}{\...name}}
```

where `\...name` is `\contentsname` or `\listfigurename` or `\listtablename` as appropriate. You probably don't need to change these, and in any case they may well be changed by the particular `\pagestyle` in use.

`\afterXtitle`

This macro is called after the title is typeset and by default it is defined to be `\afterchaptertitle`.

Essentially, the ToC, LoF and LoT titles use the same format as the chapter titles, and will be typeset according to the current chapterstyle. You can modify their appearance by either using a different chapterstyle for them than for the actual chapters, or by changing some of the macros. As examples:

- Doing

  ```
  \renewcommand{\printXtitle}[1]{\hfill\Large\itshape #1}
  ```

  will print the title right justified in a Large italic font.
- For a Large bold centered title you can do

  ```
  \renewcommand{\printXtitle}[1]{\centering\Large\bfseries #1}
  ```

- Writing

  ```
  \renewcommand{\afterXtitle}{%
                  \thispagestyle{empty}\afterchaptertitle}
  ```

  will result in the first page of the listing using the *empty* pagestyle instead of the default *chapter* pagestyle.
- Doing

  ```
  \renewcommand{\afterXtitle}{%
    \par\nobreak \mbox{}\hfill{\normalfont Page}\par\nobreak}
  ```

  will put the word 'Page' flushright on the line following the title.

9.2.2 Typesetting the entries

Commands are also provided to enable finer control over the typesetting of the different kinds of entries. The parameters defining the default layout of the entries are illustrated as part of the layouts package [Wil04c] or in [MG+04, p. 51], and are repeated in Figure 9.2.

Most of the commands in this section start as `\cft...`, where `cft` is intended as a mnemonic for *Table of **C**ontents*, *List of **F**igures*, *List of **T**ables*.

`\cftdot`

In the default ToC typesetting only the more minor entries have dotted leader lines between the sectioning title and the page number. The class provides for general leaders for all entries. The 'dot' in a leader is given by the value of `\cftdot`. Its default definition is `\newcommand{\cftdot}{.}` which gives the default dotted leader. By changing `\cftdot` you can use symbols other than a period in the leader. For example

```
\renewcommand{\cftdot}{\ensuremath{\ast}}
```

will result in a dotted leader using asterisks as the symbol.

`\cftdotsep`
`\cftnodots`

Each kind of entry can control the separation between the dots in its leader (see below). For consistency though, all dotted leaders should use the same spacing. The macro `\cftdotsep` specifies the default spacing. However, if the separation is too large then no dots will be actually typeset. The macro `\cftnodots` is a separation value that is 'too large'.

`\setpnumwidth{`⟨*length*⟩`}`
`\setrmarg{`⟨*length*⟩`}`

The page numbers are typeset in a fixed width box. The command `\setpnumwidth` can be used to change the width of the box (LaTeX 's internal `\@pnumwidth`). The title texts will end before reaching the righthand margin. `\setrmarg` can be used to set this distance (LaTeX 's internal `\@tocrmarg`). Note that the length used in `\setrmarg` should be greater than the length set in `\setpnumwidth`. These values should remain constant in any given document.

This manual requires more space for the page numbers than the default, so the following was set in the preamble:

```
\setpnumwidth{2.55em}
\setrmarg{3.55em}
```

`\cftparskip`

Normally the `\parskip` in the ToC, etc., is zero. This may be changed by changing the length `\cftparskip`. Note that the current value of `\cftparskip` is used for the ToC, LoF and LoT, but you can change the value before calling `\tableofcontents` or `\listoffigures` or `\listoftables` if one or other of these should have different values (which is not a good idea).

Again for convenience, in the following I will use `K` to stand for the *kind* of entry, as listed in Table 9.3; that is, any of the following:

- `book` for `\book` titles.

Table 9.3: Value of K in macros for styling entries in a 'List of...'

K	Kind of entry	K	Kind of entry
book	\book title	subparagraph	\subparagraph title
part	\part title	figure	figure caption
chapter	\chapter title	subfigure	subfigure caption
section	\section title	table	table caption
subsection	\subsection title	subtable	subtable caption
subsubsection	\subsubsection title	...	...

- part for \part titles
- chapter for \chapter titles
- section for \section titles
- subsection for \subsection titles
- subsubsection for \subsubsection titles
- paragraph for \paragraph titles
- subparagraph for \subparagraph titles
- figure for figure \caption titles
- subfigure for subfigure \caption titles
- table for table \caption titles
- subtable for subtable \caption titles

```
\cftbookbreak
\cftpartbreak
\cftchapterbreak
```

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

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

which encourages a page break before rather than after the entry. As usual, you can change \cftbookbreak to do other things that you feel might be useful. The macros \cftpartbreak and \cftchapterbreak apply to \part and \chapter entries, respectively, and have the same default definitions as \cftbookbreak.

```
\cftbeforeKskip
```

This length controls the vertical space before an entry. It can be changed by using \setlength.

```
\cftKindent
```

This length controls the indentation of an entry from the left margin (*indent* in Figure 9.2). It can be changed using \setlength.

```
\cftKnumwidth
```

This length controls the space allowed for typesetting title numbers (*numwidth* in Figure 9.2). It can be changed using `\setlength`. Second and subsequent lines of a multiline title will be indented by this amount.

The remaining commands are related to the specifics of typesetting an entry. This is a simplified pseudo-code version for the typesetting of numbered and unnumbered entries.

```
{\cftKfont {{\cftKname \cftKpresnum SNUM\cftKaftersnum\hfil} \cftKaftersnumb TITLE}}
        {\cftKleader}{\cftKformatpnum{PAGE}}\cftKafterpnum\par

{\cftKfont TITLE}{\cftKleader}{\cftKformatpnum{PAGE}}\cftKafterpnum\par
```

where `SNUM` is the section number, `TITLE` is the title text and `PAGE` is the page number. In the numbered entry the pseudo-code

```
{\cftKpresnum SNUM\cftKaftersnum\hfil}
```

is typeset within a box of width `\cftKnumwidth`, see the `\...numberlinebox` macros later on.

`\cftKfont`

This controls the appearance of the title (and its preceding number, if any). It may be changed using `\renewcommand`.

`\cftKfont` takes no arguments as such, but the the number and title is presented to it as an argument. Thus one may end `\cftKfont` with a macro taking one argument, say `\MakeUppercase`, and which then readjust the text as needed.

> ***Caveat.*** Please read the section entitled *About upper or lower casing TOC entries* on page 162 if you consider using upper/lower cased TOC entries and especially if you are also using the hyperref package.

`\cftKname`

The first element typeset in an entry is `\cftKname`.[3] Its default definition is

```
\newcommand*{\cftKname}{}
```

so it does nothing. However, to put the word 'Chapter' before each chapter number in a ToC and 'Fig.' before each figure number in a LoF do:

```
\renewcommand*{\cftchaptername}{Chapter\space}
\renewcommand*{\cftfigurename}{Fig.\space}
```

`\cftKpresnum` `\cftKaftersnum` `\cftKaftersnumb`

The section number is typeset within a box of width `\cftKnumwidth`. Within the box the macro `\cftKpresnum` is first called, then the number is typeset, and the `\cftKaftersnum` macro is called after the number is typeset. The last command within the box is `\hfil` to make the box contents flushleft. After the box is typeset the `\cftKaftersnumb` macro is called before typesetting the title text. All three of these can be changed by `\renewcommand`. By default they are defined to do nothing.

[3] Suggested by Danie Els.

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

In the ToC, the macros `\booknumberline`, `\partnumberline` and `\chapternumberline` are responsible respectively for typesetting the `\book`, `\part` and `\chapter` numbers, whereas `\numberline` does the same for the sectional siblings. Internally they use `\cftKpresnum`, `\cftKaftersnum` and `\cftKaftersnumb` as above. If you do not want, say, the `\chapter` number to appear you can do:

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

```
\numberlinehook{⟨num⟩}
\cftwhatismyname
\booknumberlinehook{⟨num⟩}
\partnumberlinehook{⟨num⟩}
\chapternumberlinehook{⟨num⟩}
\numberlinebox{⟨length⟩}{⟨code⟩}
\booknumberlinebox{⟨length⟩}{⟨code⟩}
\partnumberlinebox{⟨length⟩}{⟨code⟩}
\chapternumberlinebox{⟨length⟩}{⟨code⟩}
```

Inside the four `\...numberline` macros, the first thing we do is to give the `\...numberline` argument to a hook. By default this hook does nothing. But, with the right tools,[4] they can be used to record the widths of the sectional number. Which then can be used to automatically adjust the various ⟨*numwidth*⟩ and ⟨*indent*⟩ within the `\cftsetindents` macro. In order to tell the section types apart (they all use `\numberline`), the value of the `\cftwhatismyname` macro will locally reflect the current type.

As mentioned earlier, the `\book`, `\part` and `\chapter` numbers are typeset inside a box of certain fixed widths. Sometimes it can be handy *not* having this box around. For this you can redefine one of the four `\...numberlinebox` macros listed above. For example via

```
\renewcommand\chapternumberlinebox[2]{#2}
```

The first argument is the width of the box to be made. All four macros are defined similar to this (where `#1` is a length)

```
\newcommand\chapternumberlinebox[2]{%
  \hb@xt@#1{#2\hfil}}
```

```
\cftKleader
\cftKdotsep
```

`\cftKleader` defines the leader between the title and the page number; it can be changed by `\renewcommand`. The spacing between any dots in the leader is controlled by `\cftKdotsep` (`\@dotsep` in Figure 9.2). It can be changed by `\renewcommand` and its value must be either a number (e.g., 6.6 or `\cftdotsep`) or `\cftnodots` (to disable the dots). The spacing is in terms of *math units* where there are 18mu to 1em.

[4] Which we do not currently supply…, but have a look at Sniplet C.6 on page 409.

The default leaders macro is similar to

```
\newcommand{\cftsectionleader}{\normalfont\cftdotfill{\cftsectiondotsep}}
```

Note that the spacing of the dots is affected by the font size (as the math unit is affected by the font size). Also note that the `\cftchapterleader` is bold by default.

`\cftKformatpnum{`⟨*pnum*⟩`}`
`\cftKformatpnumhook{`⟨*num*⟩`}`
`\cftKpagefont`

The macro `\cftKformatpnum` typesets an entry's page number, using the `\cftKpagefont`.[5] The default definition is essentially:

```
\newcommand*{\cftKformatpnum}[1]{%
  \cftKformatpnumhook{#1}%
  \hbox to \@pnumwidth{\hfil{\cftKpagefont #1}}}
```

which sets the number right justified in a box `\@pnumwidth` wide. To have, say, a `\part` page number left justified in its box, do:

```
\renewcommand*{\cftpartformatpnum}[1]{%
  \cftpartformatpnumhook{#1}%
  \hbox to \@pnumwidth{{\cftpartpagefont #1}}}
```

The `\cftKformatpnumhook` does nothing by default (other than eating the argument), but could be redefined to record the widest page number and report it back, even reusing it to auto adjust on the next run to set `\@pnumwidth` (see `\setpnumwidth`).

`\cftKafterpnum`

This macro is called after the page number has been typeset. Its default is to do nothing. It can be changed by `\renewcommand`.

`\cftsetindents{`⟨*kind*⟩`}{`⟨*indent*⟩`}{`⟨*numwidth*⟩`}`

The command `\cftsetindents` sets the ⟨*kind*⟩ entries *indent* to the length ⟨*indent*⟩ and its *numwidth* to the length ⟨*numwidth*⟩. The ⟨*kind*⟩ argument is the name of one of the standard entries (e.g., `subsection`) or the name of entry that has been defined within the document. For example

```
\cftsetindents{figure}{0em}{1.5em}
```

will make figure entries left justified.

This manual requires more space for section numbers in the ToC than the default (which allows for three digits). Consequently the preamble contains the following:

```
\cftsetindents{section}{1.5em}{3.0em}
\cftsetindents{subsection}{4.5em}{3.9em}
\cftsetindents{subsubsection}{8.4em}{4.8em}
\cftsetindents{paragraph}{10.7em}{5.7em}
\cftsetindents{subparagraph}{12.7em}{6.7em}
```

[5] This addition to the class was suggested by Dan Luecking, CTT *Re: setting numbers in toc in their natural width box*, 2007/08/15.

Note that changing the indents at one level implies that any lower level indents should be changed as well.

Various effects can be achieved by changing the definitions of `\cftKfont`, `\cftKaftersnum`, `\cftKaftersnumb`, `\cftKleader` and `\cftKafterpnum`, either singly or in combination. For the sake of some examples, assume that we have the following initial definitions

```
\newcommand*{\cftKfont}{}
\newcommand*{\cftKaftersnum}{}
\newcommand*{\cftKaftersnumb}{}
\newcommand*{\cftKleader}{\cftdotfill{\cftKdotsep}}
\newcommand*{\cftKdotsep}{\cftdotsep}
\newcommand*{\cftKpagefont}{}
\newcommand*{\cftKafterpnum}{}
```

Note that the same font should be used for the title, leader and page number to provide a coherent appearance.

- To eliminate the dots in the leader:

  ```
  \renewcommand*{\cftKdotsep}{\cftnodots}
  ```

- To put something (e.g., a name) before the title (number):

  ```
  \renewcommand*{\cftKname}{SOMETHING }
  ```

- To add a colon after the section number:

  ```
  \renewcommand*{\cftKaftersnum}{:}
  ```

- To put something before the title number, add a double colon after the title number, set everything in bold font, and start the title text on the following line:

  ```
  \renewcommand*{\cftKfont}{\bfseries}
  \renewcommand*{\cftKleader}{\bfseries\cftdotfill{\cftKdotsep}}
  \renewcommand*{\cftKpagefont}{\bfseries}
  \renewcommand*{\cftKname}{SOMETHING }
  \renewcommand{\cftKaftersnum}{::}
  \renewcommand{\cftKaftersnumb}{\\}
  ```

  If you are adding text in the number box in addition to the number, then you will probably have to increase the width of the box so that multiline titles have a neat vertical alignment; changing box widths usually implies that the indents will require modification as well. One possible method of adjusting the box width for the above example is:

  ```
  \newlength{\mylen}                  % a "scratch" length
  \settowidth{\mylen}{\bfseries\cftKaftersnum}
  \addtolength{\cftKnumwidth}{\mylen} % add the extra space
  ```

- To set the chapter number and title as just 'NUM · TITLE', i.e. un-boxed number plus a symbolic separator, use

  ```
  \renewcommand\cftchapteraftersnumb{\enspace\textperiodcentered\enspace}
  \renewcommand\chapternumberlinebox[2]{#2}
  ```

– of couse, it works best, only if the TITLE is a single line.

- Make chapter titles lower case small caps

```
\renewcommand\cftchapterfont{\scshape\MakeTextLowercase}
```

– here we do not touch the case of any math.

- To set the section numbers flushright:

```
\setlength{\mylen}{0.5em}    % extra space at end of number
\renewcommand{\cftKpresnum}{\hfill} % note the double 'l'
\renewcommand{\cftKaftersnum}{\hspace*{\mylen}}
\addtolength{\cftKnumwidth}{\mylen}
```

In the above, the added initial `\hfill` in the box overrides the final `\hfil` in the box, thus shifting everything to the right hand end of the box. The extra space is so that the number is not typeset immediately at the left of the title text.

- To set the entry ragged left (but this only looks good for single line titles):

```
\renewcommand{\cftKfont}{\hfill\bfseries}
\renewcommand{\cftKleader}{}
```

- To set the titles ragged right instead of the usual flushright. Assuming that there are more than 100 pages in the document (otherwise adjust the length):

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

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

- To set the page number immediately after the entry text instead of at the righthand margin:

```
\renewcommand{\cftKleader}{}
\renewcommand{\cftKafterpnum}{\cftparfillskip}
```

`\cftparfillskip`

By default the `\parfillskip` value is locally set to fill up the last line of a paragraph. Just changing `\cftKleader` as in the above example puts horrible interword spaces into the last line of the title. The `\cftparfillskip` command is provided just so that the above effect can be achieved.

`\cftpagenumbersoff{`⟨*kind*⟩`}`
`\cftpagenumberson{`⟨*kind*⟩`}`

The command `\cftpagenumbersoff` will eliminate the page numbers for ⟨*kind*⟩ entries in the listing, where ⟨*kind*⟩ is the name of one of the standard kinds of entries (e.g., `subsection`, or `figure`) or the name of a new entry defined in the document.

The command `\cftpagenumberson` reverses the effect of a corresponding `\cftpagenumbersoff` for ⟨*kind*⟩.

For example, to eliminate page numbers for appendices in the ToC:

```
...
\appendix
\addtocontents{toc}{\cftpagenumbersoff{chapter}}
\chapter{First appendix}
```

If there are other chapter type headings to go into the ToC after the appendices (perhaps a bibliography or an index), then it will be necessary to do a similar

```
\addtocontents{toc}{\cftpagenumberson{chapter}}
```

after the appendices to restore the page numbering in the ToC.

Sometimes it may be desirable to make a change to the global parameters for an individual entry. For example, a figure might be placed on the end paper of a book (the inside of the front or back cover), and this needs to be placed in a LoF with the page number set as, say, 'inside front cover'. If 'inside front cover' is typeset as an ordinary page number it will stick out into the margin. Therefore, the parameters for this particular entry need to be changed.

`\cftlocalchange{`⟨*ext*⟩`}{`⟨*pnumwidth*⟩`}{`⟨*tocrmarg*⟩`}`

The command `\cftlocalchange` will write an entry into the file with extension ⟨*ext*⟩ to reset the global `\@pnumwidth` and `\@tocrmarg` parameter lengths. The command should be called again after any special entry to reset the parameters back to their usual values. Any fragile commands used in the arguments must be protected.

`\cftaddtitleline{`⟨*ext*⟩`}{`⟨*kind*⟩`}{`⟨*title*⟩`}{`⟨*page*⟩`}`
`\cftaddnumtitleline{`⟨*ext*⟩`}{`⟨*kind*⟩`}{`⟨*num*⟩`}{`⟨*title*⟩`}{`⟨*page*⟩`}`

The command `\cftaddtitleline` will write a `\contentsline` entry into ⟨*ext*⟩ for a ⟨*kind*⟩ entry with title ⟨*title*⟩ and page number ⟨*page*⟩. Any fragile commands used in the arguments must be protected. That is, an entry is made of the form:

```
\contentsline{kind}{title}{page}
```

The command `\cftaddnumtitleline` is similar to `\cftaddtitleline` except that it also includes ⟨*num*⟩ as the argument to `\numberline`. That is, an entry is made of the form

```
\contentsline{kind}{\numberline{num} title}{page}
```

As an example of the use of these commands, noting that the default LaTeX values for `\@pnumwidth` and `\@tocrmarg` are 1.55em and 2.55em respectively, one might do the following for a figure on the frontispiece page.

```
...
 this is the frontispiece page with no number
 draw or import the picture (with no \caption)
\cftlocalchange{lof}{4em}{5em} % make pnumwidth big enough for
                               % frontispiece and change margin
\cftaddtitleline{lof}{figure}{The title}{frontispiece}
\cftlocalchange{lof}{1.55em}{2.55em} % return to normal settings
\clearpage
...
```

Recall that a `\caption` command will put an entry in the `lof` file, which is not wanted here. If a caption is required, then you can either craft one youself or, assuming that your general captions are not too exotic, use the `\legend` command (see later). If the illustration is numbered, use `\cftaddnumtitleline` instead of `\cftaddtitleline`.

Inserting stuff into the content lists

The next functions were suggested by Lars Madsen who found them useful if, for example, you had two versions of the ToC and you needed some aspects to be formatted differently.

`\cftinsertcode{`⟨*name*⟩`}{`⟨*code*⟩`}`
`\cftinserthook{`⟨*file*⟩`}{`⟨*name*⟩`}`

The `\cftinserthook` is somewhat like `\addtocontents` in that it enables you to insert a code hook into the ToC, etc., where ⟨*file*⟩ is the (`toc`, `lof`, ...) file and ⟨*name*⟩ is the 'name' of the hook. The ⟨*code*⟩ for the hook is specified via `\cftinsertcode` where ⟨*name*⟩ is the name you give to the hook. These can be used to make alterations to a 'List of...' on the fly. For example:

```
\cftinsertcode{A}{%
  \renewcommand*{\cftchapterfont}{\normalfont\scshape}
  ... }% code for ToC
...
\frontmatter
\tableofcontents
\cftinsertcode{G}{...}% code for LoF
\cftinsertcode{F}{...}% code for LoF
\listoffigures
...
\cftinserthook{lof}{G}
...
\chapter{...}
...
\mainmatter
\cftinserthook{toc}{A}
\cftinserthook{lof}{F}
\chapter{...}
...
```

If you do not use `\cftinsertcode` *before* calling the command to type the 'List of...' that it is intended for then nothing will happen. No harm will come if a matching `\cftinserthook` is never used. No harm occurs either if you call `\cftinserthook` and there is no prior matching `\cftinsertcode`.

One use of these ToC hooks is reusing the ToC data to, say, create chapter ToC's. The code for this is shown in Sniplet C.7 on page 411. In the sniplet we use the following two hooks that are executed right before and right after `\chapter`, `\part`, `\book`, `\appendixpage` writes to the ToC. By default they do nothing.[6]

[6] More hooks may be added in later releases.

```
\mempreaddchaptertotochook
\mempostaddchaptertotochook
\mempreaddparttotochook
\mempostaddparttotochook
\mempreaddbooktotochook
\mempostaddbooktotochook
\mempreaddapppagetotochook
\mempostaddapppagetotochook
```

Extra chapter material in the ToC

```
\precistoctext{⟨text⟩} \precistocfont \precistocformat
```

The `\chapterprecistoc` macro puts `\precistoctext{`⟨*text*⟩`}` into the `toc` file. Further information as to the definition of this macro can be found in section 6.5.3.

About upper or lower casing TOC entries

Some designs call for upper (or lower casing) TOC entries. This *is* possible but the solution depends on whether the hyperref package is used or not.

Without hyperref one can simply end the `\cftKfont` with say `\MakeTextUppercase` and the K-type entry will be upper cased.

With hyperref the possibilities are limited. Explanation: The upper/lower casing macros are not that robust, and need the content to be simple.[7] When hyperref is used, the hyperlink is wrapped around the entry before `\cftKfont` gains access to it, and is thus generally too complicated for, say, `\MakeTextUppercase` to handle. The follow workaround draw inspiration from http://tex.stackexchange.com/q/11892/3929.

```
\settocpreprocessor{⟨type⟩}{⟨code⟩}
```

Here ⟨*type*⟩ is one of `chapter`, `part` or `book`.[8] And ⟨*code*⟩ can be something like this example:

```
\makeatletter
\settocpreprocessor{chapter}{%
  \let\tempf@rtoc\f@rtoc%
  \def\f@rtoc{%
    \texorpdfstring{\MakeTextUppercase{\tempf@rtoc}}{\tempf@rtoc}}%
}
\makeatother
```

Where `\f@rtoc` is a placeholder inside `\chapter`, `\part` and `\book`, holding the material to be written to the actual TOC file before hyperref accesses it. This way the upper casing is sneaked into the TOC file, and the bookmark part of hyperref will not complain about the `\MakeTextUppercase` in the data. Of course, you will not have upper cased titles in the bookmark list.

[7] For some definition of simple.

[8] If needed we will attempt to add a similar feature to the rest of the sectional types, please write the maintainer.

9.2.3 Example: No section number

There are at least two ways of listing section titles in the ToC without displaying their numbers and both involve the `\numberline` command which typesets the number in a box.

The first method redefines `\numberline` so it throws away the argument. We do this by modifying the `\cftKfont` macro which is called before `\numberline` and the `\cftKafterpnum` which is called after the page number has been typeset.

```
\let\oldcftsf\cftsectionfont%  save definition of \cftsectionfont
\let\oldcftspn\cftsectionafterpnum% and of \cftsectionafterpnum
\renewcommand*{\cftsectionfont}{%
  \let\oldnl\numberline%       save definition of \numberline
  \renewcommand*{\numberline}[1]{}% change it
  \oldcftsf}                   % use original \cftsectionfont
\renewcommand*{\cftsectionafterpnum}{%
  \let\numberline\oldnl%       % restore orginal \numberline
  \oldcftspn}                  % use original \cftsectionafterpnum
```

Probing a little deeper, the `\numberline` macro is called to typeset section numbers and is defined as:

```
\renewcommand*{\numberline}[1]{%
  \hb@xt@\@tempdima{\@cftbsnum #1\@cftasnum\hfil}\@cftasnumb}
```

Each kind of heading `\lets` the `\@cftbsnum` macro to `\cftKpresnum`, and the `\@cftasnum` macro to `\cftKaftersnum`, and the `\@cftasnumb` macro to `\cftKaftersnumb` as appropriate for the heading. The second method for killing the number uses a TeX method for defining a macro with a delimited argument.

```
\def\cftsectionpresnum #1\@cftasnum{}
```

The interpretation of this is left as an exercise for anyone who might be interested.

9.2.4 Example: Multicolumn entries

If the subsection entries, say, in the ToC are going to be very short it might be worth setting them in multiple columns. Here is one way of doing that which depends on using the multicol package [Mit18]. This assumes that subsections will be the lowest heading in the ToC.

```
\newcounter{toccols}
\setcounter{toccols}{3}
\newenvironment{mysection}[1]{%
  \section{#1}%
  \addtocontents{toc}{\protect\begin{multicols}{\value{toccols}}}}%
  {\addtocontents{toc}{\protect\end{multocols}}}
```

The counter `toccols` controls the number of columns to be used. For each section where you want subsections to be typeset in multiple columns in the ToC, use the `mysection` environment instead of `\section`, like:

```
\begin{mysection}{Columns}
...
```

```
\subsection{Fat}
...
\subsection{Thin}
...
\end{mysection}
```

Any ToC entries generated from within the environment will be enclosed in a `multicols` environment in the ToC. The `\protect`s have to be used because environment `\begin` and `\end` commands are fragile.

9.2.5 Example: Multiple contents

It is easy to have two ToCs, one short and one long, when they are of the same style, like this:

```
...
\renewcommand*{\contentsname}{Short contents}
\setcounter{tocdepth}{0}%  chapters and above
\tableofcontents
% \clearpage
\renewcommand*{\contentsname}{Contents}
\setcounter{tocdepth}{2}%  subsections and above
\tableofcontents
```

(Note that you can't use `\settocdepth` in this case as that writes the change into the ToC, so that the second use would override the first.)

This book has both a short and a long ToC, neither of which look like those typically associated with LaTeX. This is how they were done.

The general style for the ToC, etc., is specified in the memsty package file.

```
%%% need more space for ToC page numbers
\setpnumwidth{2.55em}
\setrmarg{3.55em}
%%% need more space for ToC section numbers
\cftsetindents{section}{1.5em}{3.0em}
\cftsetindents{subsection}{4.5em}{3.9em}
\cftsetindents{subsubsection}{8.4em}{4.8em}
\cftsetindents{paragraph}{10.7em}{5.7em}
\cftsetindents{subparagraph}{12.7em}{6.7em}
%%% need more space for LoF & LoT numbers
\cftsetindents{figure}{0em}{3.0em}
\cftsetindents{table}{0em}{3.0em}
%%% remove the dotted leaders
\renewcommand{\cftsectiondotsep}{\cftnodots}
\renewcommand{\cftsubsectiondotsep}{\cftnodots}
\renewcommand{\cftsubsubsectiondotsep}{\cftnodots}
\renewcommand{\cftparagraphdotsep}{\cftnodots}
\renewcommand{\cftsubparagraphdotsep}{\cftnodots}
\renewcommand{\cftfiguredotsep}{\cftnodots}
\renewcommand{\cfttabledotsep}{\cftnodots}
```

Three macros are defined to control the appearance of the short and the long ToC. First, the macro `\setupshorttoc` for the short version. The first few lines ensure that only chapter or part titles will be set, and any chapter precis text or `tocdepth` changes will be ignored. The rest of the code specifies how the chapter titles are to be typeset, and finally the part and book titles.

```
\newcommand*{\setupshorttoc}{%
  \renewcommand*{\contentsname}{Short contents}
  \let\oldchangetocdepth\changetocdepth
  \renewcommand*{\changetocdepth}[1]{}
  \let\oldprecistoctext\precistoctext
  \renewcommand{\precistoctext}[1]{}
  \let\oldcftchapterfillnum\cftchapterfillnum
  \setcounter{tocdepth}{0}% chapters and above
  \renewcommand*{\cftchapterfont}{\hfill\sffamily}
  \renewcommand*{\cftchapterleader}{ \textperiodcentered\space}
  \renewcommand*{\cftchapterafterpnum}{\cftparfillskip}
%%  \setpnumwidth{0em}
%%  \setpnumwidth{1.5em}
  \renewcommand*{\cftchapterfillnum}[1]{%
    {\cftchapterleader}\nobreak
    \hbox to 1.5em{\cftchapterpagefont ##1\hfil}\cftchapterafterpnum\par}
  \setrmarg{0.3\textwidth}
  \setlength{\unitlength}{\@tocrmarg}
  \addtolength{\unitlength}{1.5em}
  \let\oldcftpartformatpnum\cftpartformatpnum
  \renewcommand*{\cftpartformatpnum}[1]{%
    \hbox to\unitlength{{\cftpartpagefont ##1}}}
  \let\oldcftbookformatpnum\cftbookformatpnum
  \renewcommand*{\cftbookformatpnum}[1]{%
    \hbox to\unitlength{{\cftbookpagefont ##1}}}
```

You can do many things using the `\cft...` macros to change the appearance of a ToC but they can't be entirely coerced into specifying the paragraphing of the `\subsection` titles. The `\setupparasubsecs` also went in the preamble.

```
\newcommand*{\setupparasubsecs}{%
  \let\oldnumberline\numberline
  \renewcommand*{\cftsubsectionfont}{\itshape}
  \renewcommand*{\cftsubsectionpagefont}{\itshape}
  \renewcommand{\l@subsection}[2]{%
    \def\numberline####1{\textit{####1}~}%
    \leftskip=\cftsubsectionindent
    \rightskip=\@tocrmarg
%% \advance\rightskip 0pt plus \hsize % uncomment this for raggedright
%% \advance\rightskip 0pt plus 2em    % uncomment this for semi-raggedright
    \parfillskip=\fill
    \ifhmode ,\ \else\noindent\fi
    \ignorespaces
```

```
    {\cftsubsectionfont ##1}~{\cftsubsectionpagefont##2}%
    \let\numberline\oldnumberline\ignorespaces}
}
\AtEndDocument{\addtocontents{toc}{\par}
```

The above code changes the appearance of subsection titles in the ToC, setting each group as a single paragraph (each is normally set with a paragraph to itself). By uncommenting or commenting the noted lines in the code you can change the layout a little.

> ***Caveat.*** We have an interesting caveat regarding `\setupparasubsecs` if you are using hyperref *and* you have subsubsections, that are not shown in the ToC. You may see some inline subsection entries showing up as '… text 15 ,…', that is a strange space appears before the comma.
>
> This is an artifact due to the way hyperref wraps itself around the ToC entries, even the ones that are not typeset, and thus an end of line space survives. We fix it using `\endlinechar`:
>
> ```
> \begingroup
> \endlinechar=-1
> \tableofcontents
> \endgroup
> ```
>
> Note again that it only happen if you have subsubsections with an inline subsection entry list, *and* you are using hyperref.

Normally, section titles (and below) are set as individual paragraphs. Effectively the first thing that is done is to end any previous paragraph, and also the last thing is to end the current paragraph. Notice that the main code above neither starts nor finishes a paragraph. If the group of subsections is followed by a section title, that supplies the paragraph end. The last line above ensures that the last entry in the `toc` file is `\par` as this might be needed to finish off a group of subsections if these are the last entries.

And thirdly for the main ToC, the macro `\setupmaintoc` reverts everything back to normal.

```
\newcommand*{\setupmaintoc}{%
  \renewcommand{\contentsname}{Contents}
  \let\changetocdepth\oldchangetocdepth
  \let\precistoctext\oldprecistoctext
  \let\cftchapterfillnum\oldcftchapterfillnum
  \addtodef{\cftchapterbreak}{\par}{}
  \renewcommand*{\cftchapterfont}{\normalfont\sffamily}
  \renewcommand*{\cftchapterleader}{%
                \sffamily\cftdotfill{\cftchapterdotsep}}
  \renewcommand*{\cftchapterafterpnum}{}
  \renewcommand{\cftchapterbreak}{\par\addpenalty{-\@highpenalty}}
  \setpnumwidth{2.55em}
  \setrmarg{3.55em}
  \setcounter{tocdepth}{2}}
  \let\cftpartformatpnum\oldcftpartformatpnum
    \addtodef{\cftpartbreak}{\par}{}
```

```
\let\cftbookformatpnum\oldcftbookformatpnum
  \addtodef{\cftbookbreak}{\par}{}
```

The first few lines restore some macros to their original definitions.

```
\addtodef{\cftchapterbreak}{\par}{}
```

ensures that a chapter entry starts off with a `\par`; this is needed when the previous entry is a group of subsections and their paragraph has to be ended. The remaining code lines simply set the appearance of the chapter titles and restore that for parts and books, as well as ensuring that they start off new paragraphs.

In the document itself, `\tableofcontents` was called twice, after the appropriate setups:

```
...
\setupshorttoc
\tableofcontents
\clearpage
\setupparasubsecs
\setupmaintoc
\tableofcontents
\setlength{\unitlength}{1pt}
...
```

After all this note that I ensured that `\unitlength` was set to its default value (it had been used as a scratch length in the code for `\setupparasubsecs`).

9.3 New 'List of...' and entries

`\newlistof{`⟨*listofcom*⟩`}{`⟨*ext*⟩`}{`⟨*listofname*⟩`}`

The command `\newlistof` creates a new 'List of...', and assorted commands to go along with it. The first argument, ⟨*listofcom*⟩ is used to define a new command called `\listofcom` which can then be used like `\listoffigures` to typeset the 'List of...'. The ⟨*ext*⟩ argument is the file extension to be used for the new listing. The last argument, ⟨*listofname*⟩ is the title for the 'List of...'. Unstarred and starred versions of `\listofcom` are created. The unstarred version, `\listofcom`, will add ⟨*listofname*⟩ to the ToC, while the starred version, `\listofcom*`, makes no entry in the ToC.

As an example:

```
\newcommand{\listanswername}{List of Answers}
\newlistof{listofanswers}{ans}{\listanswername}
```

will create a new `\listofanswers` command that can be used to typeset a listing of answers under the title `\listanswername`, where the answer titles are in an `ans` file. It is up to the author of the document to specify the 'answer' code for the answers in the document. For example:

```
\newcounter{answer}[chapter]
\renewcommand{\theanswer}{\arabic{answer}}
\newcommand{\answer}[1]{
  \refstepcounter{answer}
  \par\noindent\textbf{Answer \theanswer. #1}
```

```
\addcontentsline{ans}{answer}{\protect\numberline{\theanswer}#1}\par}
```

which, when used like:

```
\answer{Hard} The \ldots
```

will print as:

Answer 1. Hard
The ...

As mentioned above, the `\newlistof` command creates several new commands in addition to `\listofcom`, most of which you should now be familiar with. For convenience, assume that `\newlistof{...}{X}{...}` has been issued so that `X` is the new file extension and corresponds to the `X` in §9.2.1. Then in addition to `\listofcom` the following new commands will be made available.

The four commands, `\Xmark`, `\Xheadstart`, `\printXtitle`, and `\afterXtitle`, are analagous to the commands of the same names described in §9.2.1 (internally the class uses the `\newlistof` macro to define the ToC, LoF and LoT). In particular the default definition of `\Xmark` is equivalent to:

```
\newcommand{\Xmark}{\markboth{listofname}{listofname}}
```

However, this may well be altered by the particular `\pagestyle` in use.

`Xdepth`

The counter `Xdepth` is analagous to the standard `tocdepth` counter, in that it specifies that entries in the new listing should not be typeset if their numbering level is greater than `Xdepth`. The default definition is equivalent to

```
\setcounter{Xdepth}{1}
```

`\insertchapterspace`
`\addtodef{`⟨*macro*⟩`}{`⟨*prepend*⟩`}{`⟨*append*⟩`}`

Remember that the `\chapter` command uses `\insertchapterspace` to insert vertical spaces into the LoF and LoT. If you want similar spaces added to your new listing then you have to modify `\insertchapterspace`. The easiest way to do this is via the `\addtodef` macro, like:

```
\addtodef{\insertchapterspace}{}%
  {\addtocontents{ans}{\protect\addvspace{10pt}}}
```

The `\addtodef` macro is described later in §18.10.

The other part of creating a new 'List of...', is to specify the formatting of the entries, i.e., define an appropriate `\l@kind` macro.

`\newlistentry[`⟨*within*⟩`]{`⟨*cntr*⟩`}{`⟨*ext*⟩`}{`⟨*level-1*⟩`}`

The command `\newlistentry` creates the commands necessary for typesetting an entry in a 'List of...'. The first required argument, ⟨*cntr*⟩ is used to define a new counter called `cntr`, unless `cntr` is already defined. The optional ⟨*within*⟩ argument can be used so that `cntr` gets reset to one every time the counter called `within` is changed. That is, the first two arguments when `cntr` is not already defined, are equivalent to calling `\newcounter{`⟨*cntr*⟩`}[`⟨*within*⟩`]`. If `cntr` is already defined, `\newcounter` is not called. `cntr` is used for the number that goes along with the title of the entry.

The second required argument, ⟨*ext*⟩, is the file extension for the entry listing. The last argument, ⟨*level-1*⟩, is a number specifying the numbering level minus one, of the entry in a listing.

Calling `\newlistentry` creates several new commands used to configure the entry. So in order to configure the list look of our previous answer example we would add

```
\newlistentry{answer}{ans}{0}
```

Assuming that `\newlistentry` is called as `\newlistentry[within]{K}{X}{N}`, where `K` and `X` are similar to the previous uses of them (e.g., `K` is the kind of entry `X` is the file extension), and `N` is an integer number, then the following commands are made available.

The set of commands `\cftbeforeKskip`, `\cftKfont`, `\cftKpresnum`, `\cftKaftersnum`, `\cftKaftersnumb`, `\cftKleader`, `\cftKdotsep`, `\cftKpagefont`, and `\cftKafterpnum`, are analagous to the commands of the same names described in §9.2.2. Their default values are also as described earlier.

The default values of `\cftKindent` and `\cftKnumwidth` are set according to the value of the ⟨*level-1*⟩ argument (i.e., `N` in this example). For `N=0` the settings correspond to those for figures and tables, as listed in Table 9.1 for the memoir class. For `N=1` the settings correspond to subfigures, and so on. For values of `N` less than zero or greater than four, or for non-default values, use the `\cftsetindents` command to set the values.

`\l@K` is an internal command that typesets an entry in the list, and is defined in terms of the above `\cft*K*` commands. It will not typeset an entry if `Xdepth` is `N` or less, where `X` is the listing's file extension.

The command `\theK` prints the value of the `K` counter. It is initially defined so that it prints arabic numerals. If the optional ⟨*within*⟩ argument is used, `\theK` is defined as

```
\renewcommand{\theK}{\thewithin.\arabic{K}}
```

otherwise as

```
\renewcommand{\theK}{\arabic{K}}
```

As an example of the independent use of `\newlistentry`, the following will set up for sub-answers.

```
\newlistentry[answer]{subanswer}{ans}{1}
\renewcommand{\thesubanswer}{\theanswer.\alph{subanswer}}
\newcommand{\subanswer}[1]{
   \refstepcounter{subanswer}
   \par\textbf{\thesubanswer) #1}
   \addcontentsline{ans}{subanswer}{\protect\numberline{\thesubanswer}#1}
\setcounter{ansdepth}{2}
```

And then:

```
\answer{Harder} The \ldots
  \subanswer{Reformulate the problem} It assists \ldots
```

will be typeset as:

Answer 2. Harder
The ...
2.a) Reformulate the problem It assists ...

By default the answer entries will appear in the List of Answers listing (typeset by the `\listofanswers` command). In order to get the subanswers to appear, the `\setcounter{ansdepth}{2}` command was used above.

To turn off page numbering for the subanswers, do

```
\cftpagenumbersoff{subanswer}
```

As another example of `\newlistentry`, suppose that an extra sectioning division below `subparagraph` is required, called `subsubpara`. The `\subsubpara` command itself can be defined via the LaTeX kernel `\@startsection` command. Also it is necessary to define a `\subsubparamark` macro, a new `subsubpara` counter, a `\thesubsubpara` macro and a `\l@subsubpara` macro. Using `\newlistentry` takes care of most of these as shown below; remember the caveats about commands with `@` signs in them (see §E.4).

```
\newcommand{\subsubpara}{\@startsection{subpara}
   {6}                              %                    level
   {\parindent}                     %  indent from left margin
   {3.25ex \@plus1ex \@minus .2ex}  %     skip above heading
   {-1em}       run-in heading with % 1em between title & text
   {\normalfont\normalsize\itshape} % italic number and title
}
\newlistentry[subparagraph]{subsubpara}{toc}{5}
\cftsetindents{subsubpara}{14.0em}{7.0em}
\newcommand*{\subsubparamark}[1]{}  % gobble heading mark
```

Each 'List of…' uses a file to store the list entries, and these files must remain open for writing throughout the document processing. TeX has only a limited number of files that it can keep open, and this puts a limit on the number of listings that can be used. For a document that includes a ToC but no other extra ancilliary files (e.g., no index or bibliography output files) the maximum number of LoX's, including a LoF and LoT, is no more than about eleven. If you try and create too many new listings LaTeX will respond with the error message:

```
No room for a new write
```

If you get such a message the only recourse is to redesign your document.

9.3.1 Example: plates

As has been mentioned earlier, some illustrations may be tipped in to a book. Often, these are called *plates* if they are on glossy paper and the rest of the book is on ordinary paper. We can define a new kind of Listing for these.

```
\newcommand{\listplatename}{Plates}
\newlistof{listofplates}{lop}{\listplatename}
\newlistentry{plate}{lop}{0}
\cftpagenumbersoff{plate}
```

This code defines the `\listofplates` command to start the listing which will be titled 'Plates' from the `\listplatename` macro. The entry name is `plate` and the file extension is `lop`. As plate pages typically do not have printed folios, the `\cftpagenumbersoff` command has been used to prohibit page number printing in the listing.

If pages are tipped in, then they are put between a verso and a recto page. The afterpage package [Car95] lets you specify something that should happen after the current page is finished. The next piece of code uses the package and its `\afterpage` macro to define two macros which let you specify something that is to be done after the next verso (`\afternextverso`) or recto (`\afternextrecto`) page has been completed.

```
\newcommand{\afternextverso}[1]{%
  \afterpage{\ifodd\c@page #1\else\afterpage{#1}\fi}}
\newcommand{\afternextrecto}[1]{%
  \afterpage{\ifodd\c@page\afterpage{#1}\else #1\fi}}
```

The `\pageref{`⟨*labelid*⟩`}` command typesets the page number corresponding to the location in the document where `\label{`⟨*labelid*⟩`}` is specified. The following code defines two macros[9] that print the page number before (`\priorpageref`) or after (`\nextpageref`) that given by `\pageref`.

```
\newcounter{mempref}
\newcommand{\priorpageref}[1]{%
  \setcounter{mempref}{\pageref{#1}}\addtocounter{mempref}{-1}\thempref}
\newcommand{\nextpageref}[1]{%
  \setcounter{mempref}{\pageref{#1}}\addtocounter{mempref}{1}\thempref}
```

With these preliminaries out of the way, we can use code like the following for handling a set of physically tipped in plates.

```
\afternextverso{\label{tip}
  \addtocontents{lop}{%
    Between pages \priorpageref{tip} and \pageref{tip}
    \par\vspace*{\baselineskip}}
  \addcontentsline{lop}{plate}{First plate}
  \addcontentsline{lop}{plate}{Second plate}
  ...
  \addcontentsline{lop}{plate}{Nth plate}
}
```

This starts off by waiting until the next recto page is started, which will be the page immediately after the plates, and then inserts the label `tip`. The `\addtocontents` macro puts its argument into the plate list `lop` file, indicating the page numbers before and after the set of plates. With the plates being physically added to the document it is not possible to use `\caption`, instead the `\addcontentsline` macros are used to add the plate titles to the `lop` file.

With a few modifications the code above can also form the basis for listing plates that are electronically tipped in but do not have printed folios or `\captions`.

9.4 Chapter precis

See section 6.5.3 on page 92.

[9] These only work for arabic page numbers.

9.5 CONTENTS LISTS AND BOOKMARKS

With the hyperref package, the table of contents is often added as a list of bookmarks thus providing a nice navigation for the user. There is one slight problem though: when using, say, parts in the document, all chapters in that part ends up as a child of this part bookmark—including the index and bibliography. A simple fix to this is to add

```
\makeatletter
\renewcommand*{\toclevel@chapter}{-1}
\makeatother
```

just before the material you would like to pull out of the part tree.

A better solution is the bookmark package, add it to the preamble, and add

```
\bookmarksetup{startatroot}
```

before the stuff you want to have moved out of, say, a part.

Ten

Floats and captions

A float environment is a particular kind of box — one that LaTeX decides where it should go although you can provide hints as to where it should be placed; all other boxes are put at the point where they are defined. Within reason you can put what you like within a float but it is unreasonable, for example, to put a float inside another float. The standard classes provide two kinds of float environments, namely `figure` and `table`. The only difference between these is the naming and numbering of any caption within the environments — a `\caption` in a `figure` environment uses `\figurename` while a `\caption` in a `table` environment uses `\tablename`. Figures and tables are numbered sequentially but the two numbering schemes are independent of each other.

The class provides means of defining new kinds of floats. It also provides additional forms of captions for use both within and outside float environments together with handles for changing the style of captions.

10.1 New float environments

It is often forgotten that the LaTeX float environments come in both starred and unstarred forms. The unstarred form typesets the float contents in one column, which is the most usual form for a book. The starred form typesets the contents of the float across the top of both columns in a twocolumn document. In a onecolumn document there is no difference between the starred and unstarred forms.

`\newfloat[`⟨*within*⟩`]{`⟨*fenv*⟩`}{`⟨*ext*⟩`}{`⟨*capname*⟩`}`

The `\newfloat` command creates two new floating environments called ⟨*fenv*⟩ and ⟨*fenv**⟩. If there is not already a counter defined for ⟨*fenv*⟩ a new one will be created to be restarted by the counter ⟨*within*⟩, if that is specified. A caption within the environment will be written out to a file with extension ⟨*ext*⟩. The caption, if present, will start with ⟨*capname*⟩. For example, the `figure` float for the class is defined as:

```
\newfloat[chapter]{figure}{lof}{\figurename}
\renewcommand{\thefigure}{%
  \ifnum\c@chapter>\z@ \thechapter.\fi \@arabic\c@figure}
```

The last bit of the definition is internal code to make sure that if a figure is in the document before chapter numbering starts, then the figure number will not be preceeded by a non-existent chapter number.

The captioning style for floats defined with `\newfloat` is the same as for the figures and tables.

The \newfloat command generates several new commands, some of which are internal LaTeX commands. For convenience, assume that the command was called as

```
\newfloat{F}{X}{capname}
```

so F is the name of the float environment and also the name of the counter for the caption, and X is the file extension. The following float environment and related commands are then created.

\begin{F} float material \end{F} \begin{F*} float material \end{F*}

The new float environment is called F, and can be used as either \begin{F} or \begin{F*}, with the matching \end{F} or \end{F*}. It is given the standard default position specification of [tbp].

Xdepth

The Xdepth counter is analogous to the standard tocdepth counter in that it specifies that entries in a listing should not be typeset if their numbering level is greater than Xdepth. The default definition is

```
\setcounter{Xdepth}{1}
```

To have a subfloat of X appear in the listing do

```
\setcounter{Xdepth}{2}
```

As an example, suppose you wanted both figures (which come with the class), and diagrams. You could then do something like the following.

```
\newcommand{\diagramname}{Diagram}
\newcommand{\listdiagramname}{List of Diagrams}
\newlistof{listofdiagrams}{dgm}{\listdiagramname}
\newfloat{diagram}{dgm}{\diagramname}
\newlistentry{diagram}{dgm}{0}
\begin{document}
 ...
\listoffigures
\listfofdiagrams
 ...
\begin{diagram}
\caption{A diagram} \label{diag1}
 ...
\end{diagram}
As diagram~\ref{diag1} shows ...
```

\setfloatadjustment{⟨*floatname*⟩}{⟨*code*⟩}

Often it is useful to add some global configuration to a given type of float such that one will not have to add this to each and every float. For example to have all (floating) figures and tables automatically centered plus have all (floating) tables typeset in \small use

```
\setfloatadjustment{figure}{\centering}
\setfloatadjustment{table}{\small\centering}
```

10.1.1 Margin floats

We also provide two environments to insert an image or table into the margin (using `\marginpar`). The construction is inspired by the Tufte LaTeX collection.

`\begin{marginfigure}[`⟨*len*⟩`]` float material `\end{marginfigure}`
`\begin{margintable}[`⟨*len*⟩`]` float material `\end{margintable}`

Because this is inserted differently than the ordinary `figure` or `table` floats, one might get into the situation where a figure float inserted before a margin float, might float *past* the margin float and thus have different caption numbering. For this reason the margin float contain a float blocking device such that any unplaced floats are forced to be placed before we start typesetting a margin figure.

The `marginfigure` and `margintable` environments can of course be adjusted using `\setfloatadjustment`, default

```
\setfloatadjustment{marginfigure}{\centering}
\setfloatadjustment{margintable}{\centering}
```

It may be useful to adjust the captioning separately, for this we have added

`\setmarginfloatcaptionadjustment{`⟨*float*⟩`}{`⟨*code*⟩`}`

where ⟨*float*⟩ is `figure` or `table`. The intent is to enable the user to choose a different captioning style (or similar) within a margin float, for example typesetting the caption ragged left/right depending on the page.

This *left/right depending on the page* is a little hard to do, so for the `\marginpar` (which the margin float use internally) we provide the following two macros

`\setmpjustification{`⟨*at left of textblock*⟩`}{`⟨*at right of textblock*⟩`}`
`\mpjustification`

Basically `\mpjustification` execute ⟨*at left of textblock*⟩ when it is executed at the left of the text block and vice versa. For it to work the margin into which the `marginpar` should do, *has* to be specified using `marginparmargin`. The default is

```
\setmpjustification{\raggedleft}{\raggedright}
```

To have both a margin figure and its caption typeset ragged against the text block, use

```
\setfloatadjustment{marginfigure}{\mpjustification}
\setmarginfloatcaptionadjustment{figure}{\captionstyle{\mpjustification}}
```

It may be useful to allow hyphenation within the raggedness, which can be done using the ragged2e package and

```
\setmpjustification{\RaggedLeft}{\RaggedRight}
```

10.2 Setting off a float

Sometimes it is desireable to set off a float, more probably an illustration than a tabular, from its surroundings. The `framed` environment, described later in Chapter 15, might come in handy for this.

The following code produces the example Figures 10.1 and 10.2.

Figure 10.1: Example framed figure

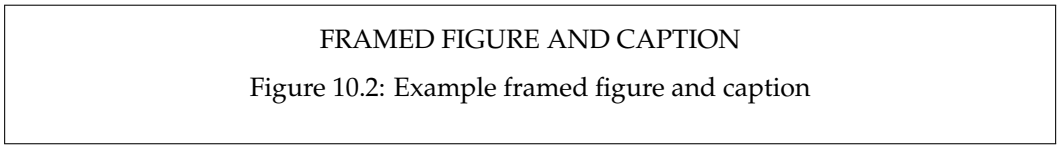


Figure 10.2: Example framed figure and caption

```
\begin{figure}
\centering
\begin{framed}\centering
FRAMED FIGURE
\end{framed}
\caption{Example framed figure}\label{fig:framef}
\end{figure}

\begin{figure}
\begin{framed}\centering
FRAMED FIGURE AND CAPTION
\caption{Example framed figure and caption}\label{fig:framefcap}
\end{framed}
\end{figure}
```

If framing seems overkill then you can use rules instead, as in the example code below which produces Figures 10.3 and 10.4.

```
\begin{figure}
\centering
\hrule\vspace{\onelineskip}
RULED FIGURE
\vspace{\onelineskip}\hrule
\vspace{\onelineskip}
\caption{Example ruled figure}\label{fig:rulef}
\end{figure}

\begin{figure}
\centering
\hrule\vspace{\onelineskip}
RULED FIGURE AND CAPTION
\vspace{\onelineskip}\hrule
\vspace{0.2pt}\hrule
\vspace{\onelineskip}
\caption{Example ruled figure and caption}\label{fig:rulefcap}
```

RULED FIGURE

Figure 10.3: Example ruled figure

RULED FIGURE AND CAPTION

Figure 10.4: Example ruled figure and caption

ILLUSTRATION 1 ILLUSTRATION 2

Figure 10.5: Example float with two illustrations

```
\hrule
\end{figure}
```

10.3 MULTIPLE FLOATS

You can effectively put what you like inside a float box. Normally there is just a single picture or tabular in a float but you can include as many of these as will fit inside the box.

Three typical cases of multiple figures/tables in a single float come to mind:

- Multiple illustrations/tabulars with a single caption.
- Multiple illustrations/tabulars each individually captioned.
- Multiple illustrations/tabulars with one main caption and individual subcaptions.

Figure 10.5 is an example of multiple illustrations in a single float with a single caption. The figure was produced by the following code.

```
\begin{figure}
\centering
\hspace*{\fill}
  {ILLUSTRATION 1} \hfill {ILLUSTRATION 2}
\hspace*{\fill}
\caption{Example float with two illustrations} \label{fig:mult1}
\end{figure}
```

GRAPHIC 1

Figure 10.6: Graphic 1 in a float

GRAPHIC 2

Figure 10.7: Graphic 2 in same float

The `\hspace*{\fill}` and `\hfill` commands were used to space the two illustrations equally. Of course `\includegraphics` or `tabular` environments could just as well be used instead of the `{ILLUSTRATION N}` text.

The following code produces Figures 10.6 and 10.7 which are examples of two separately captioned illustrations in one float.

```
\begin{figure}
\centering
\begin{minipage}{0.4\textwidth}
  \centering
  GRAPHIC 1
  \caption{Graphic 1 in a float} \label{fig:mult2}
\end{minipage}
\hfill
\begin{minipage}{0.4\textwidth}
  \centering
  GRAPHIC 2
  \caption{Graphic 2 in same float} \label{fig:mult3}
\end{minipage}
\end{figure}
```

In this case the illustrations (or graphics or tabulars) are put into separate `minipage` environments within the float, and the captions are also put within the `minipage`s. Note that any required `\label` must also be inside the `minipage`. If you wished, you could add yet another (main) caption after the end of the two `minipage`s.

It is slightly more complex if you want to put, say, both a tabulation captioned as a table and a graph, captioned as a figure, which illustrates the tabulation, as a float only permits one kind of caption. The class solves this problem by letting you define 'fixed' captions which are independent of the particular kind of the float. These are described in detail later.

Things do get a little trickier, though, if the bodies and/or the captions in a float are different heights (as in Figures 10.6 and 10.7) and you want to align them horizontally. Here are some examples.

This code produces Figures 10.8 and 10.9. The new `\hhrule` macro produces a rule twice as thick as `\hrule` does.

```
\newcommand*{\hhrule}{\hrule height 0.8pt}% double thickness

\begin{figure}
\hhrule \vspace{\onelineskip}
\null\hfill\parbox{0.45\linewidth}{%
  \centering
```

Aligned to the center of the right figure	This is the right figure which is taller than the first one (the one at the left)
Figure 10.8: Left center aligned	Figure 10.9: Right figure. This has more text than the adjacent caption (10.8) so the heights are unequal

```
  Aligned to the center of the right figure
}\hfill
\parbox{0.45\linewidth}{%
  \centering
   This is the right figure which is taller
   than the first one (the one at the left)
}\hfill\null
\vspace{\onelineskip}\hrule
\null\hfill\parbox[t]{0.4\linewidth}{%
  \caption{Left figure}\label{fig:left1}%
}\hfill
\parbox[t]{0.4\linewidth}{%
  \caption{Right figure. This has more text than the adjacent
           caption (\ref{fig:left1}) so the heights are unequal}%
           \label{fig:right1}%
}\hfill\null
\hhrule
\end{figure}
```

The following code produces Figures 10.10 and 10.11.

```
\begin{figure}
\hhrule \vspace{0.5\onelineskip}
\null\hfill\parbox[t]{0.45\linewidth}{%
  \centering
  Aligned to the top of the right figure
}\hfill
\parbox[t]{0.45\linewidth}{%
  \centering
   This is the right figure which is taller
   than the first one (the one at the left)
}\hfill\null
\vspace{0.5\onelineskip}\hrule
\null\hfill\parbox[t]{0.4\linewidth}{%
  \caption{Left top aligned}\label{fig:left2}%
}\hfill
```

Aligned to the top of the right figure	This is the right figure which is taller than the first one (the one at the left)
Figure 10.10: Left top aligned	Figure 10.11: Right figure. This has more text than the adjacent caption (10.10) so the heights are unequal

```
\parbox[t]{0.4\linewidth}{%
  \caption{Right figure. This has more text than the adjacent
           caption (\ref{fig:left2}) so the heights are unequal}%
           \label{fig:right2}%
}\hfill\null
\hhrule
\end{figure}
```

The next code produces Figures 10.12 and 10.13.

```
\begin{figure}
\hhrule \vspace{0.5\onelineskip}
\null\hfill\parbox[b]{0.45\linewidth}{%
  \centering
  Aligned to the bottom of the right figure
}\hfill
\parbox[b]{0.45\linewidth}{%
  \centering
   This is the right figure which is taller
   than the first one (the one at the left)
}\hfill\null
\vspace{0.5\onelineskip}\hrule
\null\hfill\parbox[t]{0.4\linewidth}{%
  \caption{Left bottom aligned}\label{fig:left3}%
}\hfill
\parbox[t]{0.4\linewidth}{%
  \caption{Right figure. This has more text than the adjacent
           caption (\ref{fig:left3}) so the heights are unequal}%
           \label{fig:right3}%
}\hfill\null
\hhrule
\end{figure}
```

`\newsubfloat{`⟨*float*⟩`}`

The `\newsubfloat` command creates subcaptions (`\subcaption`, `\subtop` and `\subbottom`) for use within the float environment ⟨*fenv*⟩ previously defined via

Aligned to the bottom of the right figure	This is the right figure which is taller than the first one (the one at the left)
Figure 10.12: Left bottom aligned	Figure 10.13: Right figure. This has more text than the adjacent caption (10.12) so the heights are unequal

`\newfloat[`⟨...⟩`]{`⟨*fenv*⟩`}{`⟨...⟩`}`. Subcaptions are discussed below in §10.9.

10.4 Where LaTeX puts floats

The general format for a float environment is:
`\begin{float}[`⟨*loc*⟩`] ... \end{float}` or for double column floats:
`\begin{float*}[`⟨*loc*⟩`] ... \end{float*}`
where the optional argument ⟨*loc*⟩, consisting of one or more characters, specifies a location where the float may be placed. Note that the multicol package only supports the starred floats and it will not let you have a single column float. The possible ⟨*loc*⟩ values are one or more of the following:

- `b` *bottom*: at the bottom of a page. This does not apply to double column floats as they may only be placed at the top of a page.
- `h` *here*: if possible exactly where the float environment is defined. It does not apply to double column floats.
- `p` *page*: on a separate page containing only floats (no text); this is called a *float page*.
- `t` *top*: at the top of a page.
- `!` make an extra effort to place the float at the earliest place specified by the rest of the argument.

The default for ⟨*loc*⟩ is `tbp`, so the float may be placed at the top, or bottom, or on a float page; the default works well 95% of the time. Floats of the same kind are output in definition order, except that a double column float may be output before a later single column float of the same kind, or *vice-versa*[1]. A float is never put on an earlier page than its definition but may be put on the same or later page of its definition. If a float cannot be placed, all suceeding floats will be held up, and LaTeX can store no more than 16 held up floats. A float cannot be placed if it would cause an overfull page, or it otherwise cannot be fitted according the float placement parameters. A `\clearpage` or `\cleardoublepage` or `\end{document}` flushes out all unprocessed floats, irrespective of the ⟨*loc*⟩ and float parameters, putting them on float-only pages.

`\setfloatlocations{`⟨*float*⟩`}{`⟨*locs*⟩`}`

You can set the location for all floats of type ⟨*float*⟩ to ⟨*locs*⟩ with the `\setfloatlocations` declaration. The class initialises these using:

[1] As of 2015 this has been fixed in the LaTeX kernel.

```
\setfloatlocations{figure}{tbp}
\setfloatlocations{table}{tbp}
```

`\suppressfloats[`⟨*pos*⟩`]`

You can use the command `\suppressfloats` to suppress floats at a given ⟨*pos*⟩ on the current page. `\suppressfloats[t]` prevents any floats at the top of the page and `\suppressfloats[b]` prevents any floats at the bottom of the page. The simple `\suppressfloats` prevents both top and bottom floats.

```
\FloatBlock
\FloatBlockAllowAbove
\FloatBlockAllowBelow
```

Contrary to `\suppressfloats` `\FloatBlock`[2] will block floats from passing this point, i.e. it demands LaTeX to place any unprocessed floats before proceeding. It is similar to `\clearpage` but it does not necessarily introduce a page break before proceeding.

`\FloatBlockAllowAbove` lessens the restriction a little, in a situation like this

```
\FloatBlock
some float here
```

`\FloatBlockAllowAbove` will allow the float to be placed at the top of the same page as `\FloatBlock`. `\FloatBlockAllowBelow` is the reverse situation.

It may be beneficial to be able to add `\FloatBlock` to sectional commands. This can be done via

`\setFloatBlockFor{`⟨*sectional name*⟩`}`

where ⟨*sectional name*⟩ is *without* the \, i.e.

```
\setFloatBlockFor{section}
```

The flafter package, which should have come with your LaTeX distribution, provides a means of preventing floats from moving backwards from their definition position in the text. This can be useful to ensure, for example, that a float early in a `\section{...}` is not typeset before the section heading.

Figures 10.14 and 10.15 illustrate the many float parameters and Table 10.1 lists the float parameters and the typical standard default values. The lengths controlling the spaces surroundind floats are listed in Table 10.2; typical values are shown as they depend on both the class and the size option.

Given the displayed defaults, the height of a top float must be less than 70% of the textheight and there can be no more than 2 top floats on a text page. Similarly, the height of a bottom float must not exceed 30% of the textheight and there can be no more than 1 bottom float on a text page. There can be no more than 3 floats (top, bottom and here) on the page. At least 20% of a text page with floats must be text. On a float page (one that has no text, only floats) the sum of the heights of the floats must be at least 50% of the textheight. The floats on a float page should be vertically centered.

[2] Yes, it *is* the same as `\FloatBarrier` from the placeins package, kudos to Donald Arseneau. For various reasons we cannot emulate the placeins package and its options, thus we have verbatimly copied and renamed it instead.

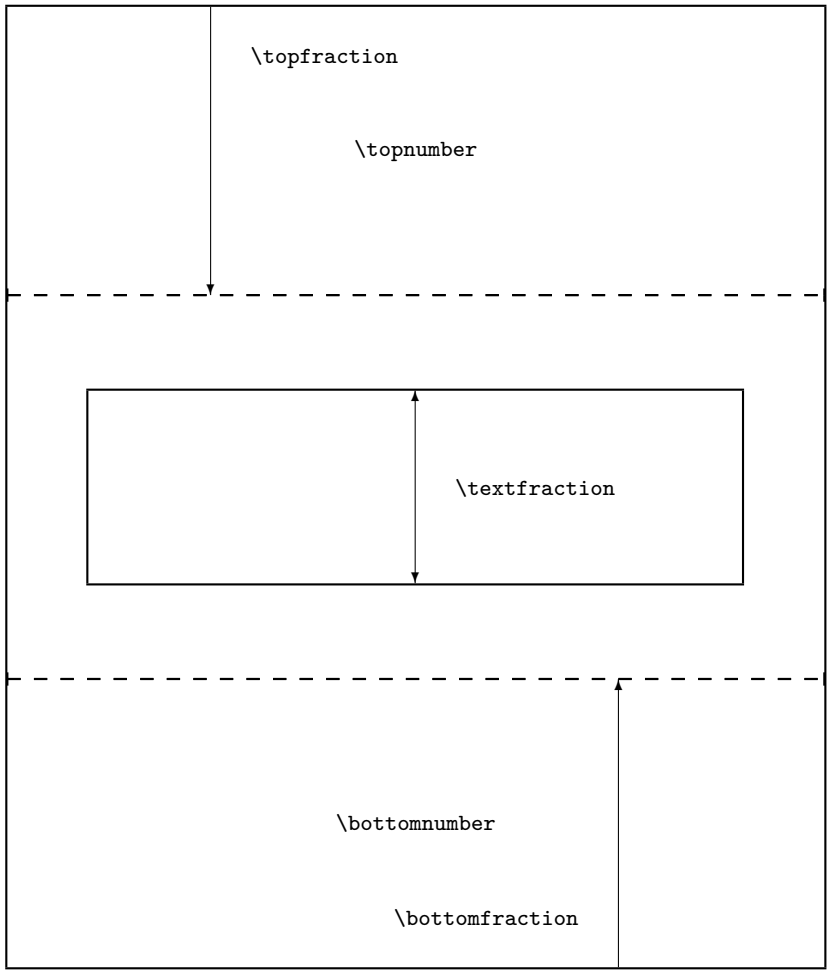


Figure 10.14: Float and text page parameters

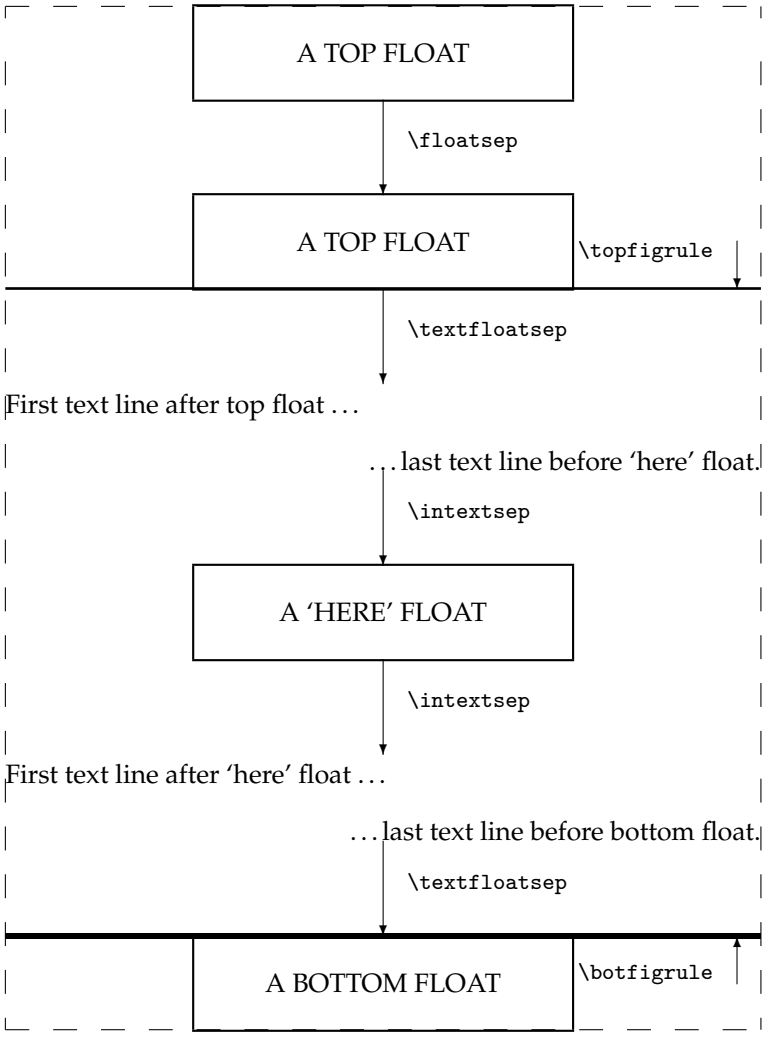


Figure 10.15: Float parameters

Table 10.1: Float placement parameters

Parameter	Controls	Default
	Counters — change with `\setcounter`	
`topnumber`	max number of floats at top of a page	2
`bottomnumber`	max number of floats at bottom of a page	1
`totalnumber`	max number of floats on a text page	3
`dbltopnumber`	like `topnumber` for double column floats	2
	Commands — change with `\renewcommand`	
`\topfraction`	max fraction of page reserved for top floats	0.7
`\bottomfraction`	max fraction of page reserved for bottom floats	0.3
`\textfraction`	min fraction of page that must have text	0.2
`\dbltopfraction`	like `\topfraction` for double column floats floats	0.7
`\floatpagefraction`	min fraction of a float page that must have float(s)	0.5
`\dblfloatpagefraction`	like `\floatpagefraction` for double column floats	0.5

Table 10.2: Float spacing parameters

Parameter	Controls	Default
	Text page lengths — change with `\setlength`	
`\floatsep`	vertical space between floats	12pt
`\textfloatsep`	vertical space between a top (bottom) float and suceeding (preceeding) text	20pt
`\intextsep`	vertical space above and below an `h` float	12pt
`\dblfloatsep`	like `\floatsep` for double column floats	12pt
`\dbltextfloatsep`	like `\textfloatsep` for double column floats	20pt
	Float page lengths — change with `\setlength`	
`\@fptop`	space at the top of the page	`0pt plus 1fil`
`\@fpsep`	space between floats	`8pt plus 2fil`
`\@fpbot`	space at the bottom of the page	`0pt plus 1fil`
`\@dblfptop`	like `\@fptop` for double column floats	`0pt plus 1fil`
`\@dblfpsep`	like `\@fpsep` for double column floats	`8pt plus 2fil`
`\@dblfpbot`	like `\@fpbot` for double column floats	`0pt plus 1fil`

Under certain extreme and unlikely conditions and with the defaults LaTeX might have trouble finding a place for a float. Consider what will happen if a float is specified as a bottom float and its height is 40% of the textheight and this is followed by a float whose height is 90% of the textheight. The first is too large to actually go at the bottom of a text page but too small to go on a float page by itself. The second has to go on a float page but it is too large to share the float page with the first float. LaTeX is stuck!

At this point it is worthwhile to be precise about the effect of a one character ⟨*loc*⟩ argument:

`b` means: 'put the float at the bottom of a page with some text above it, and nowhere else'. The float must fit into the `\bottomfraction` space otherwise it and subsequent floats will be held up.

`h` means: 'put the float at this point and nowhere else'. The float must fit into the space left on the page otherwise it and subsequent floats will be held up.

`p` means: 'put the float on a page that has no text but may have other floats on it'. There must be at least '`\floatpagefraction`' worth of floats to go on a float only page before the float will be output.

`t` means: 'put the float at the top of a page with some text below it, and nowhere else'. The float must fit into the `\topfraction` space otherwise it and subsequent floats will be held up.

`!`... means: 'ignore the `\...fraction` values for this float'.

You must try and pick a combination from these that will let LaTeX find a place to put your floats. However, you can also change the float parameters to make it easier to find places to put floats. Some examples are:

- Decrease `\textfraction` to get more 'float' on a text page, but the sum of `\textfraction` and `\topfraction` and the sum of `\textfraction` and `\bottomfraction` should not exceed 1.0, otherwise the placement algorithm falls apart. A minimum value for `\textfraction` is about 0.10 — a page with less than 10% text looks better with no text at all, just floats.
- Both `\topfraction` and `\bottomfraction` can be increased, and it does not matter if their sum exceeds 1.0. A good typographic style is that floats are encouraged to go at the top of a page, and a better balance is achieved if the float space on a page is larger at the top than the bottom.
- Making `\floatpagefraction` too small might have the effect of a float page just having one small float. However, to make sure that a float page never has more than one float on it, do:

```
\renewcommand{\floatpagefraction}{0.01}
\setlength{\@fpsep}{\textheight}
```

- Setting `\@fptop` and `\@dblftop` to `0pt`, `\@fpsep` to `8pt`, and `\@fpbot` and `\@dblfpbot` to `0pt plus 1fil` will force floats on a float page to start at the top of the page.
- Setting `\@fpbot` and `\@dblfpbot` to `0pt`, `\@fpsep` to `8pt`, and `\@fptop` and `\@dblfptop` to `0pt plus 1fil` will force floats on a float page to the bottom of the page.

If you are experimenting, a reasonable starting position is:

```
\setcounter{topnumber}{3}
\setcounter{bottomnumber}{2}
\setcounter{totalnumber}{4}
\renewcommand{\topfraction}{0.85}
\renewcommand{\bottomfraction}{0.5}
\renewcommand{\textfraction}{0.15}
\renewcommand{\floatpagefraction}{0.7}
```

and similarly for double column floats if you will have any. Actually, there is no need to try these settings as they are the default for this class.

One of LaTeX's little quirks is that on a text page, the 'height' of a float is its actual height plus `\textfloatsep` or `\floatsep`, while on a float page the 'height' is the actual height. This means that when using the default ⟨*loc*⟩ of `[tbp]` at least one of the text page float fractions (`\topfraction` and/or `\bottomfraction`) must be larger than the `\floatpagefraction` by an amount sufficient to take account of the maximum text page separation value.

10.5 Captions

Some publishers require, and some authors prefer, captioning styles other than the one style provided by standard LaTeX. Further, some demand that documents that include multi-part tables use a *continuation caption* on all but the first part of the multi-part table. For the times where such a table is specified by the author as a set of tables, the class provides a simple 'continuation' caption command to meet this requirement. It also provides a facility for an 'anonymous' caption which can be used in any float environment. Captions can be defined that are suitable for use in non-float environments, such as placing a picture in a minipage and captioning it just as though it had been put into a normal figure environment.

The commands described below are very similar to those supplied by the ccaption package [Wil01d].

As an alternative (it has more features that what the class offers) we recommend the caption package by Axel Sommerfeldt. In fact if the caption package is loaded, the class will detect it and reset it self such that caption can take over. This also means that all the macros described in the next section will have no effect if caption is loaded (either intentionally or unintentionally).

10.6 Caption styling

Just as a reminder, the default appearance of a caption for, say, a table looks like this:

Table 11.7: Title for the table

That is, it is typeset in the normal body font, with a colon after the number.

The class uses the following to specify the standard LaTeX caption style:

```
\captionnamefont{}
\captiontitlefont{}
\captionstyle{}
```

```
\captionwidth{\linewidth}
\normalcaptionwidth
\normalcaption
\captiondelim{: }
```

These macros are explained in detail below.

\captiondelim{⟨*delim*⟩}

The default captioning style is to put a delimeter in the form of a colon between the caption number and the caption title. The command `\captiondelim` can be used to change the delimeter. For example, to have an en-dash instead of the colon, `\captiondelim{-- }` will do the trick. Notice that no space is put between the delimeter and the title unless it is specified in the ⟨*delim*⟩ parameter. The class initially specifies `\captiondelim{: }` to give the normal delimeter.

\captionnamefont{⟨*fontspec*⟩}

The ⟨*fontspec*⟩ specified by `\captionnamefont` is used for typesetting the caption name; that is, the first part of the caption up to and including the delimeter (e.g., the portion 'Table 3:'). ⟨*fontspec*⟩ can be any kind of font specification and/or command and/or text. This first part of the caption is treated like:

```
{\captionnamefont Table 3: }
```

so font declarations, not font text-style commands, are needed for ⟨*fontspec*⟩. For instance,

```
\captionnamefont{\Large\sffamily}
```

to specify a large sans-serif font. The class initially specifies `\captionnamefont{}` to give the normal font.

\captiontitlefont{⟨*fontspec*⟩}

Similarly, the ⟨*fontspec*⟩ specified by `\captiontitlefont` is used for typesetting the title text of a caption. For example, `\captiontitlefont{\itshape}` for an italic title text. The class initially specifies `\captiontitlefont{}` to give the normal font.

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

By default the name and title of a caption are typeset as a block (non-indented) paragraph. `\captionstyle` can be used to alter this. Sensible values for ⟨*style*⟩ are: `\centering`, `\raggedleft` or `\raggedright` for styles corresponding to these declarations. The `\centerlastline` style gives a block paragraph but with the last line centered. The class initially specifies `\captionstyle{}` to give the normal block paragraph style.

If a caption is less than one line in length it may look odd if the ⟨*style*⟩ is `\raggedright`, say, as it will be left justified. The optional ⟨*short*⟩ argument to `\captionstyle` can be used to specify the style for such short captions if it should differ from that for multiline captions. For example, I think that short captions look better centered:

```
\captionstyle[\centering]{\raggedright}
```

\hangcaption \indentcaption{⟨*length*⟩} \normalcaption

The declaration \hangcaption causes captions to be typeset with the second and later lines of a multiline caption title indented by the width of the caption name. The declaration \indentcaption will indent title lines after the first by ⟨*length*⟩. These commands are independent of the \captionstyle{...} and have no effect on short captions. Note that a caption will not be simultaneously hung and indented. The \normalcaption declaration undoes any previous \hangcaption or \indentcaption declaration. The class initially specifies \normalcaption to give the normal non-indented paragraph style.

```
\changecaptionwidth
\captionwidth{⟨length⟩}
\normalcaptionwidth
```

Issuing the declaration \changecaptionwidth causes the captions to be typeset within a total width ⟨*length*⟩ as specified by \captionwidth. Issuing the declaration \normalcaptionwidth causes captions to be typeset as normal full width captions. The class initially specifies

```
\normalcaptionwidth
\captionwidth{\linewidth}
```

to give the normal width. If a caption is being set within the side captioned environments from the sidecap package [NG98] then it must be a \normalcaptionwidth caption.

```
\precaption{⟨pretext⟩}
\captiontitlefinal{⟨text⟩}
\postcaption{⟨posttext⟩}
```

The commands \precaption and \postcaption specify ⟨*pretext*⟩ and ⟨*posttext*⟩ that will be processed at the start and end of a caption. For example

```
\precaption{\rule{\linewidth}{0.4pt}\par}
\postcaption{\rule{\linewidth}{0.4pt}}
```

will draw a horizontal line above and below the captions. The class initially specifies

```
\precaption{}
\postcaption{}
```

to give the normal appearance.

The argument to \captiontitlefinal is put immediately after the title text but will not appear in the LoF or LoT. The default is

```
\captiontitlefinal{}
```

but it could be used instead as, say

```
\captiontitlefinal{.}
```

to put a period (full stop) after the title.

If any of the above commands are used in a float, or other, environment their effect is limited to the environment. If they are used in the preamble or the main text, their effect persists until replaced by a similar command with a different parameter value. The commands do not affect the appearance of the title in any 'List of...'.

```
\\[⟨length⟩]
\\*[⟨length⟩]
```

Table 10.3
REDESIGNED TABLE CAPTION STYLE

three	III
five	V
eight	VIII

The normal LaTeX command \\ can be used within the caption text to start a new line. Remember that \\ is a fragile command, so if it is used within text that will be added to a 'List of...' it must be protected. As examples:

```
\caption{Title with a \protect\\ new line in
         both the body and List of}
\caption[List of entry with no new line]%
        {Title with a \\ new line}
\caption[List of entry with a \protect\\ new line]%
        {Title text}
```

Effectively, a caption is typeset as though it were:

```
\precaption
{\captionnamefont NAME NUMBER\captiondelim}
{\captionstyle\captiontitlefont THE TITLE\captiontitlefinal}
\postcaption
```

Replacing the above commands by their defaults leads to the simple format:
`{NAME NUMBER: }{THE TITLE}`

As well as using the styling commands to make simple changes to the captioning style, more noticeable modifications can also be made. To change the captioning style so that the name and title are typeset in a sans font it is sufficient to do:

```
\captionnamefont{\sffamily}
\captiontitlefont{\sffamily}
```

A more obvious change in styling is shown in Table 10.3, which was coded as:

```
\begin{table}
\centering
\captionnamefont{\sffamily}
\captiondelim{}
\captionstyle{\\}
\captiontitlefont{\scshape}
\setlength{\belowcaptionskip}{10pt}
\caption{Redesigned table caption style} \label{tab:style}
\begin{tabular}{lr} \toprule
 ...
\end{table}
```

This leads to the approximate caption format (processed within `\centering`):

```
{\sffamily NAME NUMBER}{\\ \scshape THE TITLE}
```

Note that the newline command (\\) cannot be put in the first part of the format (i.e., the {\sffamily NAME NUMBER}); it has to go into the second part, which is why it is specified via \captionstyle{\\} and not \captiondelim{\\}.

If a mixture of captioning styles will be used you may want to define a special caption command for each non-standard style. For example for the style of the caption in Table 10.3:

```
\newcommand{\mycaption}[2][\@empty]{
  \captionnamefont{\sffamily\hfill}
  \captiondelim{\hfill}
  \captionstyle{\centerlastline\\}
  \captiontitlefont{\scshape}
  \setlength{\belowcaptionskip}{10pt}
  \ifx \@empty#1 \caption{#2}\else \caption[#1]{#2}\fi}
```

Remember that any code that involves the @ sign must be either in a package (sty) file or enclosed between a \makeatletter ... \makeatother pairing (see §E.4).

The code for the Table 10.3 example can now be written as:

```
\begin{table}
\centering
\mycaption{Redesigned table caption style} \label{tab:style}
\begin{tabular}{lr} \toprule
 ...
\end{table}
```

Note that in the code for \mycaption I have added two \hfill commands and \centerlastline compared with the original specification. It turned out that the original definitions worked for a single line caption but not for a multiline caption. The additional commands makes it work in both cases, forcing the name to be centered as well as the last line of a multiline title, thus giving a balanced appearence.

10.7 Continuation captions and legends

\contcaption{⟨*text*⟩}

The \contcaption command can be used to put a 'continued' or 'concluded' caption into a float environment. It neither increments the float number nor makes any entry into a float listing, but it does repeat the numbering of the previous \caption command.

Table 10.4 illustrates the use of the \contcaption command. The table was produced from the following code.

```
\begin{table}
\centering
\caption{A multi-part table} \label{tab:m}
\begin{tabular}{lc} \toprule
 just a single line & 1 \\ \bottomrule
\end{tabular}
\end{table}
```

Table 10.4: A multi-part table

just a single line	1

Table 10.4: Continued

just a single line	2

Table 10.4: Concluded

just a single line	3

```
\begin{table}
\centering
\contcaption{Continued}
\begin{tabular}{lc} \toprule
 just a single line & 2 \\ \bottomrule
\end{tabular}
\end{table}

\begin{table}
\centering
\contcaption{Concluded}
\begin{tabular}{lc} \toprule
 just a single line & 3 \\ \bottomrule
\end{tabular}
\end{table}
```

`\legend{`⟨*text*⟩`}`

The `\legend` command is intended to be used to put an anonymous caption, or legend into a float environment, but may be used anywhere.

For example, the following code was used to produce the two-line Table 10.5. The `\legend` command can be used within a float independently of any `\caption` command.

```
\begin{table}
  \centering
  \caption{Another table} \label{tab:legend}
  \begin{tabular}{lc} \toprule
```

Table 10.5: Another table

A legendary table	5
with two lines	6

The legend

Legendary table

An anonymous table	5
with two lines	6

```
   A legendary table & 5 \\
   with two lines    & 6 \\ \bottomrule
   \end{tabular}
   \legend{The legend}
\end{table}
```

Captioned floats are usually thought of in terms of the `table` and `figure` environments. There can be other kinds of float. As perhaps a more interesting example, the following code produces the titled marginal note which should be displayed near here.

```
\marginpar{\legend{LEGEND}
           This is a marginal note with a legend.}
```

LEGEND

This is a marginal note with a legend.

If you want the legend text to be included in the 'List of…' you can do it like this with the `\addcontentsline` macro.

```
\legend{Legend title}
% left justified
\addcontentsline{lot}{table}{Legend title}    % or
% indented
\addcontentsline{lot}{table}{\protect\numberline{}Legend title}
```

The first of these forms will align the first line of the legend text under the normal table numbers. The second form will align the first line of the legend text under the normal `table` titles. In either case, second and later lines of a multi-line text will be aligned under the normal title lines.

As an example, the Legendary table is produced by the following code:

```
\begin{table}
\centering
\captiontitlefont{\sffamily}
\legend{Legendary table}
\addcontentsline{lot}{table}{Legendary table (toc 1)}
\addcontentsline{lot}{table}{\protect\numberline{}
```

Table: *Named legendary table*

seven	VII
eight	VIII

```
                              Legendary table (toc 2)}
\begin{tabular}{lc} \toprule
   An anonymous table & 5 \\
   with two lines     & 6 \\ \bottomrule
\end{tabular}
\end{table}
```

Look at the List of Tables to see how the two forms of `\addcontentsline` are typeset.

`\namedlegend[`⟨*short-title*⟩`]{`⟨*long-title*⟩`}`

As a convenience, the `\namedlegend` command is like the `\caption` command except that it does not number the caption and, by default, puts no entry into a 'List of…' file. Like the `\caption` command, it picks up the name to be prepended to the title text from the float environment in which it is called (e.g., it will use `\tablename` if called within a `table` environment). The following code is the source of the *Named legendary table*.

```
\begin{table}
\centering
\captionnamefont{\sffamily}
\captiontitlefont{\itshape}
\namedlegend{Named legendary table}
\begin{tabular}{lr} \toprule
seven & VII \\
eight & VIII \\ \bottomrule
\end{tabular}
\end{table}
```

`\flegfloat{`⟨*name*⟩`}`
`\flegtocfloat{`⟨*title*⟩`}`

The macro `\flegfloat`, where `float` is the name of a float environment (e.g., `figure`) is called by the `\namedlegend` macro. It is provided as a hook that defines the ⟨*name*⟩ to be used as the name in `\namedlegend`. Two defaults are provided, `\flegtable` and `\flegfigure` defined as:

```
\newcommand{\flegtable}{\tablename}
\newcommand{\flegfigure}{\figurename}
```

which may be altered via `\renewcommand` if desired.

The macro `\flegtocfloat`, where again `float` is the name of a float environment (e.g., `table`) is also called by the `\namedlegend` macro. It is provided as a hook that can be used to add ⟨*title*⟩ to the 'List of…'. Two examplars are provided, `\flegtocfigure`

and \flegtoctable. By default they are defined to do nothing, and can be changed via \renewcommand. For instance, one could be changed for tables as:

```
\renewcommand{\flegtoctable}[1]{
  \addcontentsline{lot}{table}{#1}}
```

The \legend command produces a plain, unnumbered heading. It can also be useful sometimes to have named and numbered captions outside a floating environment, perhaps in a minipage, if you want the table or picture to appear at a precise location in your document.

\newfixedcaption[⟨*capcommand*⟩]{⟨*command*⟩}{⟨*float*⟩}
\renewfixedcaption[⟨*capcommand*⟩]{⟨*command*⟩}{⟨*float*⟩}
\providefixedcaption[⟨*capcommand*⟩]{⟨*command*⟩}{⟨*float*⟩}

The \newfixedcaption command, and its friends, can be used to create or modify a captioning ⟨*command*⟩ that may be used outside the float environment ⟨*float*⟩. Both the environment ⟨*float*⟩ and a captioning command, ⟨*capcommand*⟩, for that environment must have been defined before calling \newfixedcaption. Note that \namedlegend can be used as ⟨*capcommand*⟩.

For example, to define a new \figcaption command for captioning pictures outside the figure environment, do

```
\newfixedcaption{\figcaption}{figure}
```

The optional ⟨*capcommand*⟩ argument is the name of the float captioning command that is being aliased. It defaults to \caption. As an example of where the optional argument is required, if you want to create a new continuation caption command for non-floating tables, say \ctabcaption, then do

```
\newfixedcaption[\contcaption]{\ctabcaption}{table}
```

Captioning commands created by \newfixedcaption will be named and numbered in the same style as the original ⟨*capcommand*⟩, can be given a \label, and will appear in the appropriate 'List of…'. They can also be used within floating environments, but will not use the environment name as a guide to the caption name or entry into the 'List of…'. For example, using \ctabcaption in a figure environment will still produce a **Table…** named caption.

Sometimes captions are required on the opposite page to a figure, and a fixed caption can be useful in this context. For example, if figure captions should be placed on an otherwise empty page immediately before the actual figure, then this can be accomplished by the following hack:

```
\newfixedcaption{\figcaption}{figure}
 ...
\afterpage{ % fill current page then flush pending floats
  \clearpage
  \begin{midpage}  %  vertically center the caption
  \figcaption{The caption} %  the caption
  \end{midpage}
  \clearpage
  \begin{figure}THE FIGURE, NO CAPTION HERE\end{figure}
  \clearpage
```

```
}  % end of \afterpage
```

Note that the afterpage package [Car95] is needed, which is part of the required tools bundle. The midpage package supplies the `midpage` environment, which can be simply defined as:

```
\newenvironment{midpage}{\vspace*{\fill}}{\vspace*{\fill}}
```

The code, in particular the use of `\clearpage`, might need adjusting to meet your particular requirements.

- `\clearpage` gets you to the next page, which may be odd or even.
- `\cleardoublepage` gets you to the next odd-numbered page.
- `\cleartoevenpage` ensures that you get to the next even-numbered page.

As a word of warning, if you mix both floats and fixed environments with the same kind of caption you have to ensure that they get printed in the correct order in the final document. If you do not do this, then the 'List of…' captions will come out in the wrong order (the lists are ordered according the page number in the typeset document, *not* your source input order).

10.8 BILINGUAL CAPTIONS

Some documents require bilingual (or more) captions. The class provides a set of commands for bilingual captions. Extensions to the set, perhaps to support trilingual captioning, are left as an exercise for the document author. Essentially, the bilingual commands call the `\caption` command twice, once for each language.

Several commands for bilingual captions are provided. They all produce the same appearance in the text but differ in what they put into the 'List of…'.

```
\bitwonumcaption[⟨label⟩]{⟨short1⟩}{⟨long1⟩}%
{⟨NAME⟩}{⟨short2⟩}{⟨long2⟩}
\bionenumcaption[⟨label⟩]{⟨short1⟩}{⟨long1⟩}%
{⟨NAME⟩}{⟨short2⟩}{⟨long2⟩}
```

Bilingual captions can be typeset by the `\bitwonumcaption` command which has six arguments. The first, optional argument ⟨*label*⟩, is the name of a label, if required. ⟨*short1*⟩ and ⟨*long1*⟩ are the short (i.e., equivalent to the optional argument to the `\caption` command) and long caption texts for the main language of the document. The value of the ⟨*NAME*⟩ argument is used as the caption name for the second language caption, while ⟨*short2*⟩ and ⟨*long2*⟩ are the short and long caption texts for the second language. For example, if the main and secondary languages are English and German and a figure is being captioned:

```
\bitwonumcaption{Short}{Long}{Bild}{Kurz}{Lang}
```

If the short title text(s) is not required, then leave the appropriate argument(s) either empty or as one or more spaces, like:

```
\bitwonumcaption[fig:bi1]{}{Long}{Bild}{  }{Lang}
```

Both language texts are entered into the appropriate 'List of…', and both texts are numbered.

Figure 10.16, typeset from the following code, is an example.

EXAMPLE FIGURE WITH BITWONUMCAPTION

Figure 10.16: Long \bitwonumcaption

Bild 10.16: Lang \bitwonumcaption

EXAMPLE FIGURE WITH BIONENUMCAPTION

Figure 10.17: Long English \bionenumcaption

Bild 10.17: Lang Deutsch \bionenumcaption

```
\begin{figure}
\centering
EXAMPLE FIGURE WITH BITWONUMCAPTION
\bitwonumcaption[fig:bi1]%
    {}{Long \cs{bitwonumcaption}}%
    {Bild}{  }{Lang \cs{bitwonumcaption}}
\end{figure}
```

Both \bionenumcaption and \bitwonumcaption take the same arguments. The difference between the two commands is that \bionenumcaption does not number the second language text in the 'List of...'. Figure 10.17, typeset from the following, is an example of this.

```
\begin{figure}
\centering
EXAMPLE FIGURE WITH BIONENUMCAPTION
\bionenumcaption[fig:bi3]%
    {}{Long English \cs{bionenumcaption}}%
    {Bild}{  }{Lang Deutsch \cs{bionenumcaption}}
 \end{figure}
```

\bicaption[⟨*label*⟩]{⟨*short1*⟩}{⟨*long1*⟩}%
{⟨*NAME*⟩}{⟨*long2*⟩}

When bilingual captions are typeset via the \bicaption command the second language text is not put into the 'List of...'. The command takes 5 arguments. The optional ⟨*label*⟩ is for a label if required. ⟨*short1*⟩ and ⟨*long1*⟩ are the short and long caption texts for the main language of the document. The value of the ⟨*NAME*⟩ argument is used as the caption name for the second language caption. The last argument, ⟨*long2*⟩, is the caption text for the second language (which is not put into the 'List of...').

For example, if the main and secondary languages are English and German:

```
\bicaption{Short}{Long}{Bild}{Langlauf}
```

If the short title text is not required, then leave the appropriate argument either empty or as one or more spaces.

EXAMPLE FIGURE WITH A RULED BICAPTION

Figure 10.18: Longingly

Bild 10.18: Langlauf

Figure 10.18 is an example of using `\bicaption` and was produced by the following code:

```
\begin{figure}
\centering
   EXAMPLE FIGURE WITH A RULED BICAPTION
\precaption{\rule{\linewidth}{0.4pt}\par}
\midbicaption{\precaption{}%
              \postcaption{\rule{\linewidth}{0.4pt}}}
\bicaption[fig:bi2]%
    {Short English \cs{bicaption}}{Longingly}%
    {Bild}{Langlauf}
\end{figure}
```

`\bicontcaption{`⟨*long1*⟩`}%`
`{`⟨*NAME*⟩`}{`⟨*long2*⟩`}`

Bilingual continuation captions can be typeset via the `\bicontcaption` command. In this case, neither language text is put into the 'List of…'. The command takes 3 arguments. ⟨*long1*⟩ is the caption text for the main language of the document. The value of the ⟨*NAME*⟩ argument is used as the caption name for the second language caption. The last argument, ⟨*long2*⟩, is the caption text for the second language. For example, if the main and secondary languages are again English and German:

```
\bicontcaption{Continued}{Bild}{Fortgefahren}
```

`\midbicaption{`⟨*text*⟩`}`

The bilingual captions are implemented by calling `\caption` twice, once for each language. The command `\midbicaption`, which is similar to the `\precaption` and `\postcaption` commands, is executed just before calling the second `\caption`. Among other things, this can be used to modify the style of the second caption with respect to the first. For example, if there is a line above and below normal captions, it is probably undesirable to have a double line in the middle of a bilingual caption. So, for bilingual captions the following may be done within the float before the caption:

```
\precaption{\rule{\linewidth}{0.4pt}\par}
\postcaption{}
\midbicaption{\precaption{}%
              \postcaption{\rule{\linewidth}{0.4pt}}}
```

This sets a line before the first of the two captions, then the `\midbicaption{...}` nulls the pre-caption line and adds a post-caption line for the second caption. The class initially specifies `\midbicaption{}`.

10.9 SUBCAPTIONS

The subfigure package enables the captioning of sub-figures within a larger figure, and similarly for tables. The subfigure package may be used with the class, or you can use the class commands described below; these commands can only be used inside a float environment for which a subfloat[3] has been specified via `\newsubfloat`.

`\subcaption[`⟨*list-entry*⟩`]{`⟨*subtitle*⟩`}`

The `\subcaption` command is similar to the `\caption` command and can only be used inside a float environment. It typesets an identified ⟨*subtitle*⟩, where the identification is an alphabetic character enclosed in parentheses. If the optional ⟨*list-entry*⟩ argument is present, ⟨*list-entry*⟩ is added to the caption listings for the float. If it is not present, then ⟨*subtitle*⟩ is added to the listing.

The ⟨*subtitle*⟩ is typeset within a box which is the width of the surrounding environment, so `\subcaption` should only be used within a fixed width box of some kind, for example a `minipage` as shown below.

```
\begin{figure}
\centering
\begin{minipage}{0.3\textwidth}
  \verb?Some verbatim text?
  \subcaption{First text}
\end{minipage}
\hfill
\begin{minipage}{0.3\textwidth}
  \verb?More verbatim text?
  \subcaption{Second text}
\end{minipage}
\caption{Verbatim texts}
\end{figure}
```

As the example code shows, the `\subcaption` command provides a means of putting verbatim elements into subfigures.

`\subtop[`⟨*list-entry*⟩`][`⟨*subtitle*⟩`]{`⟨*text*⟩`}`
`\subbottom[`⟨*list-entry*⟩`][`⟨*subtitle*⟩`]{`⟨*text*⟩`}`

The command `\subtop` puts a subcaption identifier on top of ⟨*text*⟩. If both optional arguments are present, ⟨*list-entry*⟩ will be added to the appropriate listing, and ⟨*subtitle*⟩ is placed above the ⟨*text*⟩ with the identifier. If only one optional argument is present this is treated as being ⟨*subtitle*⟩; the identifier and ⟨*subtitle*⟩ are placed above the ⟨*text*⟩ and ⟨*subtitle*⟩ is added to the listing. Regardless of the optional arguments the identifier is always added to the listing and placed above the ⟨*text*⟩.

[3] See §10.3.

The \subbottom command is identical to \subtop except that the identifier, and potentially the ⟨*subtitle*⟩, is placed below the ⟨*text*⟩. Note that verbatim text cannot be used in the ⟨*text*⟩ argument to \subbottom or \subtop.

The main caption can be at either the top or the bottom of the float. The positioning of the main and subcaptions are independent. For example

```
\begin{figure}
  \subbottom{...}   % captioned as (a) below
  \subtop{...}      % captioned as (b) above
  \caption{...}
\end{figure}
```

If a figure that includes subfigures is itself continued then it may be desirable to continue the captioning of the subfigures. For example, if Figure 3 has three subfigures, say A, B and C, and Figure 3 is continued then the subfigures in the continuation should be D, E, etc.

\contsubcaption[⟨*list-entry*⟩]{⟨*subtitle*⟩}
\contsubtop[⟨*list-entry*⟩][⟨*subtitle*⟩]{⟨*text*⟩}
\contsubbottom[⟨*list-entry*⟩][⟨*subtitle*⟩]{⟨*text*⟩}
\subconcluded

The \contsubcaption, \contsuptop and \contsubbottom commands are the continued versions of the respective subcaptioning commands. These continue the subcaption numbering scheme across (continued) floats. In any event, the main caption can be at the top or bottom of the float. The \subconcluded command is used to indicate that the continued (sub) float has been concluded and the numbering scheme is reinitialized. The command should be placed immediately before the end of the last continued environment. For example:

```
\begin{figure}
\subbottom{...}  captioned as (a) below
\subbottom{...}  captioned as (b) below
\caption{...}
\end{figure}
\begin{figure}
\contsubtop{...}  captioned as (c) above
\contsubtop{...}  captioned as (d) above
\contcaption{Concluded}
\subconcluded
\end{figure}
...
\begin{table}
\caption{...}
\subtop{...}     captioned as (a) above
\subbottom{...}  captioned as (b) below
\end{table}
```

\label(⟨*bookmark*⟩){⟨*labstr*⟩}
\subcaptionref{⟨*labstr*⟩}

(a) Subfigure 1 (b) Subfigure 2

Figure 10.19: Figure with two subfigures

A `\label` command may be included in the ⟨*subtitle*⟩ argument of the subcaptioning commands. Using the normal `\ref` macro to refer to the label will typeset the number of the float (obtained from a `\label`ed main `\caption`) and the subcaption identifier. If the `\subcaptionref` macro is used instead of `\ref` then only the subcaption identifier is printed.

In cases where the hyperref package is used, the `\label` command when used inside the ⟨*subtitle*⟩ argument can take an optional ⟨*bookmark*⟩ argument, *enclosed in parenthese* not *square brackets*, which will create a bookmark field of the form 'Subfigure 4.7(d)'.

As an example to show the difference between `\subcaptionref` and `\ref`, Figure 10.19 and the paragraph immediately following this one were produced by the code shown below.

Figure 10.19 has two subfigures, namely 10.19(a) and (b).

```
   Figure \ref{fig:twosubfig} has two subfigures,
namely \ref{sf:1} and \subcaptionref{sf:2}.
\begin{figure}
\centering
\subbottom[Subfigure 1\label{sf:1}]{\fbox{SUBFIGURE ONE}}
\hfill
\subbottom[Subfigure 2\label{sf:2}]{\fbox{SUBFIGURE TWO}}
\caption{Figure with two subfigures} \label{fig:twosubfig}
\end{figure}
```

`\tightsubcaptions` `\loosesubcaptions`

As with many other aspects of typesetting the style of subcaptions may be specified. There is a small amount of vertical space surrounding a subcaption. More space is used after the `\loosesubcaptions` declaration compared to that produced after the default `\tightsubcaptions` declaration.

`\subcaptionsize{`⟨*size*⟩`}`
`\subcaptionlabelfont{`⟨*fontspec*⟩`}`
`\subcaptionfont{`⟨*fontspec*⟩`}`

The size of the font used for subcaptions is specified by `\subcaptionsize`, and the fonts for the identifier and text are specified by `\subcaptionlabelfont` for the identifier and by `\subcaptionfont` for the title text. The defaults are:

```
\subcaptionsize{\footnotesize}
\subcaptionlabelfont{\normalfont}
\subcaptionfont{\normalfont}
```

`\subcaptionstyle{`⟨*style*⟩`}` `\raggedleft` `\centering` `\raggedright` `\centerlastline`

The identifier and title of a subcaption is typeset as a block (i.e., non-indented) paragraph by specifying

```
\subcaptionstyle{}
```

Other styles are available by calling `\subcaptionstyle` with a styling ⟨*cmd*⟩. Values that you might use are: `\centering` for a centered paragraph, `\raggedleft` or `\raggedright` for ragged left or right paragraphs, or `\centerlastline` which calls for a block paragraph with the last line centered.

`\hangsubcaption`
`\shortsubcaption`
`\normalsubcaption`

The `\hangsubcaption` declaration causes subcaptions to be typeset with the identifier above the title. Following the `\shortsubcaption` declaration subcaptions that are less than a full line in length are typeset left justified instead of centered. The `\normalsubcaption` declaration, which is the default, undoes any previous `\hangsubcaption` or `\shortsubcaption` declaration, so that subcaptions are normally centered.

10.10 SIDE CAPTIONS

Typically captions are put either above or below the element they are describing. Sometimes it is desireable to put a caption at the side of the element instead.

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

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

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

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

`\sidecapwidth` `\sidecapsep` `\setsidecaps{`⟨*sep*⟩`}{`⟨*width*⟩`}`

The caption is set in a box `\sidecapwidth` wide (the default is `\marginparwidth`) offset `\sidecapsep` (default `\marginparsep`) into the margin. The command `\setsidcaps` sets the `\sidecapsep` and `\sidecapwidth` to the given values. Changing the marginpar parameters, for example with `\setmarginnotes`, will not change the side caption settings. Note also that `\checkandfixthelayout` neither checks nor fixes the side caption parameters.

```
\sidecapmargin{⟨margin⟩}
\ifscapmargleft \scapmargleftfttrue \scapmargleftfalse
```

If the float is a single column float in a twocolumn document then the caption is always[4] placed in the adjacent margin, otherwise the `\sidecapmargin` command controls the margin where the sidecaption will be placed. The possible values for ⟨*margin*⟩ are one of: `left`, `right`, `inner`, or `outer`. If `left` or `right` is specified the caption will go into the left or right margin. If `inner` or `outer` is specified then in a two sided document the caption will be on different sides of the typeblock according to whether it is a recto or verso page; in a one sided document the caption margin is fixed. The left margin is the default.

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

```
\setsidecappos{⟨pos⟩}
```

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

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

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

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

The other kinds of simple captions can also be put at the side of a float. The positioning and styling commands for these are exactly those for `sidecaption`. Bilingual captions, which are not considered to be simple, can only be placed above or below a float; no facilities are provided for setting them at the side..

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

Sidecaptions may be continued with the `sidecontcaption` environment.

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

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

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

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

Caveat: Note that the `side...` envs expect the body of the float to be *taller* than the typeset caption/legend. In case you write a long caption/legend for a short float, you may want to visit this answer: http://tex.stackexchange.com/a/228412/3929.

[4] Well, nearly always. See the `\overridesidecapmargin` command later.

10.10.1 Tweaks

`\sidecapstyle`

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

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

You can change the command to suit your purposes; for example, uncommenting the `\captionnamefont` line would result in the caption's float name being set in a bold font.

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

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

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

```
\overridescapmargin{right}
```

to fix this.

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

`\sidecapfloatwidth{`⟨*length*⟩`}`

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

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

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

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

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

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

Table 10.6: Permitted arguments for some sidecaption related commands

\sidecapmargin	\overridescapmargin
left	left
right	right
inner	
outer	

Table 10.6 was created by the following code.

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

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

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

```
\sidecapmargin{left}
```

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

10.11 HOW LATEX MAKES CAPTIONS

This section provides an overview of how LaTeX creates captions and gives some examples of how to change the captioning style directly. The section need not be looked at more than once unless you like reading LaTeX code or you want to make changes to LaTeX's style of captioning not supported by the class.

The LaTeX kernel provides tools to help in the definition of captions, but it is the particular class that decides on their format.

`\caption[`⟨*short*⟩`]{`⟨*long*⟩`}`

The kernel (in `ltfloat.dtx`) defines the caption command via

```
\def\caption{%
    \refstepcounter\@captype \@dblarg{\@caption\@captype}}
```

`\@captype`

`\@captype` is defined by the code that creates a new float environment and is set to the environment's name (see the code for `\@xfloat` in `ltfloat.dtx`). For a `figure` environment, there is an equivalent to

```
\def\@captype{figure}
```

`\@caption{`⟨*type*⟩`}[`⟨*short*⟩`]{`⟨*long*⟩`}`

The kernel also provides the `\@caption` macro as:

```
\long\def\@caption#1[#2]#3{\par
  \addcontentsline{\csname ext@#1\endcsname}{#1}%     <-
    {\protect\numberline{\csname the#1\endcsname}%
     {\ignorespaces #2}}
  \begingroup
    \@parboxrestore
    \if@minipage
      \@setminipage
    \fi
    \normalsize
    \@makecaption{\csname fnum@#1\endcsname}%         <-
                 {\ignorespaces #3}\par
  \endgroup}
```

where ⟨*type*⟩ is the name of the environment in which the caption will be used. Putting these three commands together results in the user's view of the caption command as `\caption[`⟨*short*⟩`]{`⟨*long*⟩`}`.

It is the responsibilty of the class (or package) which defines floats to provide definitions for `\ext@type`, `\fnum@type` and `\@makecaption` which appear in the definition of `\@caption` (in the lines marked `<-` above).

`\ext@type`

This macro holds the name of the extension for a 'List of…' file. For example for the `figure` float environment there is the definition equivalent to

```
\newcommand{\ext@figure}{lof}
```

`\fnum@type`

This macro is responsible for typesetting the caption number. For example, for the `figure` environment there is the definition equivalent to

```
\newcommand{\fnum@figure}{\figurename~\thefigure}
```

`\@makecaption{`⟨*number*⟩`}{`⟨*text*⟩`}`

The `\@makecaption` macro, where ⟨*number*⟩ is a string such as 'Table 5.3' and ⟨*text*⟩ is the caption text, performs the typesetting of the caption, and is defined in the standard classes (in `classes.dtx`) as the equivalent of:

```
\newcommand{\@makecaption}[2]{
  \vskip\abovecaptionskip          <- 1
  \sbox\@tempboxa{#1: #2}          <- 2
  \ifdim \wd\@tempboxa >\hsize
    #1: #2\par                     <- 3
  \else
    \global \@minipagefalse
    \hb@xt@\hsize{\hfil\box\@tempboxa\hfil}
  \fi
  \vskip\belowcaptionskip}         <- 4
```

`\abovecaptionskip` `\belowcaptionskip`

Vertical space is added before and after a caption (lines marked 1 and 4 in the code for `\@makecaption` above) and the amount of space is given by the lengths `\abovecaptionskip` and `\belowcaptionskip`. The standard classes set these to 10pt and 0pt respectively. If you want to change the space before or after a caption, use `\setlength` to change the values. In figures, the caption is usually placed below the illustration. The actual space between the bottom of the illustration and the baseline of the first line of the caption is the `\abovecaptionskip` plus the `\parskip` plus the `\baselineskip`. If the illustration is in a `center` environment then additional space will be added by the `\end{center}`; it is usually better to use the `\centering` command rather than the `center` environment.

The actual typesetting of a caption is effectively performed by the code in lines marked 2 and 3 in the code for `\@makecaption`; note that these are where the colon that is typeset after the number is specified. If you want to make complex changes to the default captioning style you may have to create your own version of `\@caption` using `\renewcommand`. On the other hand, many such changes can be achieved by changing the definition of the the appropriate `\fnum@type` command(s). For example, to make the figure name and number bold:

```
\renewcommand{\fnum@figure}{\textbf{\figurename~\thefigure}}
```

REMEMBER: If you are doing anything involving commands that include the `@` character, and it's not in a class or package file, you have to do it within a `\makeatletter` and `\makeatother` pairing (see §E.4). So, if you modify the `\fnum@figure` command anywhere in your document it has to be done as:

```
\makeatletter
\renewcommand{\fnum@figure}{......}
```

A THOUSAND WORDS...

FIGURE 10.20: A picture is worth a thousand words

ANOTHER THOUSAND WORDS...

Figure 10.21 — A different kind of figure caption

`\makeatother`

As an example, Figure 10.20 was created by the following code:

```
\makeatletter
\renewcommand{\fnum@figure}{\textsc{\figurename~\thefigure}}
\makeatother
\begin{figure}
\centering
A THOUSAND WORDS\ldots
\caption{A picture is worth a thousand words}\label{fig:sc}
\end{figure}
```

As another example, suppose that you needed to typeset the `\figurename` and its number in a bold font, replace the colon that normally appears after the number by a long dash, and typeset the actual title text in a sans-serif font, as is illustrated by the caption for Figure 10.21. The following code does this.

```
\makeatletter
\renewcommand{\fnum@figure}[1]{\textbf{\figurename~\thefigure}
                                --- \sffamily}
\makeatother
\begin{figure}
 \centering
 ANOTHER THOUSAND WORDS\ldots
\caption{A different kind of figure caption}\label{fig:sf}
\end{figure}
```

Perhaps a little description of how this works is in order. Doing a little bit of TeX's macro processing by hand, the typesetting lines in `\@makecaption` (lines 2 and 3) get instantiated like:

```
\fnum@figure{\figurename~\thefigure}: text
```

Redefining `\fnum@figure` to take one argument and then not using the value of the argument essentially gobbles up the colon. Using

```
\textbf{\figurename~\thefigure}
```

in the definition causes `\figurename` and the number to be typeset in a bold font. After this comes the long dash. Finally, putting `\sffamily` at the end of the redefinition causes any following text (i.e., the actual title) to be typeset using the sans-serif font.

If you do modify `\@makecaption`, then spaces in the definition may be important; also you must use the comment (%) character in the same places as I have done above. Hopefully, though, the class provides the tools that you need to make most, if not all, of any likely caption styles.

10.12 Footnotes in captions

If you want to have a caption with a footnote, think long and hard as to whether this is really essential. It is not normally considered to be good typographic practice, and to rub the point in LaTeX does not make it necessarily easy to do. However, if you (or your publisher) insists, read on.

If it is present, the optional argument to `\caption` is put into the 'List of...' as appropriate. If the argument is not present, then the text of the required argument is put into the 'List of...'. In the first case, the optional argument is moving, and in the second case the required argument is moving. The `\footnote` command is fragile and must be `\protected` (i.e., `\protect\footnote{}`) if it is used in a moving argument. If you don't want the footnote to appear in the 'List of...', use a footnoteless optional argument and a footnoted required argument.

You will probably be surprised if you just do, for example:

```
\begin{figure}
...
\caption[For LoF]{For figure\footnote{The footnote}}
\end{figure}
```

because (a) the footnote number may be greater than you thought, and (b) the footnote text has vanished. This latter is because LaTeX won't typeset footnotes from a float. To get an actual footnote within the float you have to use a minipage, like:

```
\begin{figure}
\begin{minipage}{\linewidth}
...
\caption[For LoF]{For figure\footnote{The footnote}}
\end{minipage}
\end{figure}
```

If you are using the standard classes you may now find that you get two footnotes for the price of one. Fortunately this will not occur with this class, nor will an unexpected increase of the footnote number.

When using a minipage as above, the footnote text is typeset at the bottom of the minipage (i.e., within the float). If you want the footnote text typeset at the bottom of the page, then you have to use the `\footnotemark` and `\footnotetext` commands like:

```
\begin{figure}
...
\caption[For LoF]{For figure\footnotemark}
\end{figure}
```

```
\footnotetext{The footnote}
```

This will typeset the argument of the `\footnotetext` command at the bottom of the page where you called the command. Of course, the figure might have floated to a later page, and then it's a matter of some manual fiddling to get everything on the same page, and possibly to get the footnote marks to match correctly with the footnote text.

At this point, you are on your own.

10.13 THE CLASS VERSUS THE CAPTION PACKAGE (AND ITS FRIENDS)

For some, the configurations for captions provided by the class, are either a bit too complicated or not complicated enough.

The caption package by Alex Sommerfeldt may provide a simpler and much more extensive configuration interface for captions. The package can be used with the class, but there are a few caveats:

(a) All of the special configuration macros provided by the class will no longer have any effect (caption overwrites the core, and thus our interfaces will have no effect),

(b) `\abovecaptionskip` will be reset to 10 pt, and `\belowcaptionskip` to zero. (The class would set both to 0.5`\onelineskip`, so if you need to change these, move the change until after caption has been loaded)

Eleven

Rows and columns

The class provides extensions to the standard `array` and `tabular` environments. These are based partly on a merging of the array [MC18], dcolumn [Car14], delarray [Car14], tabularx [Car16], and booktabs [Fea16] packages. Much of the material in this chapter strongly reflects the documentation of these packages.

Note. *As of September 2018: The array, delarray, tabularx and dcolumn packages are no longer embedded into the class, but rather being autoloaded from the LaTeX installation.*[1] *As the embededed versions were just carbon copies,*[2] *we get the same result but just loading the packages, with less maintenance. Plus these packages are part of the LaTeX core packages and thus is available in all LaTeX installations.*

We have kept the documentation we had written for the manual (and updated it slightly), but refer to [MC18], [Car14], [Car16] and [Car14] for the 100% up to date documentation.

Additionally, new kinds of tabular environments are also provided.

11.1 General

`\[ \begin{array}[`⟨*pos*⟩`]{`⟨*preamble*⟩`}` rows `\end{array} \]`
`\begin{tabular}[`⟨*pos*⟩`]{`⟨*preamble*⟩`}` rows `\end{tabular}`
`\begin{tabular*}{`⟨*width*⟩`}[`⟨*pos*⟩`]{`⟨*preamble*⟩`}` rows `\end{tabular*}`
`\begin{tabularx}{`⟨*width*⟩`}[`⟨*pos*⟩`]{`⟨*preamble*⟩`}` rows `\end{tabularx}`

The `array` and `tabular` environments are traditional and the others are extensions. The `array` is for typesetting math and has to be within a math environment of some kind. The `tabular` series are for typesetting ordinary text.

The optional ⟨*pos*⟩ argument can be one of `t`, `c`, or `b` (the default is `c`), and controls the vertical position of the array or tabular; either the `top` or the `center`, or the `bottom` is aligned with the baseline. Each row consists of elements separated by `&`, and finished with `\\`. There may be as many rows as desired. The number and style of the columns is specified by the ⟨*preamble*⟩. The width of each column is wide enough to contain its longest entry and the overall width of the `array` or `tabular` is sufficient to contain all the columns. However, the `tabular*` and `tabularx` environments provide more control over the width through their ⟨*width*⟩ argument.

[1] As the embedded versions were more or less carbon copies, it makes much more sense to let the LaTeX-team maintain them, than us having to replace the embedded copy each time they are updated or bug fixed.

[2] With edited error messages.

Table 11.1: The array and tabular preamble options.

`l`	Left adjusted column.
`c`	Centered adjusted column.
`r`	Right adjusted column.
`p{`⟨*width*⟩`}`	Equivalent to `\parbox[t]{`⟨*width*⟩`}`.
`m{`⟨*width*⟩`}`	Defines a column of width ⟨*width*⟩. Every entry will be centered in proportion to the rest of the line. It is somewhat like `\parbox{`⟨*width*⟩`}`.
`b{`⟨*width*⟩`}`	Coincides with `\parbox[b]{`⟨*width*⟩`}`.
`>{`⟨*decl*⟩`}`	Can be used before an `l`, `r`, `c`, `p`, `m` or a `b` option. It inserts ⟨*decl*⟩ directly in front of the entry of the column.
`<{`⟨*decl*⟩`}`	Can be used after an `l`, `r`, `c`, `p{..}`, `m{..}` or a `b{..}` option. It inserts ⟨*decl*⟩ right after the entry of the column.
`|`	Inserts a vertical line. The distance between two columns will be enlarged by the width of the line.
`@{`⟨*decl*⟩`}`	Suppresses inter-column space and inserts ⟨*decl*⟩ instead.
`!{`⟨*decl*⟩`}`	Can be used anywhere and corresponds with the `|` option. The difference is that ⟨*decl*⟩ is inserted instead of a vertical line, so this option doesn't suppress the normally inserted space between columns in contrast to `@{...}`.
`*{`⟨*num*⟩`}{`⟨*opts*⟩`}`	Equivalent to ⟨*num*⟩ copies of ⟨*opts*⟩
`w{`⟨*align*⟩`}{`⟨*width*⟩`}`	*New in* array *v2.4h, 2018.* Here ⟨*align*⟩ is one of `l`, `c` or `r`. The construction corresponds to every cell in the column being formated as `\makebox[`⟨*width*⟩`][`⟨*align*⟩`]{`⟨*cell*⟩`}`. It will silently overprint if the contents are wider than ⟨*width*⟩.
`W{`⟨*align*⟩`}{`⟨*width*⟩`}`	*New in* array *v2.4h, 2018.* Similar to `w`, but issues an overfull warning if the contents is too wide.
`D{`⟨*ssep*⟩`}{`⟨*osep*⟩`}{`⟨*places*⟩`}`	Column entries aligned on a 'decimal point'

11.2 THE PREAMBLE

You use the ⟨*preamble*⟩ argument to the array and tabular environments to specify the number of columns and how you want column entries to appear. The preamble consists of a sequence of options, which are listed in Table 11.1.

Examples of the options include:

- A flush left column with bold font can be specified with `>{\bfseries}l`.

```
\begin{center}
\begin{tabular}{>{\large}c >{\large\bfseries}l >{\large\itshape}c}
                  \toprule
A   & B  & C \\
100 & 10 & 1 \\ \bottomrule
\end{tabular}
```

```
\end{center}
```

A	**B**	*C*
100	**10**	*1*

- In columns which have been generated with `p`, `m` or `b`, the default value of `\parindent` is 0pt.

```
\begin{center}
\begin{tabular}{m{1cm}m{1cm}m{1cm}} \toprule
1 1 1 1 1 1 1 1 1 1 1 1 &
2 2 2 2 2 2 2 2         &
3 3 3 3                 \\ \bottomrule
\end{tabular}
\end{center}
```

1 1 1 1 1 1 1 1 1 1 1 1	2 2 2 2 2 2 2 2	3 3 3 3

The `\parindent` for a particular column can be changed with, for example

```
>{\setlength{\parindent}{1cm}}p
```

```
\begin{center}
\begin{tabular}{>{\setlength{\parindent}{5mm}}p{2cm} p{2cm}} \toprule
1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 &
1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 \\ \bottomrule
\end{tabular}
\end{center}
```

1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0	1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0

- The specification `>{$}c<{$}` generates a column in math mode in a `tabular` environment. When used in an `array` environment the column is in LR mode (because the additional $'s cancel the existing $'s).
- Using `c!{\hspace{1cm}}c` you get space between two columns which is enlarged by one centimeter, while `c@\hspace{1cm}}c` gives you exactly one centimeter space between two columns.
- Elsewhere reasons are given why you should not use vertical lines (e.g., the `|` option) in tables. Any examples that use vertical lines are for illustrative purposes only where it is advantageous to denote column boundaries, for example to show different spacing effects.

11.2.1 D column specifiers

Recommended alternative

As an alternative to the D column (through using the dcolumn package), you can use the siunitx package which have the added bonus of many more configuration and formatting features. See [Wri19] for details.

In financial tables dealing with pounds and pence or dollars and cents, column entries should be aligned on the separator between the numbers. The D column specifier is provided for columns which are to be aligned on a 'decimal point'. The specifier takes three arguments.

`D{`⟨*ssep*⟩`}{`⟨*osep*⟩`}{`⟨*places*⟩`}`

⟨*ssep*⟩ is the single character which is used as the separator in the source `.tex` file. Thus it will usually be '`.`' or '`,`'.

⟨*osep*⟩ is the separator in the output, this may be the same as the first argument, but may be any math-mode expression, such as `\cdot`. A D column always uses math mode for the digits as well as the separator.

⟨*places*⟩ should be the maximum number of decimal places in the column (but see below for more on this). If this is negative, any number of decimal places can be used in the column, and all entries will be centred on (the leading edge of) the separator. Note that this can cause a column to be too wide; for instance, compare the first two columns in the example below.

Here are some example specifications which, for convenience, employ the `\newcolumntype` macro described later.

```
\newcolumntype{d}[1]{D{.}{\cdot}{#1}}
```

This defines `d` to be a column specifier taking a single argument specifying the number of decimal places, and the `.tex` file should use '`.`' as the separator, with `\cdot` (·) being used in the output.

```
\newcolumntype{.}{D{.}{.}{-1}}
```

The result of this is that '`.`' specifies a column of entries to be centered on the '.'.

```
\newcolumntype{,}{D{,}{,}{2}}
```

And the result of this is that '`,`' specifies a column of entries with at most two decimal places after a ','.

The following table is typeset from this code:

```
\begin{center}
 \begin{tabular}{|d{-1}|d{2}|.|,|}
 1.2   & 1.2   &1.2    &1,2    \\
 1.23  & 1.23  &12.5   &300,2  \\
 1121.2& 1121.2&861.20 &674,29 \\
 184   & 184   &10     &69     \\
 .4    & .4    &       &,4     \\
       &       &.4     &
 \end{tabular}
 \end{center}
```

1·2	1·2	1.2	1,2
1·23	1·23	12.5	300,2
1121·2	1121·2	861.20	674,29
184	184	10	69
·4	·4		,4
		.4	

Note that the first column, which had a negative ⟨*places*⟩ argument is wider than the second column, so that the decimal point appears in the middle of the column.

The third ⟨*places*⟩ argument may specify *both* the number of digits to the left and to the right of the decimal place. The third column in the next table below is set with `D{.}{.}{5.1}` and in the second table, `D{.}{.}{1.1}`, to specify 'five places to the left and one to the right' and 'one place to the left and one to the right' respectively. (You may use ',' or other characters, not necessarily '.' in this argument.) The column of figures is then positioned such that a number with the specified numbers of digits is centred in the column.

Be careful if you have table headings inserted, say, with

```
\multicolumn{1}{c}{...}
```

to over-ride the `D` column type, as the overall result may not look quite as good as you might expect. In the next pair of tabulars the first column is set with

```
D{.}{.}{-1}
```

to produce a column centered on the '.', and the second column is set with

```
D{.}{.}{1}
```

to produce a right aligned column.

Source for example 11.1

```
\begin{center}\small
\begin{tabular}[t]{|D..{-1}|D..{1}|D..{5.1}|}
\multicolumn{1}{|c|}{head} &
\multicolumn{1}{c|}{head}  &
\multicolumn{1}{c|}{head}    \\[3pt]
1       & 2       & 3        \\
1.2     & 1.2     & 1.2      \\
11212.2 & 11212.2 & 11212.2  \\
.4      & .4      & .4
\end{tabular}
\hfill
\begin{tabular}[t]{|D..{-1}|D..{1}|D..{1.1}|}
\multicolumn{1}{|c|}{wide heading} &
\multicolumn{1}{c|}{wide heading} &
\multicolumn{1}{c|}{wide heading} \\[3pt]
1       & 2       & 3              \\
1.2     & 1.2     & 1.2            \\
```

Typeset example 11.1: Tabular with narrow and wide headings

head	head	head		wide heading	wide heading	wide heading
1	2	3		1	2	3
1.2	1.2	1.2		1.2	1.2	1.2
11212.2	11212.2	11212.2		.4	.4	.4
.4	.4	.4				

```
.4      & .4      & .4
\end{tabular}
\end{center}
```

In both of these tables the first column is set with `D{.}{.}{-1}` to produce a column centered on the '.', and the second column is set with `D{.}{.}{1}` to produce a right aligned column.

The centered (first) column produces columns that are wider than necessary to fit in the numbers under a heading as it has to ensure that the decimal point is centered. The right aligned (second) column does not have this drawback, but under a wide heading a column of small right aligned figures is somewhat disconcerting.

The notation for the ⟨*places*⟩ argument also enables columns that are centred on the mid-point of the separator, rather than its leading edge; for example

```
D{+}{\,\pm\,}{3,3}
```

will give a symmetric layout of up to three digits on either side of a ± sign.

11.2.2 Defining new column specifiers

You can easily type

`>{`⟨*some declarations*⟩`}{c}<{`⟨*some more declarations*⟩`}`

when you have a one-off column in a table, but it gets tedious if you often use columns of this form. The `\newcolumntype` lets you define a new column option like, say

`\newcolumntype{x}{>{`⟨*some declarations*⟩`}{c}<{`⟨*some more declarations*⟩`}}`

and you can then use the `x` column specifier in the preamble wherever you want a column of this kind.

`\newcolumntype{`⟨*char*⟩`}[`⟨*nargs*⟩`]{`⟨*spec*⟩`}`

The ⟨*char*⟩ argument is the character that identifies the option and ⟨*spec*⟩ is its specification in terms of the regular preamble options. The `\newcolumntype` command is similar to `\newcommand` — ⟨*spec*⟩ itself can take arguments with the optional ⟨*nargs*⟩ argument declaring their number.

For example, it is commonly required to have both math-mode and text columns in the same alignment. Defining:

```
\newcolumntype{C}{>{$}c<{$}}
\newcolumntype{L}{>{$}l<{$}}
```

```
\newcolumntype{R}{>{$}r<{$}}
```

Then `C` can be used to get centred text in an `array`, or centred math-mode in a `tabular`. Similarly `L` and `R` are for left- and right-aligned columns.

The ⟨*spec*⟩ in a `\newcolumntype` command may refer to any of the primitive column specifiers (see table 11.1 on page 212), or to any new letters defined in other `\newcolumntype` commands.

`\showcols`

A list of all the currently active `\newcolumntype` definitions is sent to the terminal and log file if the `\showcols` command is given.

11.2.3 Surprises

- A preamble of the form `{wx*{0}{abc}yz}` is treated as `{wxyz}`
- An incorrect positional argument, such as `[Q]`, is treated as `[t]`.
- A preamble such as `{cc*{2}}` with an error in a $*$-form will generate an error message that is not particularly helpful.
- Error messages generated when parsing the column specification refer to the preamble argument *after* it has been re-written by the `\newcolumntype` system, not to the preamble entered by the user.
- Repeated `<` or `>` constructions are allowed. `>{`⟨*decs1*⟩`}>{`⟨*decs2*⟩`}` is treated the same as `>{`⟨*decs2*⟩`}{`⟨*decs1*⟩`}`.

  The treatment of multiple `<` or `>` declarations may seem strange. Using the obvious ordering of `>{`⟨*decs1*⟩`}{`⟨*decs2*⟩`}` has the disadvantage of not allowing the settings of a `\newcolumntype` defined using these declarations to be overriden.
- The `\extracolsep` command may be used in `@`-expressions as in standard LaTeX, and also in `!`-expressions.

  The use of `\extracolsep` is subject to the following two restrictions. There must be at most one `\extracolsep` command per `@`, or `!` expression and the command must be directly entered into the `@` expression, not as part of a macro definition. Thus

  ```
  \newcommand{\ef}{\extracolsep{\fill}} ... @{\ef}
  ```

  does not work. However you can use something like

  ```
  \newcolumntype{e}{@{\extracolsep{\fill}}
  ```

  instead.
- As noted by the LaTeX book [Lam94], a `\multicolumn`, with the exception of the first column, consists of the entry and the *following* inter-column material. This means that in a tabular with the preamble `|l|l|l|l|` input such as `\multicolumn{2}{|c|}` in anything other than the first column is incorrect.

  In the standard array/tabular implementation this error is not noticeable as a `|` takes no horizontal space. But in the class the vertical lines take up their natural width and you will see two lines if two are specified — another reason to avoid using `|`.

11.3 THE ARRAY ENVIRONMENT

Mathematical arrays are usually produced using the `array` environment.

`\[ \begin{array}[`⟨*pos*⟩`]{`⟨*preamble*⟩`} rows \end{array} \]`
`\[ \begin{array}[`⟨*pos*⟩`]`⟨*left*⟩`{`⟨*preamble*⟩`}`⟨*right*⟩ rows `\end{array} \]`

Math formula are usually centered in the columns, but a column of numbers often looks best flush right, or aligned on some distinctive feature. In the latter case the `D` column scheme is very handy.

```
\[ \begin{array}{lcr}
   a +b +c & d - e - f & 123 \\
   g-h     &  j k      & 45 \\
    l      &   m       & 6
  \end{array} \]
```

$$\begin{array}{lcr} a+b+c & d-e-f & 123 \\ g-h & jk & 45 \\ l & m & 6 \end{array}$$

Arrays are often enclosed in brackets or vertical lines or brackets or other symbols to denote math constructs like matrices. The delimeters are often large and have to be indicated using `\left` and `\right` commands.

```
\[ \left[ \begin{array}{cc}
          x_{1} & x_{2} \\
          x_{3} & x_{4}
          \end{array} \right] \]
```

$$\left[\begin{array}{cc} x_1 & x_2 \\ x_3 & x_4 \end{array} \right]$$

The class's `array` environment is an extension of the standard environment in that it provides a system of implicit `\left` `\right` pairs. If you want an array surrounded by parentheses, you can enter:

```
\[  \begin{array}({cc})a&b\\c&d\end{array}   \]
```

$$\begin{array}({cc})a&b\\c&d\end{array}$$

Or, a litle more complex

```
\[ \begin{array}({c})
     \begin{array}|{cc}|
       x_{1} & x_{2} \\
       x_{3} & x_{4}
     \end{array} \\
        y \\
        z
   \end{array} \]
```

$$\left(\begin{array}{|cc|} x_1 & x_2 \\ x_3 & x_4 \end{array} \right)$$
$$\begin{array}{c} y \\ z \end{array}$$

And you can do things like this:

```
\[ a = {\begin{array}|{*{20}{c}}|
        x-\lambda & 1         & 0 \\
        0         & x-\lambda & 1 \\
        0         &           & x-\lambda \\
        \end{array}
       }^{2} \]
```

$$a = \begin{vmatrix} x-\lambda & 1 & 0 \\ 0 & x-\lambda & 1 \\ 0 & & x-\lambda \end{vmatrix}^2$$

As another example, a construct equivalent to plain TeX's `\cases` could be defined by:

```
\[  f(x)=\begin{array}\{{lL}.
          0         & if $x=0$\\
          \sin(x)/x & otherwise
          \end{array}  \]
```

$$f(x) = \begin{cases} 0 & \text{if } x = 0 \\ \sin(x)/x & \text{otherwise} \end{cases}$$

Here L denotes a column of left aligned L-R text, as described earlier. Note that as the delimiters must always be used in pairs, the '`.`' must be used to denote a 'null delimiter'.

This feature is especially useful if the `[t]` or `[b]` arguments are also used. In these cases the result is not equivalent to surrounding the environment by `\left`...`\right`, as can be seen from the following example:

```
\begin{array}[t]({c}) 1\\2\\3 \end{array}
\begin{array}[c]({c}) 1\\2\\3 \end{array}
\begin{array}[b]({c}) 1\\2\\3 \end{array}
\quad\mbox{not}\quad
\left(\begin{array}[t]{c} 1\\2\\3 \end{array}\right)
\left(\begin{array}[c]{c} 1\\2\\3 \end{array}\right)
\left(\begin{array}[b]{c} 1\\2\\3 \end{array}\right)
```

$$\begin{array}{c}1\\2\\3\end{array} \quad \begin{array}{c}1\\2\\3\end{array} \quad \begin{array}{c}1\\2\\3\end{array} \quad \text{not} \quad \left(\begin{array}{c}\\1\\2\\3\end{array}\right) \left(\begin{array}{c}1\\2\\3\end{array}\right) \left(\begin{array}{c}1\\2\\3\\\end{array}\right)$$

Table 11.2: Demonstrating the parts of a table

		spanner		
stub	head subhead	head subhead	head	head
A	a	b	c	d
B	e	f	g	h
	cut-in head			
C	i	j	k	l
D	m	n	o	p

11.4 TABLES

A table is one way of presenting a large amount of information in a limited space. Even a simple table can presents facts that could require several wordy paragraphs — it is not only a picture that is worth a thousand words.

A table should have at least two columns, otherwise it is really a list, and many times has more. The leftmost column is often called the *stub* and it typically contains a vertical listing of the information categories in the other columns. The columns have *heads* (or *headings*) at the top indicating the nature of the entries in the column, although it is not always necessary to provide a head for the stub if the heading is obvious from the table's caption. Column heads may include subheadings, often to specify the unit of measurement for numeric data.

A less simple table may need two or more levels of headings, in which case *decked heads* are used. A decked head consists of a *spanner head* and the two or more column heads it applies to. A horizontal *spanner rule* is set between the spanner and column heads to show which columns belong to the spanner.

Double decking, and certainly triple decking or more, should be avoided as it can make it difficult following them down the table. It may be possible to use a *cut-in head* instead of double decking. A cut-in head is one that cuts across the columns of the table and applies to all the matter below it. To try and clarify, the parts of a table are shown diagrammatically in Table 11.2.

No mention has been made of vertical rules in a table, and this is deliberate. There should be no vertical rules in a table. Rules, if used at all, should be horizontal only, and these should be single, not double or triple. It's not that ink is expensive, or that practically no typesetting is done by hand any more, it is simply that the visual clutter should be eliminated.

For example, in Table 11.3 which was produced from the code below, Table 11.3() is from the LaTeX book and Table 11.3(a) is how Simon Fear [Fea16] suggests it should be cleaned up. Notice how both the revised code and the table are generally simpler than the originals.

```
\begin{table}
\centering
\caption{Two views of one table} \label{tab:twoviews}
```

Table 11.3: Two views of one table

(a) Before

gnats	gram	$13.65
	each	.01
gnu	stuffed	92.50
emu		33.33
armadillo	frozen	8.99

(b) After

Item		
Animal	Description	Price ($)
Gnat	per gram	13.65
	each	0.01
Gnu	stuffed	92.50
Emu	stuffed	33.33
Armadillo	frozen	8.99

```
\subtop[Before]{\label{tab:before}%
 \begin{tabular}{||l|lr||} \hline
 gnats     & gram      & \$13.65 \\ \cline{2-3}
           & each      & .01 \\ \hline
 gnu       & stuffed   & 92.50 \\ \cline{1-1} \cline{3-3}
 emu       &           & 33.33 \\ \hline
 armadillo & frozen    & 8.99 \\ \hline
 \end{tabular}}
\hfill
\subtop[After]{\label{tab:after}%
 \begin{tabular}{@{}llr@{}} \toprule
 \multicolumn{2}{c}{Item} \\ \cmidrule(r){1-2}
 Animal & Description & Price (\$)\\ \midrule
 Gnat   & per gram    & 13.65 \\
        & each        & 0.01 \\
 Gnu    & stuffed     & 92.50 \\
 Emu    & stuffed     & 33.33 \\
 Armadillo & frozen   & 8.99 \\ \bottomrule
 \end{tabular}
}
\end{table}
```

Columns of numbers often end with a line giving the total or result. A horizontal rule may be drawn to separate the result from the rest but a small amount of white space may do just as well, as in Table 11.4.

Some other points are:

- Put the units in the column heading (not in the body of the table).
- Always precede a decimal point by a digit; thus 0.1 *not* just .1.
- Do not use 'ditto' signs or any other such convention to repeat a previous value. In many circumstances a blank will serve just as well. If it won't, then repeat the value.

Not every table requires all the elements mentioned above. For instance, in Charles Dicken's *David Copperfield* (1849–1850) Mr Wilkins Micawber states:

Table 11.4: Micawber's law

Income	£20-0-0	£20-0-0
Expenditure	£19-0-6	£20-0-6
Result	happiness	misery

Table 11.5: A narrow table split half and half

Relative contents of odd isotopes for heavy elements

Element	Z	γ	Element	Z	γ
Sm	62	1.480	W	74	0.505
Gd	64	0.691	Os	76	0.811
Dy	66	0.930	Pt	78	1.160
...			...		

> 'Annual income twenty pounds, annual expenditure nineteen six, result happiness. Annual income twenty pounds, annual expenditure twenty pounds ought and six, result misery.'

This can also be represented in tabular form[3] as in Table 11.4.

Long narrow tables do not look well on the page. In such cases the table could be set half and half instead, as in Table 11.5.

11.5 FEAR'S RULES

Simon Fear disapproves of the default LaTeX table rules and wrote the booktabs package [Fea16] to provide better horizontal rules. Like many typographers, he abhors vertical rules. The following is taken almost verbatim from the booktabs package.

In the simplest of cases a table begins with a top rule, has a single row of column headings, then a dividing rule, and after the columns of data it is finished off with a final rule. The top and bottom rules are normally set heavier (i.e., thicker or darker) than any intermediate rules.

```
\toprule[⟨width⟩] \bottomrule[⟨width⟩] \heavyrulewidth
\midrule[⟨width⟩] \lightrulewidth
\aboverulesep \belowrulesep \doublerulesep
```

All the rule commands here go immediately after the closing `\\` of the preceding row (except of course `\toprule`, which comes right after the start of the environment). Each rule has an optional length argument, ⟨*width*⟩, which you can use to locally change the default width of the rule.

`\toprule` draws a rule (with a default width of `\heavyrulewidth`), and `\belowrulesep` vertical space inserted below it.

`\midrule` draws a rule (default `\lightrulewidth` width) with `\aboverulesep` space above it and `\belowrulesep` below it.

[3] But putting Josh Billings' (Henry Wheeler Shaw) corollary: 'Live within your income, even if you have to borrow to do it.' into tabular form would not work.

`\bottomrule` draws a rule with a default width of `\heavyrulewidth`. There is `\aboverulesep` space above it and `\belowrulesep` space below it.

If a rule immediately follows another the space between them is `\doublerulesep`, but as you are not going to use double rules you won't be concerned about this.

`\cmidrule[`⟨*width*⟩`](`⟨*trim*⟩`){`⟨*m–n*⟩`}`
`\cmidrulewidth` `\cmidrulekern`

Spanner rules do not extend the full width of the table, and the `\cmidrule` is provided for that purpose. This draws a rule, default thickness `\cmidrulewidth`, across columns ⟨*m*⟩ to ⟨*n*⟩ inclusive (where ⟨*m*⟩ and ⟨*n*⟩ are column numbers, with ⟨*m*⟩ not greater than ⟨*n*⟩).

Generally, this rule should not come to the full width of the end columns, and this is especially the case if you have to begin a `\cmidrule` straight after the end of another one. You can use the optional trimming argument commands, which are `(r)`, `(l)` and `(rl)` or `(lr)`, indicated whether the right and/or left ends of the rule should be trimmed. Note the exceptional use of parentheses instead of brackets for this optional argument.

`\cmidrule` draws a rule from column m to n with a default thickness of `\cmidrulewidth`. Adjacent `\cmidrule`s, for example

```
... \\ \cmidrule{2-3}\cmidrule{5-7}
```

have the same vertical alignment. It is best not to leave any space between these commands. The space above and below is normally `\aboverulesep` and `\belowrulesep`.

If you are forced into having double spanner rules then you will reluctantly have to insert the command `\morecmidrules` between the commands for the upper and lower rules. For example:

```
... \\ \cmidrule{2-3}\cmidrule{5-7}\morecmidrules\cmidrule{2-3}
```

will draw double rules across columns 2 and 3. You must finish off the rules for one row before starting the lower set of rules. There must not be any space surrounding the `\morecmidrules` macro. The upper and lower rules are separated by `\cmidrulesep`.

`\addlinespace[`⟨*width*⟩`]` `\defaultaddspace`

Occasionally extra space between certain rows of a table may be helpful; for example, before the last row if this is a total. This is simply a matter of inserting `\addlinespace` after the `\\` alignment marker. You can think of `\addlinespace` as being a white rule of width ⟨*width*⟩. The default space is `\defaultaddspace` which gives rather less than a whole line space. If another rule follows the amount of whitespace is increased by `\doublerulesep`.

`\specialrule{`⟨*width*⟩`}{`⟨*abovespace*⟩`}{`⟨*belowspace*⟩`}`

You can, but should not, generate peculiar spacing between rules by using `\specialrule`. The three required arguments are the width, ⟨*width*⟩, of the rule and the spaces above (⟨*abovespace*⟩) and below (⟨*belowspace*⟩). `\specialrule` ignores a preceding rule but if there is a following one then ⟨*belowspace*⟩ will be increased by `\doublerulesep`.

The default dimensions are

`\heavyrulewidth` = 0.08em
`\lightrulewidth` = 0.05em
`\cmidrulewidth` = 0.03em
`\belowrulesep` = 0.65ex

\aboverulesep = 0.4ex
\defaultaddspace = 0.5em
\cmidrulekern = 0.25em

The last of these, \cmidrulekern, is the amount by which a \cmidrule is trimmed at the ends indicated in the () option. In the construction

```
\cmidrule(r){1-2}\cmidrule(l){3-4}
```

there is a total of 0.5em separating the two rules. Currently the only way to get special effects is to reset \cmidrulekern as appropriate; the amount of trimming is not available as an argument in the current implementation of \cmidrule.

An example of the commands in use was given by the code to produce Table 11.3(a) on page 221:

```
\begin{tabular}{@{}llr@{}} \toprule
\multicolumn{2}{c}{Item} \\ \cmidrule(r){1-2}
Animal & Description & Price (\$)\\ \midrule
Gnat   & per gram  & 13.65 \\
       & each      & 0.01 \\
Gnu    & stuffed   & 92.50 \\
Emu    & stuffed   & 33.33 \\
Armadillo & frozen & 8.99 \\ \bottomrule
\end{tabular}
```

11.5.1 Fills

The rules described previously go between rows. Sometimes it may be desirable to to put a rule or something similar within a row.

\downbracefill \hrulefill \upbracefill

Examples of \downbracefill, \hrulefill, and \upbracefill are illustrated in Table 11.6, typeset from the code below. Surprisingly these are ordinary text commands, not math mode commands.

```
\begin{table}
\centering
\caption{Example table with fills} \label{tab:fills}
\begin{tabular}{rrrrr}
1 & 2   & 3   & 4    & 5 \\
Q &     & fgh & jklm & qwerty \\
v & as  & x   &    y &      z \\
g & nnn & \multicolumn{3}{c}{\upbracefill} \\
\multicolumn{3}{c}{\downbracefill} & pq & dgh \\
k & j   & t   & co   & ytrewq \\
1 & 2   & 3   & \multicolumn{2}{c}{\hrulefill}
\end{tabular}
\end{table}
```

Here are the same fills, but this time in an array environment. are shown afterwards. Notice the $s in the array surrounding the fills. Normally $...$ is used to typeset a small amount of math mode material in the middle of text. In this case, as the array is already

Table 11.6: Example table with fills

1	2	3	4	5
Q		fgh	jklm	qwerty
v	as	x	y	z
g	nnn	$\underbrace{\qquad\qquad}$	pq	dgh
$\overbrace{\qquad\qquad}$			co	ytrewq
k	j	t		
1	2	3	$\underline{\qquad\qquad}$	

in math mode the `$...$` are used to typeset a small amount of text material within math mode.

```
\begin{displaymath}
\begin{array}{rrrrr}
1 & 2   & 3   & 4    & 5 \\
Q &     & fgh & jklm & qwerty \\
v & as  & x   &    y &      z \\
g & nnn & \multicolumn{3}{c}{$\upbracefill$} \\
\multicolumn{3}{c}{$\downbracefill$} & pq & dgh \\
k & j   & t   & co   & ytrewq \\
1 & 2   & 3   & \multicolumn{2}{c}{\hrulefill}
\end{array}
\end{displaymath}
```

1	2	3	4	5
Q		fgh	$jklm$	$qwerty$
v	as	x	y	z
g	nnn	$\underbrace{\qquad\qquad}$	pq	dgh
$\overbrace{\qquad\qquad}$			co	$ytrewq$
k	j	t		
1	2	3	$\underline{\qquad\qquad}$	

You can define your own 'fill'. For example:

```
\newcommand*{\upbracketfill}{%
  \vrule height 4pt depth 0pt\hrulefill%
  \vrule height 4pt depth 0pt}
```

is a fill that has the appearance of a (horizontal) bracket. It can be used like this:

```
\begin{displaymath}
\begin{array}{cccc}
1 & 2 & 3 & 4 \\
a & \multicolumn{2}{c}{\upbracketfill} & d \\
A & B & C & D
\end{array}
\end{displaymath}
```

Figure 11.1: Example of a regular `tabular`

Multicolumn entry!		THREE	FOUR
one	The width of this column is fixed (5.5pc).	three	Column four will act in the same way as column two, with the same width.

$$\begin{array}{cccc} 1 & 2 & 3 & 4 \\ a & \llcorner\!\!\!\!\!\!\lrcorner & & d \\ A & B & C & D \end{array}$$

11.6 TABULAR ENVIRONMENTS

`\begin{tabular}[`⟨*pos*⟩`]{`⟨*format*⟩`}` rows `\end{tabular}`
`\begin{tabular*}{`⟨*width*⟩`}[`⟨*pos*⟩`]{`⟨*format*⟩`}` rows `\end{tabular*}`
`\begin{tabularx}{`⟨*width*⟩`}[`⟨*pos*⟩`]{`⟨*format*⟩`}` rows `\end{tabularx}`

A table created using the `tabular` environment comes out as wide as it has to be to accomodate the entries. On the other hand, both the `tabular*` and `tabularx` environments let you specify the overall width of the table via the additional ⟨*width*⟩ atrgument.

The `tabular*` environment makes any necessary adjustment by altering the intercolumn spaces while the `tabularx` environment alters the column widths. Those columns that can be adjusted are noted by using the letter `X` as the column specifier in the ⟨*format*⟩. Once the correct column widths have been calculated the `X` columns are converted to `p` columns.

11.6.1 Examples

The following code is used for a regular `tabular`.

```
\begin{figure}
\centering
\caption{Example of a regular \texttt{tabular}}
\begin{tabular}{|c|p{5.5pc}|c|p{5.5pc}|}  \hline
\multicolumn{2}{|c|}{Multicolumn entry!} & THREE & FOUR \\ \hline
one &
\raggedright\arraybackslash The width of this column is fixed
(5.5pc). & three &
\raggedright\arraybackslash Column four will act in the same
way as column two, with the same width.\\
\hline
\end{tabular}
\end{figure}
```

Figure 11.2: Example `tabularx` and `tabular*` with widths of 250pt

```
\begin{tabularx}{250pt}{|c|X|c|X|}
```

Multicolumn entry!		THREE	FOUR
one	The width of this column depends on the width of the table.[4]	three	Column four will act in the same way as column two, with the same width.

```
\begin{tabular*}{250pt}{|c|p{5.5pc}|c|p{5.5pc}|}
```

Multicolumn entry!		THREE	FOUR	
one	The width of this column is fixed (5.5pc).	three	Column four will act in the same way as column two, with the same width.	

The following examples use virtually the same contents, the major differences are the specifications of the environment.

Note that the horizontal rules extend beyond the last column. There are no `X` columns and the total width required to set the `tabular*` is less than the 250pt specified for the width.

Compare the previous narrow `tabular*` with the next one which is set with

```
\begin{tabular*}{300pt}%
      {|@{\extracolsep{\fill}}c|p{5.5pc}|c|p{5.5pc}|}
```

The main differences between the `tabularx` and `tabular*` environments are:

- `tabularx` modifies the widths of the *columns*, whereas `tabular*` modifies the widths of the intercolumn *spaces*.
- `tabular` and `tabular*` environments may be nested with no restriction, however if one `tabularx` environment occurs inside another, then the inner one *must* be enclosed by `{ }`.
- The body of the `tabularx` environment is in fact the argument to a command, and so certain constructions which are not allowed in command arguments (like `\verb`) may not be used.[5]
- `tabular*` uses a primitive capability of TeX to modify the inter column space of an alignment. `tabularx` has to set the table several times as it searches for the best column widths, and is therefore much slower. Also the fact that the body is expanded several times may break certain TeX constructs.

`\tracingtabularx`

[5] Actually, `\verb` and `\verb*` may be used, but they may treat spaces incorrectly, and the argument can not contain an unmatched `{` or `}`, or a `%` character.

Figure 11.3: Example `tabularx` and `tabular*` with widths of 300pt

```
\begin{tabularx}{300pt}{|c|X|c|X|}
```

Multicolumn entry!		THREE	FOUR
one	The width of this column depends on the width of the table.	three	Column four will act in the same way as column two, with the same width.

```
\begin{tabular*}{300pt}%
{|@{\extracolsep{\fill}}c|p{5.5pc}|c|p{5.5pc}|}
```

Multicolumn entry!		THREE	FOUR
one	The width of this column's text is fixed (5.5pc).	three	Column four will act in the same way as column two, with the same width.

Following the `\tracingtabularx` declaration all later `tabularx` environments will print information about column widths as they repeatedly re-set the tables to find the correct widths.

By default the `X` specification is turned into `p{`⟨*some value*⟩`}`. Such narrow columns often require a special format, which can be achieved by using the `>` syntax. For example, `>{\small}X`. Another format which is useful in narrow columns is raggedright, however LaTeX's `\raggedright` macro redefines `\\` in a way which conflicts with its use in `tabular` or `array` environments.

`\arraybackslash`

For this reason the command `\arraybackslash` is provided; this may be used after a `\raggedright`, `\raggedleft` or `\centering` declaration. Thus a `tabularx` format may include

```
>{\raggedright\arraybackslash}X
```

These format specifications may of course be saved using the command, `\newcolumntype`. After specifying, say,

```
\newcolumntype{Y}{>{\small\raggedright\arraybackslash}X}
```

then `Y` could be used in the `tabularx` format argument.

`\tabularxcolumn`

The `X` columns are set using the `p` column, which corresponds to `\parbox[t]`. You may want them set using, say, the `m` column, which corresponds to `\parbox[c]`. It is not possible to change the column type using the `>` syntax, so another system is provided. `\tabularxcolumn` should be defined to be a macro with one argument, which expands to the `tabular` format specification that you want to correspond to `X`. The argument will be replaced by the calculated width of a column.

The default definition is

```
\newcommand{\tabularxcolumn}[1]{p{#1}}
```

This may be changed, for instance

```
\renewcommand{\tabularxcolumn}[1]{>{\small}m{#1}}
```

so that `X` columns will be typeset as `m` columns using the `\small` font.

Normally all `X` columns in a single table are set to the same width, however it is possible to make `tabularx` set them to different widths. A format argument of

```
{>{\hsize=.5\hsize}X>{\hsize=1.5\hsize}X}
```

specifies two columns, where the second will be three times as wide as the first. If you think you need to do things like this try and redesign your table. However, if you must you should follow these two rules.

- Make sure that the sum of the widths of all the `X` columns is unchanged. (In the above example, the new widths still add up to twice the default width, the same as two standard `X` columns.)
- Do not use `\multicolumn` entries which cross any `X` column.

`tabularx` will not set `X` columns to a negative width. If the widths of the 'normal' columns of the table already total more than the requested total width you will get the warning '`X columns too narrow (table too wide)`'. The `X` columns will be set to a width of 1em and so the table itself will be wider than the requested total width given in the argument to the environment.

The standard `\verb` macro does not work inside a `tabularx`, just as it does not work in the argument to any macro.

```
\TX@verb
```

The 'poor man's `\verb`' (and `\verb*`) defined here is based on page 382 of the *TeXbook*. As explained there, doing verbatim this way means that spaces are not treated correctly, and so `\verb*` may well be useless. 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
```

It must ensure that the real definition is restored afterwards.

This version of `\verb` and `\verb*` are subject to the following restictions:

1. Spaces in the argument are not read verbatim, but may be skipped according to TeX'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 Computer Typewriter font is being used.

11.7 SPACES AND RULES

11.7.1 Spacing

Sometimes tabular rows appear vertically challenged.

`\arraystretch`

The macro `\arraystretch` controls the spacing between rows. The normal space is multiplied by the value of `\arraystretch`, whose default definition is

```
\newcommand{\arraystretch}{1.0}
```

If this is changed to 1.25, for example, the row spacing is increased by 25%.

`\extrarowheight`

If the length `\extrarowheight` is positive, its value is added to the normal height of every row of the array or table, while the depth will remain the same. This is important for tables with horizontal lines because those lines normally touch the capital letters. For example

```
\begin{table}
\centering
\caption{The array and tabular format options.}%
 \label{tab:tabpream}
    \setlength{\extrarowheight}{1pt}
\begin{tabular}{cp{9cm}} \toprule
...
```

was used for Table 11.1.

`\arraycolsep` `\tabcolsep`

The length `\arraycolsep` is half the width of the horizontal space between columns in an `array` environment and similarly the length `\tabcolsep` is half the space between columns in an `tabular` or `tabular*` environment.

`\arrayrulewidth` `\doublerulesep`

The length `\arrayrulewidth` is the width of the line created by a `|` in the format, or by an `\hline`, `\cline` or `\vline` command. The length `\doublerulesep` is the space between lines created by two successive `|` options in the format or by successive `\hline` commands.

11.7.2 Special variations on horizontal lines

The family of `tabular` environments allows vertical positioning with respect to the baseline of the text in which the environment appears. By default the environment appears centered, but this can be changed to align with the first or last line in the environment by supplying a `t` or `b` value to the optional position argument. However, this does not work when the first or last element in the environment is a `\hline` command — in that case the environment is aligned at the horizontal rule.

Here is an example:

Tables with no hline commands versus tables with some hline commands used.

```
Tables
\begin{tabular}[t]{l}
 with no\\ hline\\ commands
\end{tabular} versus tables
\begin{tabular}[t]{|l|}
 \hline
 with some\\ hline\\ commands\\
 \hline
\end{tabular} used.
```

\firsthline \lasthline \extratabsurround

Using `\firsthline` and `\lasthline` will cure the problem, and the tables will align properly as long as their first or last line does not contain extremely large objects.

Tables with no line commands versus tables with some line commands used.

```
Tables
\begin{tabular}[t]{l}
  with no\\ line\\ commands
\end{tabular} versus tables
\begin{tabular}[t]{|l|}
 \firsthline
  with some\\ line\\ commands\\
 \lasthline
\end{tabular} used.
```

The implementation of these two commands contains an extra dimension, which is called `\extratabsurround`, to add some additional space at the top and the bottom of such an environment. This is useful if such tables are nested.

11.7.3 Handling of rules

There are two possible approaches to the handling of horizontal and vertical rules in tables:

1. rules can be placed into the available space without enlarging the table, or
2. rules can be placed between columns or rows thereby enlarging the table.

The class implements the second possibility while the default implementation in the LaTeX kernel implements the first concept.

With standard LaTeX adding rules to a table will not affect the width or height of the table (unless double rules are used), e.g., changing a format from `lll` to `l|l|l` does not affect the document other than adding rules to the table. In contrast, with the class a table that just fits the `\textwidth` might now produce an overfull box. (But you shouldn't have vertical rules in the first place.)

11.8 Free tabulars

All the tabular environments described so far put the table into a box, which LaTeX treats like a large complex character, and characters are not broken across pages. If you have a

long table that runs off the bottom of the page you can turn to, say, the longtable [Car98b] or xtab [Wil00e] packages which will automatically break tables across page boundaries. These have various bells and whistles, such as automatically putting a caption at the top of each page, repeating the column heads, and so forth.

11.8.1 Continuous tabulars

`\begin{ctabular}[`⟨*pos*⟩`]{`⟨*format*⟩`}` rows `\end{ctabular}`

The `ctabular` environment is similar to `tabular`, but with a couple of differences, the main one being that the table will merrily continue across page breaks. The ⟨*format*⟩ argument is the same as for the previous `array` and `tabular` environments, but the optional ⟨*pos*⟩ argument controls the horizontal position of the table, not the vertical. The possible argument value is one of the following characters:

- `l` left justified,
- `c` centered, or
- `r` right justified;

the default is `c`.

```
\begin{ctabular}{lcr}  \toprule
LEFT & CENTER & RIGHT \\  \midrule
l & c & r \\
l & c & r \\
l & c & r \\
l & c & r \\  \bottomrule
\end{ctabular}
```

LEFT	CENTER	RIGHT
l	c	r
l	c	r
l	c	r
l	c	r

An example use is for setting two texts in parallel, for instance a poem and it's translation, without having to be concerned about page breaks.

Je suis Françoys, dont il me pois,	I am François, which is unfortunate,
Né de Paris emprès Pointoise,	born in Paris near Pointoise,
Et de la corde d'une toise	and with a six-foot stretch of rope,
Sçaura mon col que mon cul poise.	my neck will know my arse's weight.

François Villon, 1431–1463?

The `ctabular` environment will probably not be used within a `table` environment (which defeats the possibility of the table crossing page boundaries). To caption a `ctabular` you can define a fixed caption. For example:

```
\newfixedcaption{\freetabcaption}{table}
```

And then `\freetabcaption` can be used like the normal `\caption` within a `table` float.

Table 11.7: Example automatic row ordered table

one	two	three	four	five
six	seven	eight	nine	ten
eleven	twelve	thirteen	fourteen	

11.8.2 Automatic tabulars

A tabular format may be used just to list things, for example the names of the members of a particular organisation, or the names of LaTeX environments.

Especially when drafting a document, or when the number of entries is likely to change, it is convenient to be able to tabulate a list of items without having to explicitly mark the end of each row.

`\autorows[`⟨*width*⟩`]{`⟨*pos*⟩`}{`⟨*num*⟩`}{`⟨*style*⟩`}{`⟨*entries*⟩`}`

The `\autorows` macro lists the ⟨*entries*⟩ in rows; that is, the entries are typeset left to right and top to bottom. The ⟨*num*⟩ argument is the number of columns. The ⟨*entries*⟩ argument is a comma-separated list of the names to be tabulated; there must be no comma after the last of the names before the closing brace. Table 11.7 was set by `\autorows` using:

```
\begin{figure}
\freetabcaption{Example automatic row ordered table}
               \label{tab:autorows}
\autorows{c}{5}{c}{one, two, three, four, five,
                   six, seven, eight, nine, ten,
                   eleven, twelve, thirteen, fourteen }
\end{figure}
```

The ⟨*pos*⟩ argument controls the horizontal position of the tabular and the ⟨*style*⟩ argument specifies the location of the entries in the columns; each column is treated identically. The value of a ⟨*pos*⟩ or ⟨*style*⟩ argument is one of the following characters:

`l` left justified,

`c` centered, or

`r` right justified.

Each column is normally the same width, which is large enough to accomodate the widest entry in the list. A positive ⟨*width*⟩ (e.g., `{0.8\textwidth}`), defines the overall width of the table, and the column width is calculated by dividing ⟨*width*⟩ by the number of columns. Any negative value for the ⟨*width*⟩ width lets each column be wide enough for the widest entry in that column; the column width is no longer a constant.

The examples in Figure 11.4 illustrate the effect of the ⟨*width*⟩ argument (the default value is 0pt). The principal elements of the code for the Figure are:

```
\begin{figure}
...
\autorows[-1pt]{c}{5}{c}{one, two, three, four, five,
                   six, seven, eight, nine, ten,
                   eleven, twelve, thirteen, fourteen }
...
```

⟨*width*⟩ = -1pt

one	two	three	four	five
six	seven	eight	nine	ten
eleven	twelve	thirteen	fourteen	

⟨*width*⟩ = 0pt (the default)

one	two	three	four	five
six	seven	eight	nine	ten
eleven	twelve	thirteen	fourteen	

⟨*width*⟩ = 0.9\textwidth

one	two	three	four	five
six	seven	eight	nine	ten
eleven	twelve	thirteen	fourteen	

Figure 11.4: Changing the width of a row ordered table

```
\autorows[0pt]{c}{5}{c}{one, two, three,
                        ... fourteen }
...
\autorows[0.9\textwidth]{c}{5}{c}{one, two, three,
                                  ... fourteen }
\caption{Changing the width of a row ordered table}
        \label{fig:arw}
\end{figure}
```

`\autocols[`⟨*width*⟩`]{`⟨*pos*⟩`}{`⟨*num*⟩`}{`⟨*style*⟩`}{`⟨*entries*⟩`}`

The `\autocols` macro lists its ⟨*entries*⟩ in columns, proceeding top to bottom and left to right. The arguments, are the same as for `\autorows`, except that a negative ⟨*width*⟩ is treated as if it were zero. The column width is always constant throughout the table and is normally sufficient for the widest entry. A positive or zero ⟨*width*⟩ has the same effect as for `\autorows`.

If you need to include a comma within one of the entries in the list for either `\autorows` or `\autocols` you have to use a macro. For instance:

```
\newcommand*{\comma}{,}
```

The examples in Figure 11.5, from the following code elements, illustrate these points.

```
\begin{figure}
...
\autocols{c}{5}{c}{one\comma{} two, three, four, five,
                   six, seven, eight, nine, ten,
                   eleven, twelve, thirteen, fourteen }
...
\autocols[0.9\textwidth]{c}{5}{c}{one\comma{} two, three,
                                  ... fourteen }
\caption{Changing the width of a column ordered table}
\label{fig:acw}
```

⟨*width*⟩ = `0pt` (the default)

one, two	five	eight	eleven	thirteen
three	six	nine	twelve	fourteen
four	seven	ten		

⟨*width*⟩ = `0.9\textwidth`

one, two	five	eight	eleven	thirteen
three	six	nine	twelve	fourteen
four	seven	ten		

Figure 11.5: Changing the width of a column ordered table

```
\end{figure}
```

Twelve

Page notes

The standard classes provide the `\footnote` command for notes at the bottom of the page. The class provides several styles of footnotes and you can also have several series of footnotes for when the material gets complicated. The normal `\marginpar` command puts notes into the margin, which may float around a little if there are other `\marginpar`s on the page. The class additionally supplies commands for fixed marginal notes and sidebars.

12.1 FOOTNOTES

A footnote can be considered to be a special kind of float that is put at the bottom of a page.

`\footnote[`⟨*num*⟩`]{`⟨*text*⟩`}`

In the main text, the `\footnote` command puts a marker at the point where it is called, and puts the ⟨*text*⟩, preceded by the same mark, at the bottom of the page. If the optional ⟨*num*⟩ is used then its value is used for the mark, otherwise the `footnote` counter is stepped and provides the mark's value. The `\footnote` command should be used in paragraph mode where it puts the note at the bottom of the page, or in a `minipage` where it puts the note at the end of the `minipage`. Results are likely to be peculiar if it is used anywhere else (like in a `tabular`).

`\footnotemark[`⟨*num*⟩`]`
`\footnotetext[`⟨*num*⟩`]{`⟨*text*⟩`}`

You can use `\footnotemark` to put a marker in the main text; the value is determined just like that for `\footnote`. Footnote text can be put at the bottom of the page via `\footnotetext`; if the optional ⟨*num*⟩ is given it is used as the mark's value, otherwise the value of the `footnote` counter is used. It may be helpful, but completely untrue, to think of `\footnote` being defined like:

```
\newcommand{\footnote}[1]{\footnotemark\footnotetext{#1}}
```

In any event, you can use a combination of `\footnotemark` and `\footnotetext` to do footnoting where LaTeX would normally get upset.

`\footref{`⟨*label*⟩`}`

On occasions it may be desireable to make more than one reference to the text of a footnote. This can be done by putting a `\label` in the footnote and then using `\footref` to refer to the label; this prints the footnote mark. For example:

```
...\footnote{...values for the kerning.\label{fn:kerning}} ...
...
```

```
... The footnote\footref{fn:kerning} on \pref{fn:kerning} ... \\
```

In this manual, the last line above prints:

... The footnote[17]

`\multfootsep`

In the standard classes if two or more footnotes are applied sequentially[1,2] then the markers in the text are just run together. The class, like the footmisc [Fai00] and ledmac packages, inserts a separator between the marks. In the class the macro `\multfootsep` is used as the separator. Its default definition is:

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

`\feetabovefloat`
`\feetbelowfloat`

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

12.1.1 A variety of footnotes

`\verbfootnote[`⟨*num*⟩`]{`⟨*text*⟩`}`

The macro `\verbfootnote` is like the normal `\footnote` except that its ⟨*text*⟩ agument can contain verbatim material. For example, the next two paragraphs are typeset by this code:

```
    Below, footnote~\ref{fn1} is a \verb?\footnote? while
footnote~\ref{fn2} is a \verb?\verbfootnote?.

    The \verb?\verbfootnote? command should
appear\footnote{There may be some problems if color is
 used.\label{fn1}}
to give identical results as the normal \verb?\footnote?,
but it can include some verbatim
text\verbfootnote{The \verb?\footnote? macro, like all
                  other macros except for \verb?\verbfootnote?,
                  can not contain verbatim text in its
                  argument.\label{fn2}}
in the \meta{text} argument.
```

Below, footnote 3 is a `\footnote` while footnote 4 is a `\verbfootnote`.

The `\verbfootnote` command should appear[3] to give identical results as the normal `\footnote`, but it can include some verbatim text[4] in the ⟨*text*⟩ argument.

[1] One footnote

[2] Immediately followed by another

[3] There may be some problems if color is used.

[4] The `\footnote` macro, like all other macros except for `\verbfootnote`, can not contain verbatim text in its argument.

```
\plainfootnotes
\twocolumnfootnotes
\threecolumnfootnotes
\paragraphfootnotes
```

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

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

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

```
\footfudgefiddle
```

Paragraphed footnotes may overflow the bottom of a page. TeX has to estimate the amount of space that the paragraph will require once all the footnotes are assembled into it. It then chops off the main text to leave the requisite space at the bottom of the page, following which it assembles and typesets the paragraph. If it underestimated the size then the footnotes will run down the page too far. If this happens then you can change `\footfudgefiddle` to make TeX be more generous in its estimation. The default is 64 and a value about 10% higher should fix most overruns.

```
\renewcommand*{\footfudgefiddle}{70}
```

You must use an integer in the redefinition as the command is going to be used in a place where TeX expects an integer.

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

If you need further series you can create you own. A new footnote series is created by the `\newfootseries` macro, where ⟨*series*⟩ is an alphabetic identifier for the series. This is most conveniently a single (upper case) letter, for example `P`.

Calling, say, `\newfootnoteseries{Q}` creates a set of macros equivalent to those for the normal `\footnote` but with the ⟨*series*⟩ appended. These include `\footnoteQ`,

\footnotemarkQ, \footnotetextQ and so on. These are used just like the normal \footnote and companions.

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

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

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

This can then be used like:

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

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

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

If you can't specify a particular footnote style using the class facilities the footmisc package [Fai00] provides a range of styles. A variety of styles also comes with the ledmac package [Wil03b] which additionally provides several classes of footnotes that can be mixed on a page.

12.1.2 Styling

The parameters controlling the vertical spacing of footnotes are illustrated in Figure 12.1.

There is a discussion in §4.2 starting on page 67 about how to style the \thanks command; footnotes can be similarly styled.

The \footnote macro (and its relations) essentially does three things:

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

```
\@makefnmark
\@thefnmark
```

The \footnote macro calls the kernel command \@makefnmark to typeset the footnote marker at the point where \footnote is called (the value of the marker is kept in the macro \@thefnmark which is defined by the \footnote or \footnotemark macros). The default definition typesets the mark as a superscript and is effectively

```
\newcommand*{\@makefnmark}{\hbox{\textsuperscript{\@thefnmark}}}
```

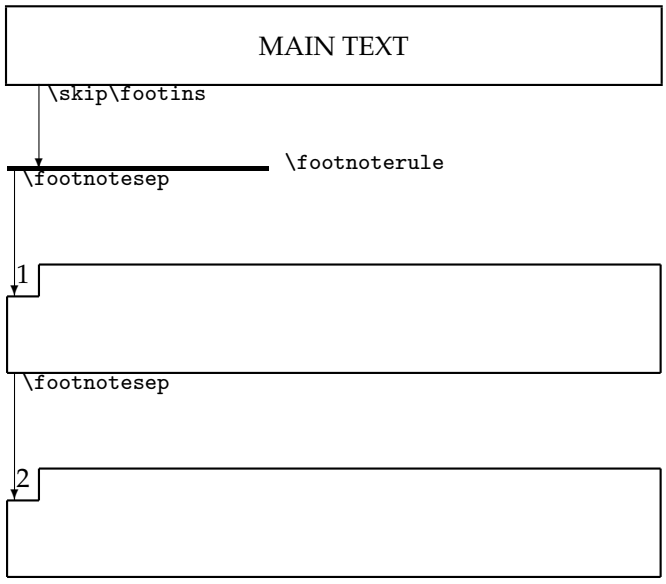


Figure 12.1: Footnote layout parameters

You can change this if, for example, you wanted the marks to be in parentheses at the baseline.

```
\renewcommand*{\@makefnmark}{{\footnotesize (\@thefnmark)}}
```

or, somewhat better to take account of the size of the surrounding text

```
\renewcommand*{\@makefnmark}{\slashfracstyle{(\@thefnmark)}}
```

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

The class macro for typesetting the marker at the foot of the page is `\footfootmark`. The appearance of the mark is controlled by `\footmarkstyle`. The default specification is

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

where the `#1` indicates the position of `\@thefnmark` in the style. The default results in the mark being set as a superscript. For example, to have the marker set on the baseline and followed by a right parenthesis, do

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

`\footmarkwidth` `\footmarksep` `\footparindent`

The mark is typeset in a box of width `\footmarkwidth` If this is negative, the mark is outdented into the margin, if zero the mark is flush left, and when positive the mark is indented. The mark is followed by the text of the footnote. Second and later lines of the text are offset by the length `\footmarksep` from the end of the box. The first line of a paragraph within a footnote is indented by `\footparindent`. The default values for these lengths are:

Table 12.1: Some footnote text styles

\footmarkwidth	\footmarksep	Comment
1.8em	-1.8em	Flushleft, regular indented paragraph (the default)
1.8em	0em	Indented, block paragraph hung on the mark
0em	0em	Flushleft, block paragraph
-1.8em	1.8em	Block paragraph, flushleft, mark in the margin
-1sp	0em	Block paragraph, flushleft, mark in the margin but flush against the text

```
\setlength{\footmarkwidth}{1.8em}
\setlength{\footmarksep}{-\footmarkwidth}
\setlength{\footparindent}{1em}
```

\foottextfont

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

Altogether, the class specifies

```
\footmarkstyle{\textsuperscript{#1}}
\setlength{\footmarkwidth}{1.8em}
\setlength{\footmarksep}{-1.8em}
\setlength{\footparindent}{1em}
\newcommand{\foottextfont}{\footnotesize}
```

to replicate the standard footnote layout.

You might like to try the combinations of \footmarkwidth and \footmarksep listed in Table 12.1 to see which you might prefer. Not listed in the Table, to get the marker flushleft and then the text set as a block paragraph you can try:

```
\setlength{\footmarkwidth}{1.8em}
\setlength{\footmarksep}{0em}
\footmarkstyle{#1\hfill}
```

As an example of a rather different scheme, in at least one discipline the footnoted text in the main body has a marker at each end. It is possible to define a macro to do this:

```
\newcommand{\wrapfootnote}[1]{\stepcounter\@mpfn%
  %  marks in the text
  \protected@xdef\@thefnmark{\thempfn}%
  \@footnotemark #1\@footnotemark%
  %  marks at the bottom
  \protected@xdef\@thefnmark{\thempfn--\thempfn}%
  \@footnotetext}
```

The macro is based on a posting to CTT by Donald Arseneau in November 2003, and is used like this:

```
Some
\wrapfootnote{disciplines}{For example, Celtic studies.}
require double marks in the text.
```

Some [5]disciplines[5] require double marks in the text.

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

Any footnotes after this point will be set according to:

```
\setlength{\footmarkwidth}{-1.0em}
\setlength{\footmarksep}{-\footmarkwidth}
\footmarkstyle{#1}
```

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

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

This, as shown by `\@fnsymbol{1}, ...\@fnsymbol{9}` produces the series:

∗, †, ‡, §, ¶, ‖, ∗∗, ††, and ‡‡.

Robert Bringhurst quotes the following as the traditional ordering (at least up to ¶):

∗, †, ‡, §, ‖, ¶, ∗∗, ††, and ‡‡.

You can obtain this sequence by redefining `\@fnsymbol` as:

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

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

To get the footnote reference marks set with symbols use:

```
\renewcommand*{\thefootnote}{\fnsymbol{footnote}}
```

or to use roman numerals instead of the regular arabic numbers:

```
\renewcommand*{\thefootnote}{\roman{footnote}}
```

`\footnoterule`

[5–5]For example, Celtic studies.

The rule separating footnotes from the main text is specified by `\footnoterule`:

```
\newcommand*{\footnoterule}{%
  \kern-3pt%
  \hrule width 0.4\columnwidth
  \kern 2.6pt}
```

If you don't want a rule (but you might later), then the easiest method is:

```
\let\oldfootnoterule\footnoterule
\renewcommand*{\footnoterule}{}
```

and if you later want rules you can write:

```
\let\footnoterule\oldfootnoterule
```

In Figure 12.1 we see that the footnotes are separated from the text by `\skip\footins`. We provide a special interface to set this skip:

Added 2015/04/22

`\setfootins{`⟨*length for normal*⟩`}{`⟨*length for minipage*⟩`}`

The default is similar to

```
\setfootins{\bigskipamount}{\bigskipamount}
```

Internally `\setfootins` also sets the skips being used by `\twocolumnfootnotes` and friends.

12.2 MARGINAL NOTES

Some marginalia can also be considered to be kinds of floats. The class provides the standard margin notes via `\marginpar`. Remember that the width of the margin note, the separation from the text, and the separation from one `\marginpar` to another is controlled by `\setmarginnotes`, see Section 2.5 on page 20.

`\marginpar[`⟨*left-text*⟩`]{`⟨*text*⟩`}`
`\marginparmargin{`⟨*placement*⟩`}`
`\reversemarginpar`
`\normalmarginpar`

Just as a reminder, the `\marginpar` macro puts ⟨*text*⟩ into the margin alongside the typeblock — the particular margin depends on the document style and the particular page.

The interface for specifying which margin `\marginpar` (and friends) write to, have long been quite cluttered, so we have in 2010 adopted a more textual and natural interface. For `\marginpar` the macro is named `\marginparmargin{`⟨*placement*⟩`}` with possible placements: *left*, *right*, *outer*, and *inner*. The interpretation of which is explained in Figure 12.2. The default corresponds to `\marginparmargin{outer}`.

The original convoluted methods of specifying the margin for `\marginpar` is deprecated, although still supported; if you need to know what they are then you can read all about them in `memoir.dtx`.

Sometimes LaTeX gets confused near a page break and a note just after a break may get put into the wrong margin (the wrong margin for the current page but the right one if

`\Xmargin{`⟨*placement*⟩`}` for possible placements: *left*, *right*, *outer*, and *inner*

Two column document If the note is placed in the first column, to the left, otherwise to the right, irrespective the document being one- or two-side and of the users choices

One sided document If user specified *left*, notes are placed to the left, otherwise to the right.

Two sided document depends on whether a recto or verso page:

Recto (odd) page note is placed on the right if the user specified *right* or *outer*, otherwise the note is placed on the left.

Verso (even) page note is placed on the left if the user specified *left* or *outer*, otherwise the note is placed on the right.

Figure 12.2: Interpretation of the arguments to the `\Xmargin` commands for specifying the side in which to place side note like material. `X` here equals `marginpar`, `sidepar`, `sidebar`, or `sidefoot`.

the note fell on the previous page). If this occurs then inserting the `\strictpagecheck` declaration before any `\marginpar` command is used will prevent this, at the cost of at least one additional LaTeX run.

12.3 Side notes

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

```
\sidepar[⟨left⟩]{⟨right⟩}
\sideparmargin{⟨placement⟩}
\sideparfont
\sideparform
\sideparvshift
```

The `\sidepar` macro is similar to `\marginpar` except that it produces side notes that do not float — they may overlap.

The same spacing is used for both `\marginpar` and `\sidepar`, namely the lengths `\marginparsep` and `\marginparwidth`. See `\setmarginnotes`, in Section 2.5 on page 20.

The length `\sideparvshift` can be used to make vertical adjustments to the position of `\sidepar` notes. By default this is set to a value of 0pt which should align the top of the note with the text line.

The command `\sideparfont` is used to specify the font used for the `\sidepar`, default is `\normalfont\normalsize`.

While `\sideparfont` holds the font settings for the sidepar, the local adjustment is kept in `\sideparform`, the default is

```
\newcommand*{\sideparform}{%
  \ifmemtortm\raggedright\else\raggedleft\fi}
```

Which is a special construction the makes the text go flush against the text block on side specified via `\sideparmargin`. Since the margin par area is usually quite narrow it might be an idea to use a ragged setup which enables hyphenation. This can be achieved by

```
\usepackage{ragged2e}
\newcommand*{\sideparform}{%
  \ifmemtortm\RaggedRight\else\RaggedLeft\fi}
```

The macro `\sideparmargin{`⟨*placement*⟩`}` can be used to specify which margin the side note should go to. ⟨*placement*⟩ should be one of *left*, *right*, *outer*, or *inner*. Interpretation of which is explained in Figure 12.2. For some now forgotten reason the default corresponds to `\sideparmargin{left}`.[6]

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

The original convoluted methods of specifying the margin for `\sidepar` is deprecated, although still supported; if you need to know what they are then you can read all about them in `memoir.dtx`.

`\parnopar`

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

Bastiaan Veelo has kindly provided example code for another form of a side note, the code is shown in Sniplet C.3 on page 408.

Bastiaan also noted that it provided an example of using the `\foremargin` length. If you want to try it out, either put the code in your preamble, or put it into a package (i.e., `.sty` file) without the `\makeat...` commands.

12.4 SIDEBARS

Sidebars are typeset in the margin and usually contain material that is ancilliary to the main text. They may be long and extend for more than one page.[7]

6 As not to change existing documents, we have decided to leave it like that.
7 Donald Arseneau's help has been invaluable in getting the sidebar code to work.

`\sidebar{`⟨*text*⟩`}`

The `\sidebar` command is like `\marginpar` in that it sets the ⟨*text*⟩ in the margin. However, unlike `\marginpar` the ⟨*text*⟩ will start near the top of the page, and may continue onto later pages if it is too long to go on a single page. If multiple `\sidebar` commands are used on a page, the several ⟨*text*⟩s are set one after the other.

`\sidebarmargin{`⟨*margin*⟩`}`

The macro `\sidebarmargin{`⟨*placement*⟩`}` can be used to specify which margin the side note should go to. ⟨*placement*⟩ should be one of *left*, *right*, *outer*, or *inner*. Interpretation of which is explained in Figure 12.2. The default corresponds to `\sidebarmargin{outer}`.

`\sidebarfont` `\sidebarform`

The sidebar ⟨*text*⟩ is typeset using the `\sidebarfont`, whose initial definition is

```
\newcommand{\sidebarfont}{\normalsize\normalfont}
```

Sidebars are normally narrow so the text is set raggedright to reduce hyphenation problems and stop items in environments like `itemize` from overflowing. More accurately, the text is set according to `\sidebarform` which is defined as:

```
\newcommand*{\sidebarform}{%
  \ifmemtortm\raggedright\else\raggedleft\fi}
```

Which is a special construction the makes the text go flush against the text block on side specified via `\sideparmargin`. Since the margin par area is usually quite narrow it might be an idea to use a ragged setup which enables hyphenation. This can be achieved by

```
\usepackage{ragged2e}
\newcommand*{\sidebarform}{%
  \ifmemtortm\RaggedRight\else\RaggedLeft\fi}
```

You may run into problems if the `\sidebar` command comes near a pagebreak, or if the sidebar text gets typeset alongside main text that has non-uniform line spacing (like around a `\section`). Further, the contents of sidebars may not be typeset if they are too near to the end of the document.

`\sidebarwidth` `\sidebarhsep` `\sidebarvsep`

The ⟨*text*⟩ of a `\sidebar` is typeset in a column of width `\sidebarwidth` and there is a horizontal gap of `\sidebarhsep` between the main text and the sidebar. The length `\sidebarvsep` is the vertical gap between sidebars that fall on the same page; it also has a role in controlling the start of sidebars with respect to the top of the page.

`\sidebartopsep`
`\setsidebarheight{`⟨*height*⟩`}`

The length `\sidebartopsep` controls the vertical position of the top of a sidebar. The default is 0pt which aligns it with the top of the typeblock. The command `\setsidebarheight` sets the height of sidebars to ⟨*height*⟩, without making any allowance for `\sidebartopsep`. The ⟨*length*⟩ argument should be equivalent to an integral number of lines. For example:

```
\setsidebarheight{15\onelineskip}
```

The default is the \textheight.

Perhaps you would like sidebars to start two lines below the top of the typeblock but still end at the bottom of the typeblock? If so, and you are using the calc package [TJ05], then the following will do the job:

```
\setlength{\sidebartopskip}{2\onelineskip}
\setsidebarheight{\textheight-\sidebartopskip}
```

The alignment of the text in a sidebar with the main text may not be particularly good and you may wish to do some experimentation (possibly through a combination of \sidebarvsep and \setsidebarheight) to improve matters.

Although you can set the parameters for your sidebars individually it is more efficient to use the \setsidebars command; it *must* be used if you change the font and/or the height.

\setsidebars{⟨*hsep*⟩}{⟨*width*⟩}{⟨*vsep*⟩}{⟨*topsep*⟩}{⟨*font*⟩}{⟨*height*⟩}

The \setsidebars command can be used to set the sidebar parameters. \sidebarhsep is set to ⟨*hsep*⟩, \sidebarwidth is set to ⟨*width*⟩, \sidebarvsep is set to ⟨*vsep*⟩, \sidebartopsep is set to ⟨*topsep*⟩, \sidebarfont is set to ⟨*font*⟩, and finally \setsidebarheight is used to set the height to ⟨*height*⟩. The default is:

```
\setsidebars{\marginparsep}{\marginparwidth}{\onelineskip}%
            {0pt}{\normalsize\normalfont}{\textheight}
```

Any, or all, of the arguments can be a *, in which case the parameter corresponding to that argument is unchanged. Repeating the above example of changing the topskip and the height, assuming that the other defaults are satisfactory except that the width should be 3cm and an italic font should be used:

```
\setsidebars{*}{3cm}{*}{2\onelineskip}{\itshape}%
            {\textheight-\sidebartopsep}
```

Changing the marginpar parameters, for example with \setmarginnotes, will not affect the sidebar parameters.

Note that \checkandfixthelayout neither checks nor fixes any of the sidebar parameters. This means, for instance, that if you change the \textheight from its default value and you want sidebars to have the same height then after changing the \textheight you have to call \checkandfixthelayout and then call \setsidebars with the (new) \textheight. For instance:

```
...
\settypeblocksize{40\baselineskip}{5in}{*}
...
\checkandfixthelayout
\setsidebars{...}{...}{...}{...}{...}{\textheight}
```

Unfortunately if a sidebar is on a double column page that either includes a double column float or starts a new chapter then the top of the sidebar comes below the float or the chapter title. I have been unable to eliminate this 'feature'.

12.5 SIDE FOOTNOTES

Besides three already mentioned macros for writing in the margin (\marginpar, \sidepar, and \sidebar) memoir also provide a functionality to add side footnotes. Ac-

tually two ways: one is to internally make \footnote use \marginpar to write in the margin, the other is to collect all side footnotes bottom up in the margin.

```
\footnotesatfoot
\footnotesinmargin
```

\footnotesatfoot (the default) causes \footnote to place its text at the bottom of the page. By issuing \footnotesinmargin \footnote (and friends like \footnotetext)will internally use \marginpar to write the footnote to the page.

12.5.1 Bottom aligned side footnotes

Bottom aligned footnotes works just like regular footnotes, just with a separate macro \sidefootenote{⟨*text*⟩}, and here the side footnotes are placed at the bottom of the specified margin (more or like as if one had taken the footnotes from the bottom of the page and moved it to the margin instead). All the major functionality is the same as for the normal \footnote command.[8]

```
\sidefootnote[⟨num⟩]{⟨text⟩}
\sidefootnotemark[⟨num⟩]
\sidefootnotetext[⟨num⟩]{⟨text⟩}
```

By default the regular footnotes and the side footnotes use different counters. If one would like them to use the same counter, issue the following in the preamble:

```
\letcountercounter{sidefootnote}{footnote}
```

12.5.2 Setting the layout for \sidefootnote

There are several possibilities to change the appearance of the \sidefootnote:

Specifying the margin in which the side footnote should go, is done by

```
\sidefootmargin{⟨keyword⟩}
```

where ⟨*keyword*⟩ can be *left*, *right*, *outer*, and *inner*, and their meaning is explained in Figure 12.2. The default is *outer*.

```
\sidefoothsep
\sidefootwidth
\sidefootvsep
```

\sidefoothsep is a length controlling the separation from the text to the side footnote column, default \marginparsep. \sidefootwidth is length controlling the width of the side footnote column, default \marginparwidth, and \sidefootvsep is the vertical distance between two side footnotes, default \onelineskip.

```
\sidefootadjust
\setsidefootheight{⟨height⟩}
\sidefootfont
```

\sidefootadjust is a length which specifies the placement of the side footnote column in relation to the bottom of the text block, the default is 0pt. \setsidefootheight sets the maximal height of the side footnote column, default

8 \sidefootnote does not make sense inside minipages...

`textwidth`. Lastly `\sidefootfont` holds the general font setting for the side footnote,[9] default `\normalfont\footnotesize`.

The macro

`\setsidefeet{`⟨*hsep*⟩`}{`⟨*width*⟩`}{`⟨*vsep*⟩`}{`⟨*adj*⟩`}{`⟨*font*⟩`}{`⟨*height*⟩`}`

sets the specifications all six settings above in one go.. An '*' means 'use the current value'. So memoir internally use the following default

```
\setsidefeet{\marginparsep}{\marginparwidth}%
{\onelineskip}{0pt}%
{\normalfont\footnotesize}{\textheight}%
```

It is recommended to use this macro along with the other macros in the preamble to specify document layout.

12.5.3 Styling `\sidefootnote`

`\sidefootmarkstyle{`⟨*code*⟩`}`

controls how the side footnote counter is typeset in the side footnote. The default is

```
\sidefootmarkstyle{\textsuperscript{#1}}
```

The mark is typeset in a box of width `\sidefootmarkwidth` If this is negative, the mark is outdented into the margin, if zero the mark is flush left, and when positive the mark is indented. The mark is followed by the text of the footnote. Second and later lines of the text are offset by the length `\sidefootmarksep` from the end of the box. The first line of a paragraph within a footnote is indented by `\sidefootparindent`. The default values for these lengths are:

```
\setlength{\sidefootmarkwidth}{0em}
\setlength{\sidefootmarksep}{0em}
\setlength{\sidefootparindent}{1em}
```

Caveat: It is natural to specify a length as `\sidefootparindent` as a LaTeX length, but it has a down side. If, as we do here, set the value to 1em, then since the size of the em unit changes with the current font size, one will actually end up with an indent corresponding to the font size being used when the

```
\setlength{\sidefootparindent}{1em}
```

was issued, not when it has used (where the font size most often will be `\footnotesize`).

At this point we consider this to be a *feature* not an error. One way to get pass this problem it the following

```
\begingroup% keep font change local
\sidefoottextfont
\global\setlength\sidefootparindent{1em}
\endgroup
```

Then it will store the value of em corresponding to the font being used.

`\sidefoottextfont`

9 There is a similar macro to control the font of the text alone.

holds the font being used by the side footnote, default `\normalfont\footnotesize`.

`\sidefootform`

is used to specify the raggedness of the text. Default

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

which is much like `\raggedright` but allows some hyphenation. One might consider using

```
\usepackage{ragged2e}
\newcommand*{\sidefootform}{\RaggedRight}
```

Which does something similar.

12.5.4 Side footnote example

In the margin you will find the result of the following code:

```
Testing\sidefootnote{This is test} bottom aligned
footnotes.\sidefootnote{This is another side
footnote, spanning several lines.

And several paragraphs}\sidefootnote{And number three}
```

Testing[1] bottom aligned footnotes.[2,3]

12.6 ENDNOTES

Reimplemented, December 2010[10]

Endnotes are often used instead of footnotes so as not to interrupt the flow of the main text. Although endnotes are normally put at the end of the document, they may instead be put at the end of each chapter.

The endnotes package already uses the command `\endnote` for an endnote, so the class uses `\pagenote` for an endnote so as not to clash if you prefer to use the package.

`\makepagenote`
`\pagenote[`⟨*id*⟩`]{`⟨*text*⟩`}`
`\printpagenotes` `\printpagenotes*`

The general principle is that notes are written out to a file which is then input at the place where the notes are to be printed. The note file has an `ent` extension, like the table of contents file has a `toc` extension.

You have to put `\makepagenote` in your preamble if you want endnotes. This will open the `ent` note file which is called `\jobname.ent`.

In the body of the text use use `\pagenote` to create an endnote, just as you would use `\footnote` to create a footnote. In the books that I have checked there are two common methods of identifying an endnote:

[1] This is test

[2] This is another side footnote, spanning several lines.

And several paragraphs

[3] And number three

10 The former implementation had some difficulties handling certain types of input. A few of the macros used to format the output are no longer supported/used in the new implementation.

1. Like a footnote, put a number in the text at the location of the note and use the same number to identify the note when it finally gets printed.
2. Put no mark in the text, but when it is finally printed use a few words from the text to identify the origin of the note. The page number is often used as well with this method.

The ⟨*text*⟩ argument of `\pagenote` is the contents of the note and if the optional ⟨*id*⟩ argument is not used the result is similar to having used `\footnote` — a number in the main text and the corresponding number in the endnotes listing (as in 1 above). For the second reference style (2 above) use the optional ⟨*id*⟩ argument for the 'few words', and no mark will be put into the main text but ⟨*id*⟩ will be used as the identification in the listing.

For one set of endnotes covering the whole document put `\printpagenotes` where you want them printed, typically before any bibliography or index. The `\printpagenotes` macro inputs the `ent` endnote file for printing and then closes it to any further notes.

For notes at the end of each chapter put `\printpagenotes*`, which inputs the `ent` file for printing then empties it ready for more notes, at the end of each chapter.

The simple use is like this:

```
\documentclass[...]{memoir}
...
\makepagenote
...
\begin{document}
\chapter{One}
...\pagenote{An end note.} ...
...\pagenote{Fascinating information.}
...
\chapter{Last}% chapter 9
...\pagenote{Another note.}% 30th note
...
...
\printpagenotes
...
\end{document}
```

This will result in an endnote listing looking like Figure 12.3.

For notes at the end of each chapter:

```
\documentclass[...]{memoir}
...
\makepagenote
...
\begin{document}
\chapter{One}
...\pagenote{An end note.} ...
...
\printpagenotes*
\chapter{Last}
```

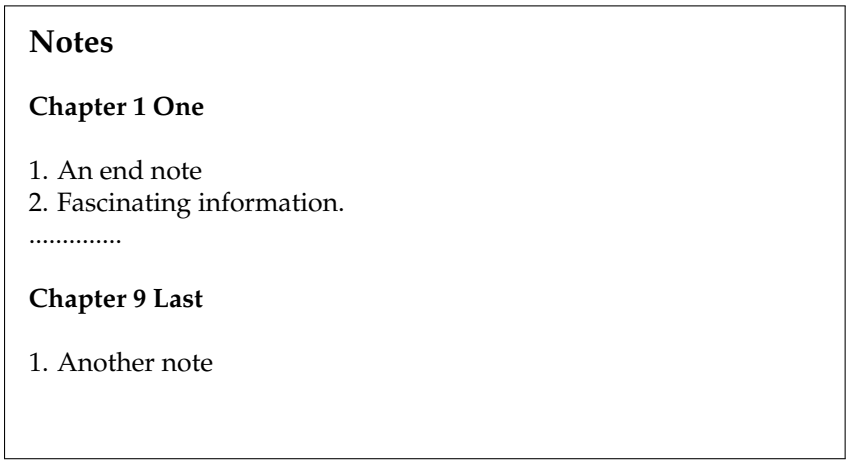


Figure 12.3: Example endnote listing

```
...\pagenote{Another note.} ...
...
\printpagenotes*
%%% no more chapters
...
\end{document}
```

\continuousnotenums \notepageref

The pagenote counter is used for the notes. By default the endnotes are numbered per chapter. If you want the numbering to be continuous throughout the document use the \continuousnotenums declaration. Normally the information on which page a note was created is discarded but will be made available to notes in the endnote listing following the \notepageref declaration. Both \continuousnotenums and \notepageref can only be used in the preamble.

\notesname \notedivision

When \printpagenotes (or \printpagenotes*) is called the first thing it does is call the macro \notedivision. By default this is defined as:

```
\newcommand*{\notedivision}{\chapter{\notesname}}
```

with

```
\newcommand*{\notesname}{Notes}
```

In other words, it will print out a heading for the notes that will be read from the ent file. \print... then closes the ent file for writing and after this \inputs it to get and process the notes.

12.6.1 Changing the appearance

In the text

```
\notenumintext{⟨num⟩}
```

The `pagenote` counter is used for pagenotes. The macro `\notenumintext` is called by `\pagenote` with the value of the `pagenote` counter as the ⟨*num*⟩ argument to print the value of the `pagenote` counter in the main text. By default it is printed as a superscript, but this can be changed, or even eliminated.

```
\newcommand*{\notenumintext}[1]{\textsuperscript{#1}}
```

In the page note list

To better understand how a page note entry is formatted in the page note list, we start with the following pseudo code (it is not exactly what you will see in the `.ent` file, but macros will end up being called in this manner)

```
\prenoteinnotes
\noteidinnotes{⟨notenum⟩}{⟨id⟩}
\pageinnotes{⟨auto generated note label key⟩}
\prenotetext
  ⟨Page note text⟩
\postnotetext
\postnoteinnotes
```

At the start and end we have the two macros `\prenoteinnotes` and `\postnoteinnotes`, they take care of preparing for and ending an entry in the list. The list is typeset in a manner where each item is (at least) a paragraph, so the default definition is

```
\newcommand{\prenoteinnotes}{\par\noindent}
\newcommand{\postnoteinnotes}{\par}
```

A user could change this to make it look a bit more like a list construction. For example the following would give a hanging indentation

```
\renewcommand{\prenoteinnotes}{\par\noindent\hangindent 2em}
```

The `\noteidinnotes` calls `\idtextinnotes` to print the note ⟨*id*⟩ if it was given as the optional argument to `pagenote`, otherwise it calls `\notenuminnotes` to print the note number.

```
\noteidinnotes{⟨notenum⟩}{⟨id⟩}
\idtextinnotes{⟨id⟩}
\notenuminnotes{⟨num⟩}
```

These are defined respectively as:

```
\newcommand*{\idtextinnotes}[1]{[#1]\space}
\newcommand*{\notenuminnotes}[1]{\normalfont #1.\space}
```

Next we execute `\pageinnotes{`⟨*note label key*⟩`}` which does nothing by default. But if `\notepageref` is issued in the preamble two things happen, (1) each page note issues a label such that we can refer back to its page, and (2) `\pageinnotes` calls `\printpageinnotes` (or if hyperref is loaded `\printpageinnoteshyperref`)

\pageinnotes{⟨*auto generated note label key*⟩} \printpageinnotes{⟨*auto generated note label key*⟩} \printpageinnoteshyperref{⟨*auto generated note label key*⟩} \pagerefname

Default definitions

```
\newcommand*{\printpageinnotes}[1]{%
    (\pagerefname\ \pageref{#1})\space}
\newcommand\printpageinnoteshyperref[1]{%
   (\hyperref[#1]{\pagerefname\ \pageref*{#1}})\space}
```

That is if hyperref is loaded the entire text ⟨*page 3*⟩ will be the text of a hyperlink.

\prenotetext \postnotetext

The actual text part of the page note is enclosed by \prenotetext and postnotetext. By default they do nothing, but could easily be redefined such that (only) the entry text would be in italic:

```
\renewcommand\prenotetext{\begingroup\itshape}
\renewcommand\postnotetext{\endgroup}
```

\addtonotes{⟨*text*⟩}

The macro \addtonotes inserts ⟨*text*⟩ into the ent file.

Note. *As the argument to* \pagenote *and* \addtonotes *is moving you may have to* \protect *any fragile commands. If you get strange error messages, try using* \protect *and see if they go away.*

Internally in \pagenote \addtonotes is used to provide chapter devisions into the note list. It will detect both numbered and unnumbered chapters. The actual text is provided using

\pagenotesubhead{⟨*chapapp*⟩}{⟨*num*⟩}{⟨*title*⟩} \pagenotesubheadstarred{⟨*chapapp*⟩}{⟨*num*⟩}{⟨*title*⟩} \pnchap \pnschap

The macro \pagenotesubhead typesets the subheadings in an endnote list. The ⟨*chapapp*⟩ argument is normally \chaptername but if the notes are from an appendix then \appendixname is used instead. ⟨*num*⟩ is the number of the chapter, or blank if there is no number. Lastly, ⟨*title*⟩ is \pnchap for regular chapters which defaults to the ToC entry, or \pnschap for starred chapters which defaults to the normal title. The default definition of \pagenotesubhead is very simply:

```
\newcommand*{\pagenotesubhead}[3]{%
  \section*{#1 #2 #3}}
\newcommand\pagenotesubheadstarred{\pagenotesubhead} % i.e. the same
```

By default this means that the header for starred chapters will be something like »Chapter Title«, which may look odd. In that case redefine `\pagenotesubheadstarred` to something similar to

```
\renewcommand\pagenotesubheadstarred[3]{\section*{#3}}
```

Just remember that unless you have specified `\continuousnotenums` in the preamble the note counter (`pagenote`) will only be reset at the start of any numbered chapters (because it is tied to changes in the chapter counter).

The scheme is set up under the assumption that notes will only be printed at the end of the document. If you intend to put them at the end of each chapter, then you will probably want to change the definitions of the `\notedivision` and `\pagenotesubhead` macros. For example:

```
\renewcommand*{\notedivision}{\section*{\notesname}}
\renewcommand*{\pagenotesubhead}[3]{}
```

and remember to use `\printpagenotes*` at each place you want the current set of notes to be printed.

Say you have written a document with footnotes, but later on decide on using end notes (page notes) instead. In that case you can use `\foottopagenote` to make `\footnote`, `\footnotemark` and `\footnotetext` works as if it was implemented using end notes. On the other hand `\pagetofootnote` makes all page notes into footnotes (note that this might not work, because there are places where page notes can be issued but foot notes cannot).

```
\foottopagenote
\pagetofootnote
```

In either conversion the optional argument will be ignored as for `\pagenote` it can be arbitrary text whereas for `\footnote` it must be a number.

Thirteen

Decorative text

> Too servile a submission to the books and opinions of the ancients has spoiled many an ingenious man, and plagued the world with an abundance of pedants and coxcombs.
>
> James Puckle (1677?–1724)

By now we have covered most aspects of typesetting. As far as the class is concerned this chapter describes the slightly more fun task of typesetting epigraphs.

Some authors like to add an interesting quotation at either the start or end of a chapter. The class provides commands to assist in the typesetting of a single epigraph. Other authors like to add many such quotations and the class provides environments to cater for these as well. Epigraphs can be typeset at either the left, the center or the right of the typeblock. A few example epigraphs are exhibited here, and others can be found in an article by Christina Thiele [Thi99] where she reviewed the epigraph package [Wil00a] which is included in the class.

13.1 Epigraphs

The original inspiration for `\epigraph` was Doug Schenck's for the epigraphs in our book [SW94]. That was hard wired for the purpose at hand. The version here provides much more flexibility.

```
\epigraph{⟨text⟩}{⟨source⟩}
```

The command `\epigraph` typesets an epigraph using ⟨*text*⟩ as the main text of the epigraph and ⟨*source*⟩ being the original author (or book, article, etc.) of the quoted text. By default the epigraph is placed at the right hand side of the typeblock, and the ⟨*source*⟩ is typeset at the bottom right of the ⟨*text*⟩.

```
\begin{epigraphs}
\qitem{⟨text⟩}{⟨source⟩}
...
\end{epigraphs}
```

The `epigraphs` environment typesets a list of epigraphs, and by default places them at the right hand side of the typeblock. Each epigraph in an `epigraphs` environment is specified

by a `\qitem` (analagous to the `\item` command in ordinary list environments). By default, the ⟨*source*⟩ is typeset at the bottom right of the ⟨*text*⟩.

13.2 GENERAL

> Example is the school of mankind,
> and they will learn at no other.
>
> *Letters on a Regicide Peace*
> EDMUND BURKE

The commands described in this section apply to both the `\epigraph` command and the `epigraphs` environment. But first of all, note that an epigraph immediately after a heading will cause the first paragraph of the following text to be indented. If you want the initial paragraph to have no indentation, then start it with the `\noindent` command.

```
\epigraphwidth
\epigraphposition{⟨flush⟩}
```

The epigraphs are typeset in a minipage of width `\epigraphwidth`. The default value for this can be changed using the `\setlength` command. Typically, epigraphs are typeset in a measure much less than the width of the typeblock. The horizontal position of an epigraph in relation to the main typeblock is controlled by the ⟨*flush*⟩ argument to the `\epigraphposition` declaration. The default value is `flushright`, so that epigraphs are set at the right hand side of the typeblock. This can be changed to `flushleft` for positioning at the left hand side or to `center` for positioning at the center of the typeblock.

```
\epigraphtextposition{⟨flush⟩}
```

In order to avoid bad line breaks, the epigraph ⟨*text*⟩ is normally typeset raggedright. The ⟨*flush*⟩ argument to the `\epigraphtextposition` declaration controls the ⟨*text*⟩ typesetting style. By default this is `flushleft` (which produces raggedright text). The sensible values are `center` for centered text, `flushright` for raggedleft text, and `flushleftright` for normal justified text.

If by any chance you want the ⟨*text*⟩ to be typeset in some other layout style, the easiest way to do this is by defining a new environment which sets the paragraphing parameters to your desired values. For example, as the ⟨*text*⟩ is typeset in a minipage, there is no paragraph indentation. If you want the paragraphs to be indented and justified then define a new environment like:

```
\newenvironment{myparastyle}{\setlength{\parindent}{1em}}{}
```

and use it as:

```
\epigraphtextposition{myparastyle}
```

```
\epigraphsourceposition{⟨flush⟩}
```

The ⟨*flush*⟩ argument to the `\epigraphsourceposition` declaration controls the position of the ⟨*source*⟩. The default value is `flushright`. It can be changed to `flushleft`, `center` or `flushleftright`.

For example, to have epigraphs centered with the ⟨*source*⟩ at the left, add the following to your document.

```
\epigraphposition{center}
\epigraphsourceposition{flushleft}
```

`\epigraphfontsize{`⟨*fontsize*⟩`}`

Epigraphs are often typeset in a smaller font than the main text. The ⟨*fontsize*⟩ argument to the `\epigraphfontsize` declaration sets the font size to be used. If you don't like the default value (`\small`), you can easily change it to, say `\footnotesize` by:

```
\epigraphfontsize{\footnotesize}
```

`\epigraphrule`

By default, a rule is drawn between the ⟨*text*⟩ and ⟨*source*⟩, with the rule thickness being given by the value of `\epigraphrule`. The value can be changed by using `\setlength`. A value of `0pt` will eliminate the rule. Personally, I dislike the rule in the list environments.

```
\beforeepigraphskip
\afterepigraphskip
```

The two `...skip` commands specify the amount of vertical space inserted before and after typeset epigraphs. Again, these can be changed by `\setlength`. It is desireable that the sum of their values should be an integer multiple of the `\baselineskip`.

Note that you can use normal LaTeX commands in the ⟨*text*⟩ and ⟨*source*⟩ arguments. You may wish to use different fonts for the ⟨*text*⟩ (say roman) and the ⟨*source*⟩ (say italic).

The epigraph at the start of this section was specified as:

```
\epigraph{Example is the school of mankind,
          and they will learn at no other.}
 {\textit{Letters on a Regicide Peace}\\ \textsc{Edmund Burke}}
```

13.3 EPIGRAPHS BEFORE CHAPTER HEADINGS

> If all else fails, immortality can always be assured by spectacular error.
>
> John Kenneth Galbraith

The `\epigraph` command and the `epigraphs` environment typeset an epigraph at the point in the text where they are placed. The first thing that a `\chapter` command does is to start off a new page, so another mechanism is provided for placing an epigraph just before a chapter heading.

`\epigraphhead[`⟨*distance*⟩`]{`⟨*text*⟩`}`

The `\epigraphhead` macro stores ⟨*text*⟩ for printing at ⟨*distance*⟩ below the header on a page. ⟨*text*⟩ can be ordinary text or, more likely, can be either an `\epigraph` command or an `epigraphs` environment. By default, the epigraph will be typeset at the righthand margin. If the command is immediately preceded by a `\chapter` or `\chapter*` command, the epigraph is typeset on the chapter title page.

The default value for the optional ⟨*distance*⟩ argument is set so that an `\epigraph` consisting of a single line of quotation and a single line denoting the source is aligned with

the bottom of the 'Chapter X' line produced by the `\chapter` command using the *default* chapterstyle. In other cases you will have to experiment with the ⟨*distance*⟩ value. The value for ⟨*distance*⟩ can be either a integer or a real number. The units are in terms of the current value for `\unitlength`. A typical value for ⟨*distance*⟩ for a single line quotation and source for a `\chapter*` might be about 70 (points). A positive value of ⟨*distance*⟩ places the epigraph below the page heading and a negative value will raise it above the page heading.

Here's some example code:

```
\chapter*{Celestial navigation}
\epigraphhead[70]{\epigraph{Star crossed lovers.}{\textit{The Bard}}}
```

The ⟨*text*⟩ argument is put into a minipage of width `\epigraphwidth`. If you use something other than `\epigraph` or `epigraphs` for the ⟨*text*⟩ argument, you may have to do some positioning of the text yourself so that it is properly located in the minipage. For example

```
\chapter{Short}
\renewcommand{\epigraphflush}{center}
\epigraphhead{\centerline{Short quote}}
```

The `\epigraphhead` command changes the page style for the page on which it is specified, so there should be no text between the `\chapter` and the `\epigraphhead` commands. The page style is identical to the *plain* page style except for the inclusion of the epigraph. If you want a more fancy style for epigraphed chapters you will have to do some work yourself.

`\epigraphforheader[`⟨*distance*⟩`]{`⟨*text*⟩`}`
`\epigraphpicture`

The `\epigraphforheader` macro takes the same arguments as `\epigraphhead` but puts ⟨*text*⟩ into a zero-sized picture at the coordinate position `(0,-<distance>)`; the macro `\epigraphpicture` holds the resulting picture. This can then be used as part of a chapter pagestyle, as in

```
\makepagestyle{mychapterpagestyle}
...
\makeoddhead{mychapterpagestyle}{}{}{\epigraphpicture}
```

Of course the ⟨*text*⟩ argument for `\epigraphforheader` need not be an `\epigraph`, it can be arbitrary text.

`\dropchapter{`⟨*length*⟩`}`
`\undodrop`

If a long epigraph is placed before a chapter title it is possible that the bottom of the epigraph may interfere with the chapter title. The command `\dropchapter` will lower any subsequent chapter titles by ⟨*length*⟩; a negative ⟨*length*⟩ will raise the titles. The command `\undodrop` restores subsequent chapter titles to their default positions. For example:

```
\dropchapter{2in}
\chapter{Title}
\epigraphhead{long epigraph}
\undodrop
```

`\cleartoevenpage[`⟨*text*⟩`]`

On occasions it may be desirable to put something (e.g., an epigraph, a map, a picture) on the page facing the start of a chapter, where the something belongs to the chapter that is about to start rather than the chapter that has just ended. In order to do this in a document that is going to be printed doublesided, the chapter must start on an odd numbered page and the pre-chapter material put on the immediately preceding even numbered page. The `\cleartoevenpage` command is like `\cleardoublepage` except that the page following the command will be an even numbered page, and the command takes an optional argument which is applied to the skipped page (if any).

Here is an example:

```
... end previous chapter.
\cleartoevenpage
\begin{center}
\begin{picture}... \end{picture}
\end{center}
\chapter{Next chapter}
```

If the style is such that chapter headings are put at the top of the pages, then it would be advisable to include `\thispagestyle{empty}` (or perhaps `plain`) immediately after `\cleartoevenpage` to avoid a heading related to the previous chapter from appearing on the page.

If the something is like a figure with a numbered caption and the numbering depends on the chapter numbering, then the numbers have to be hand set (unless you define a special chapter command for the purpose). For example:

```
... end previous chapter.
\cleartoevenpage[\thispagestyle{empty}] % a skipped page to be empty
\thispagestyle{plain}
\addtocounter{chapter}{1} % increment the chapter number
\setcounter{figure}{0}    % initialise figure counter
\begin{figure}
...
\caption{Pre chapter figure}
\end{figure}

\addtocounter{chapter}{-1} % decrement the chapter number
\chapter{Next chapter}     % increments chapter & resets figure numbers
\addtocounter{figure}{1}   % to account for pre-chapter figure
```

13.3.1 Epigraphs on book or part pages

If you wish to put an epigraphs on `\book` or `\part` pages you have to do a little more work than in other cases. This is because these division commands do some page flipping before and after typesetting the title.

One method is to put the epigraph into the page header as for epigraphs before `\chapter` titles. By suitable adjustments the epigraph can be placed anywhere on the page, independently of whatever else is on the page. A similar scheme may be used

for epigraphs on other kinds of pages. The essential trick is to make sure that the *epigraph* pagestyle is used for the page. For an epigraphed bibliography or index, the macros `\prebibhook` or `\preindexhook` can be appropriately modified to do this.

The other method is to subvert the `\beforepartskip` command for epigraphs before the title, or the `\afterpartskip` command for epigraphs after the title (or the equivalents for `\book` pages).

For example:

```
\let\oldbeforepartskip\beforepartskip % save definition
\renewcommand*{\beforepartskip}{%
  \epigraph{...}{...}% an epigraph
  \vfil}
\part{An epigraphed part}
...
\renewcommand*{\beforepartskip}{%
  \epigraph{...}{...}% another epigraph
  \vfil}
\part{A different epigraphed part}
...
\let\beforepartskip\oldbeforepartskip % restore definition
\part{An unepigraphed part}
...
```

Fourteen

Poetry

The typesetting of a poem should ideally be dependent on the particular poem. Individual problems do not usually admit of a general solution, so this manual and code should be used more as a guide towards some solutions rather than providing a ready made solution for any particular piece of verse.

The doggerel used as illustrative material has been taken from [Wil??].

Note that for the examples in this section I have made no attempt to do other than use the minimal (La)TeX capabilities; in particular I have made no attempt to do any special page breaking so some stanzas may cross onto the next page — most undesireable for publication.

The standard LaTeX classes provide the `verse` environment which is defined as a particular kind of list. Within the environment you use `\\` to end a line, and a blank line will end a stanza. For example, here is the text of a single stanza poem:

```
\newcommand{\garden}{
I used to love my garden \\
But now my love is dead \\
For I found a bachelor's button \\
In black-eyed Susan's bed.
}
```

When this is typeset as a normal LaTeX paragraph (with no paragraph indentation), i.e.,

```
\noident\garden
```

it looks like:

I used to love my garden
But now my love is dead
For I found a bachelor's button
In black-eyed Susan's bed.

Typesetting it within the `verse` environment produces:

> I used to love my garden
> But now my love is dead
> For I found a bachelor's button
> In black-eyed Susan's bed.

The stanza could also be typeset within the `alltt` environment, defined in the standard alltt package [Bra97], using a normal font and no `\\` line endings.

```
\begin{alltt}\normalfont
I used to love my garden
But now my love is dead
For I found a bachelor's button
In black-eyed Susan's bed.
\end{alltt}
```

which produces:

I used to love my garden
But now my love is dead
For I found a bachelor's button
In black-eyed Susan's bed.

The `alltt` environment is like the `verbatim` environment except that you can use LaTeX macros inside it. In the `verse` environment long lines will be wrapped and indented but in the `alltt` environment there is no indentation.

Some stanzas have certain lines indented, often alternate ones. To typeset stanzas like this you have to add your own spacing. For instance:

```
\begin{verse}
There was an old party of Lyme \\
Who married three wives at one time. \\
\hspace{2em} When asked: 'Why the third?' \\
\hspace{2em} He replied: 'One's absurd, \\
And bigamy, sir, is a crime.'
\end{verse}
```

is typeset as:

> There was an old party of Lyme
> Who married three wives at one time.
> When asked: 'Why the third?'
> He replied: 'One's absurd,
> And bigamy, sir, is a crime.'

Using the `alltt` environment you can put in the spacing via ordinary spaces. That is, this:

```
\begin{alltt}\normalfont
There was an old party of Lyme
Who married three wives at one time.
      When asked: 'Why the third?'
      He replied: 'One's absurd,
And bigamy, sir, is a crime.'
\end{alltt}
```

is typeset as

There was an old party of Lyme

Who married three wives at one time.
When asked: 'Why the third?'
He replied: 'One's absurd,
And bigamy, sir, is a crime.'

More exotically you could use the TeX `\parshape` command[1]:

```
\parshape = 5 0pt \linewidth 0pt \linewidth
              2em \linewidth 2em \linewidth 0pt \linewidth
\noindent There was an old party of Lyme \\
Who married three wives at one time. \\
When asked: 'Why the third?' \\
He replied: 'One's absurd, \\
And bigamy, sir, is a crime.' \par
```

which will be typeset as:

There was an old party of Lyme
Who married three wives at one time.
When asked: 'Why the third?'
He replied: 'One's absurd,
And bigamy, sir, is a crime.'

This is about as much assistance as standard (La)TeX provides, except to note that in the `verse` environment the `\\*` version of `\\` will prevent a following page break. You can also make judicious use of the `\needspace` macro to keep things together.

Some books of poetry, and especially anthologies, have two or more indexes, one, say for the poem titles and another for the first lines, and maybe even a third for the poets' names. If you are not using memoir then the index [Jon95] and multind [Lon91] packages provide support for multiple indexes in one document.

14.1 Classy verse

The code provided by the memoir class is meant to help with some aspects of typesetting poetry but does not, and cannot, provide a comprehensive solution to all the requirements that will arise.

The main aspects of typesetting poetry that differ from typesetting plain text are:

- Poems are usually visually centered on the page.
- Some lines are indented, and often there is a pattern to the indentation.
- When a line is too wide for the page it is broken and the remaining portion indented with respect to the original start of the line.

These are the ones that the class attempts to deal with.

`\begin{verse}[`⟨*length*⟩`] ... \end{verse}`
`\versewidth`

1 See the *TeXbook* for how to use this.

The `verse` environment provided by the class is an extension of the usual LaTeX environment. The environment takes one optional parameter, which is a length; for example `\begin{verse}[4em]`. You may have noticed that the earlier verse examples are all near the left margin, whereas verses usually look better if they are typeset about the center of the page. The length parameter, if given, should be about the length of an average line, and then the entire contents will be typeset with the mid point of the length centered horizontally on the page.

The length `\versewidth` is provided as a convenience. It may be used, for example, to calculate the length of a line of text for use as the optional argument to the `verse` environment:

```
\settowidth{\versewidth}{This is the average line,}
\begin{verse}[\versewidth]
```

`\vleftmargin`

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

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

If necessary the poem could even be moved into the left margin by giving `\vleftmargin` a negative length value, such as -1.5em.

`\stanzaskip`

The vertical space between stanzas is the length `\stanzaskip`. It can be changed by the usual methods.

`\vin`
`\vgap`
`\vindent`

The command `\vin` is shorthand for `\hspace*{\vgap}` for use at the start of an indented line of verse. The length `\vgap` (initially 1.5em) can be changed by `\setlength` or `\addtolength`. When a verse line is too long to fit within the typeblock it is wrapped to the next line with an initial indent given by the value of the length `vindent`. Its initial value is twice the default value of `\vgap`.

`\\[`⟨*length*⟩`]`
`\\*[`⟨*length*⟩`]`
`\\![`⟨*length*⟩`]`

Each line in the `verse` environment, except possibly for the last line in a stanza, must be ended by `\\`, which comes in several variants. In each variant the optional ⟨*length*⟩ is the vertical space to be left before the next line. The `\\*` form prohibits a page break after the line. The `\\!` form is to be used only for the last line in a stanza when the lines are being numbered; this is because the line numbers are incremented by the `\\` macro. It would normally be followed by a blank line.

`\verselinebreak[`⟨*length*⟩`]`
`\\>[`⟨*length*⟩`]`

Using `\verselinebreak` will cause later text in the line to be typeset indented on the following line. If the optional ⟨*length*⟩ is not given the indentation is twice `\vgap`, otherwise it is ⟨*length*⟩. The broken line will count as a single line as far as the `altverse` and `patverse` environments are concerned. The macro `\\>` is shorthand for `\verselinebreak`, and unlike other members of the `\\` family the optional ⟨*length*⟩ is the indentation of the following partial line, not a vertical skip. Also, the `\\>` macro does not increment any line number.

`\vinphantom{`⟨*text*⟩`}`

Verse lines are sometimes indented according to the space taken by the text on the previous line. The macro `\vinphantom` can be used at the start of a line to give an indentation as though the line started with ⟨*text*⟩. For example here are a few lines from the portion of *Fridthjof's Saga* where Fridthjof and Ingeborg part:

Source for example 14.1

```
\settowidth{\versewidth}{Nay, nay, I leave thee not,
                         thou goest too}
\begin{verse}[\versewidth]
\ldots \\*
His judgement rendered, he dissolved the Thing. \\*
\flagverse{Ingeborg} And your decision? \\*
\flagverse{Fridthjof} \vinphantom{And your decision?}
                      Have I ought to choose? \\*
Is not mine honour bound by his decree? \\*
And that I will redeem through Angantyr \\*
His paltry gold doth hide in Nastrand's flood. \\*
Today will I depart. \\*
\flagverse{Ingeborg} \vinphantom{Today will I depart.}
                     And Ingeborg leave? \\*
\flagverse{Fridthjof} Nay, nay, I leave thee not,
                      thou goest too. \\*
\flagverse{Ingeborg} Impossible! \\*
\flagverse{Fridthjof} \vinphantom{Impossible!}
                      O! hear me, ere thou answerest.
\end{verse}
```

Use of `\vinphantom` is not restricted to the start of verse lines — it may be used anywhere in text to leave some some blank space. For example, compare the two lines below, which are produced by this code:

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

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

`\vleftofline{`⟨*text*⟩`}`

Typeset example 14.1: Phantom text in verse

...
His judgement rendered, he dissolved the Thing.
Ingeborg And your decision?
Fridthjof Have I ought to choose?
Is not mine honour bound by his decree?
And that I will redeem through Angantyr
His paltry gold doth hide in Nastrand's flood.
Today will I depart.
Ingeborg And Ingeborg leave?
Fridthjof Nay, nay, I leave thee not, thou goest too.
Ingeborg Impossible!
Fridthjof O! hear me, ere thou answerest.

A verse line may start with something, for example open quote marks, where it is desireable that it is ignored as far as the alignment of the rest of the line is concerned[2] — a sort of 'hanging left punctuation'. When it is put at the start of a line in the `verse` environment the ⟨*text*⟩ is typeset but ignored as far as horizontal indentation is concerned. Compare the two examples.

Source for example 14.2

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

Source for example 14.3

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

2 Requested by Matthew Ford who also provided the example text.

Typeset example 14.2: Verse with regular quote marks

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

Typeset example 14.3: Verse with hanging left quote marks

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

```
And everyone who calls \ldots ’’\,’
\end{verse}
```

14.1.1 Indented lines

Within the `verse` environment stanzas are normally separated by a blank line in the input.

`\begin{altverse}` ... `\end{altverse}`

Individual stanzas within `verse` may, however, be enclosed in the `altverse` environment. This has the effect of indenting the 2nd, 4th, etc., lines of the stanza by the length `\vgap`.

`\begin{patverse}` ... `\end{patverse}`
`\begin{patverse*}` ... `\end{patverse*}`
`\indentpattern{`⟨*digits*⟩`}`

As an alternative to the `altverse` environment, individual stanzas within the `verse` environment may be enclosed in the `patverse` environment. Within this environment the indentation of each line is specified by an indentation pattern, which consists of an array of digits, d_1 to d_n, and the nth line is indented by d_n times `\vgap`. However, the first line is not indented, irrespective of the value of d_1.

The indentation pattern for a `patverse` or `patverse*` environment is specified via the `\indentpattern` command, where ⟨*digits*⟩ is a string of digits (e.g., 3213245281).

With the `patverse` environment, if the pattern is shorter than the number of lines in the stanza, the trailing lines will not be indented. However, in the `patverse*` environment the pattern keeps repeating until the end of the stanza.

14.1.2 Numbering

```
\flagverse{⟨flag⟩}
\vleftskip
```

Putting `\flagverse` at the start of a line will typeset ⟨*flag*⟩, for example the stanza number, ending at a distance `\vleftskip` before the line. The default for `\vleftskip` is 3em.

The lines in a poem may be numbered.

```
\linenumberfrequency{⟨nth⟩}
\setverselinenums{⟨first⟩}{⟨startat⟩}
```

The declaration `\linenumberfrequency{`⟨*nth*⟩`}` will cause every ⟨*nth*⟩ line of succeeding verses to be numbered. For example, `\linenumberfrequency{5}` will number every fifth line. The default is `\linenumberfrequency{0}` which prevents any numbering. The `\setverselinenums` macro can be used to specify that the number of the first line of the following `verse` shall be ⟨*first*⟩ and the first printed number shall be ⟨*startat*⟩. For example, perhaps you are quoting part of a numbered poem. The original numbers every tenth line but if your extract starts with line 7, then

```
\linenumberfrequency{10}
\setverselinenums{7}{10}
```

is what you will need.

```
\thepoemline
\linenumberfont{⟨font-decl⟩}
```

The `poemline` counter is used in numbering the lines, so the number representation is `\thepoemline`, which defaults to arabic numerals, and they are typeset using the font specified via `\linenumberfont`; the default is

```
\linenumberfont{\small\rmfamily}
```

for small numbers in the roman font.

```
\verselinenumbersright
\verselinenumbersleft
\vrightskip
```

Following the declaration `\verselinenumbersright`, which is the default, any verse line numbers will be set in the righthand margin. The `\verselinenumbersleft` declaration will set any subsequent line numbers to the left of the lines. The numbers are set at a distance `\vrightskip` (default 1em) into the margin.

14.2 TITLES

The `\PoemTitle` command is provided for typesetting titles of poems.

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

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

\poemtoc{⟨*sec*⟩}

The kind of entry made in the ToC by \PoemTitle is defined by \poemtoc. The initial definition is:

```
\newcommand{\poemtoc}{section}
```

for a section-like ToC entry. This can be changed to, say, chapter or subsection or ....

\poemtitlemark{⟨*forhead*⟩} \poemtitlepstyle

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

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

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

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

14.2.1 Main Poem Title layout parameters

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

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

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

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

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

The elements are detailed in the next section.

14.2.2 Detailed Poem Title layout parameters

\beforePoemTitleskip
\PoemTitlenumfont
\midPoemTitleskip
\PoemTitlefont
\afterPoemTitleskip

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

```
\newcommand*{\PoemTitleheadstart}{\vspace{\beforePoemTitleskip}}
\newlength{\beforePoemTitleskip}
  \setlength{\beforePoemTitleskip}{1\onelineskip}
```

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

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

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

```
\newcommand*{\printPoemTitlenonum}{}
```

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

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

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

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

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

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

14.3 EXAMPLES

Here are some sample verses using the class facilities.

First a Limerick, but titled and centered:

```
\renewcommand{\poemtoc}{subsection}
\PlainPoemTitle
\PoemTitle{A Limerick}
\settowidth{\versewidth}{There was a young man of Quebec}
\begin{verse}[\versewidth]
There was a young man of Quebec \\
Who was frozen in snow to his neck. \\
\vin When asked: ‘Are you friz?’ \\
\vin He replied: ‘Yes, I is, \\
But we don’t call this cold in Quebec.’
\end{verse}
```

which gets typeset as below. The `\poemtoc` is redefined to `subsection` so that the `\poemtitle` titles are entered into the ToC as subsections. The titles will not be numbered because of the `\PlainPoemTitle` declaration.

A Limerick

There was a young man of Quebec
Who was frozen in snow to his neck.
    When asked: ‘Are you friz?’
    He replied: ‘Yes, I is,
But we don’t call this cold in Quebec.’

Next is the Garden verse within the `altverse` environment. Unlike earlier renditions this one is titled and centered.

```
\settowidth{\versewidth}{But now my love is dead}
\PoemTitle{Love’s lost}
\begin{verse}[\versewidth]
\begin{altverse}
\garden
\end{altverse}
\end{verse}
```

Note how the alternate lines are automatically indented in the typeset result below.

Love’s lost

I used to love my garden
    But now my love is dead
For I found a bachelor’s button
    In black-eyed Susan’s bed.

It is left up to you how you might want to add information about the author of a poem. Here is one example of a macro for this:

```
\newcommand{\attrib}[1]{%
   \nopagebreak{\raggedleft\footnotesize #1\par}}
```

This can be used as in the next bit of doggerel.

```
\PoemTitle{Fleas}
\settowidth{\versewidth}{What a funny thing is a flea}
\begin{verse}[\versewidth]
What a funny thing is a flea. \\
You can't tell a he from a she. \\
But he can. And she can. \\
Whoopee!
\end{verse}
\attrib{Anonymous}
```

Fleas

What a funny thing is a flea.
You can't tell a he from a she.
But he can. And she can.
Whoopee!

Anonymous

The next example demonstrates the automatic line wrapping for overlong lines.

```
\PoemTitle{In the beginning}
\settowidth{\versewidth}{And objects at rest tended to
                         remain at rest}
\begin{verse}[\versewidth]
Then God created Newton, \\
And objects at rest tended to remain at rest, \\
And objects in motion tended to remain in motion, \\
And energy was conserved
   and momentum was conserved
   and matter was conserved \\
And God saw that it was conservative.
\end{verse}
\attrib{Possibly from \textit{Analog}, circa 1950}
```

In the beginning

Then God created Newton,
And objects at rest tended to remain at rest,
And objects in motion tended to remain in motion,

And energy was conserved and momentum was conserved and
matter was conserved
And God saw that it was conservative.

Possibly from *Analog*, circa 1950

The following verse demonstrates the use of a forced linebreak; I have used the `\\>` command instead of the more descriptive, but discursive, `\verselinebreak`. It also has a slightly different title style.

```
\renewcommand{\PoemTitlefont}{%
              \normalfont\large\itshape\centering}
\poemtitle{Mathematics}
\settowidth{\versewidth}{Than Tycho Brahe, or Erra Pater:}
\begin{verse}[\versewidth]
In mathematics he was greater \\
Than Tycho Brahe, or Erra Pater: \\
For he, by geometric scale, \\
Could take the size of pots of ale;\\
\settowidth{\versewidth}{Resolve by}%
Resolve, by sines \\>[\versewidth] and tangents straight, \\
If bread or butter wanted weight; \\
And wisely tell what hour o' the day \\
The clock does strike, by Algebra.
\end{verse}
\attrib{Samuel Butler (1612--1680)}
```

Mathematics

In mathematics he was greater
Than Tycho Brahe, or Erra Pater:
For he, by geometric scale,
Could take the size of pots of ale;
Resolve, by sines
and tangents straight,
If bread or butter wanted weight;
And wisely tell what hour o' the day
The clock does strike, by Algebra.

Samuel Butler (1612–1680)

Another limerick, but this time taking advantage of the `patverse` environment. If you are typesetting a series of limericks a single `\indentpattern` will do for all of them.

```
\settowidth{\versewidth}{There was a young lady of Ryde}
\indentpattern{00110}
\needspace{7\onelineskip}
\PoemTitle{The Young Lady of Ryde}
\begin{verse}[\versewidth]
\begin{patverse}
```

```
There was a young lady of Ryde \\
Who ate some apples and died. \\
The apples fermented \\
Inside the lamented \\
And made cider inside her inside.
\end{patverse}
\end{verse}
```

Note that I used the `\needspace` command to ensure that the limerick will not get broken across a page.

The Young Lady of Ryde

There was a young lady of Ryde
Who ate some apples and died.
The apples fermented
Inside the lamented
And made cider inside her inside.

The next example is a song you may have heard of. This uses `\flagverse` for labelling the stanzas, and because the lines are numbered they can be referred to.

```
\settowidth{\versewidth}{In a cavern, in a canyon,}
\PoemTitle{Clementine}
\begin{verse}[\versewidth]
\linenumberfrequency{2}
\begin{altverse}
\flagverse{1.} In a cavern, in a canyon, \\
Excavating for a mine, \\
Lived a miner, forty-niner, \label{vs:49} \\
And his daughter, Clementine. \\!
\end{altverse}

\begin{altverse}
\flagverse{\textsc{chorus}} Oh my darling, Oh my darling, \\
Oh my darling Clementine. \\
Thou art lost and gone forever, \\
Oh my darling Clementine.
\end{altverse}
\linenumberfrequency{0}
\end{verse}
The ‘forty-niner’ in line~\ref{vs:49} of the song
refers to the gold rush of 1849.
```

Clementine

1. In a cavern, in a canyon,

Excavating for a mine,
Lived a miner, forty-niner,
And his daughter, Clementine.

CHORUS Oh my darling, Oh my darling,
Oh my darling Clementine.
Thou art lost and gone forever,
Oh my darling Clementine.

The 'forty-niner' in line 3 of the song refers to the gold rush of 1849.

The last example is a much more ambitious use of `\indentpattern`. In this case it is defined as:

```
\indentpattern{0135554322112346898779775545653222345544456688778899}
```

and the result is shown on the next page.

Mouse's Tale

Fury said to
a mouse, That
he met
in the
house,
'Let us
both go
to law:
I will
prosecute
you. —
Come, I'll
take no
denial;
We must
have a
trial:
For
really
this
morning
I've
nothing
to do.'
Said the
mouse to
the cur,
Such a
trial,
dear sir,
With no
jury or
judge,
would be
wasting
our breath.'
'I'll be
judge,
I'll be
jury.'
Said
cunning
old Fury;
'I'll try
the whole
cause
and
condemn
you
to
death.'

Lewis Carrol, *Alice's Adventures in Wonderland*, 1865

Fifteen

Boxes, verbatims and files

The title of this chapter indicates that it deals with three disconnected topics, but there is method in the seeming peculiarity. By the end of the chapter you will be able to write LaTeX code that lets you put things in your document source at one place and have them typeset at a different place, or places. For example, if you are writing a text book that includes questions and answers then you could write a question and answer together yet have the answer typeset at the end of the book.

Writing in one place and printing in another is based on outputting stuff to a file and then inputting it for processing at another place or time. This is just how LaTeX produces the ToC. It is often important when writing to a file that LaTeX does no processing of any macros, which implies that we need to be able to write verbatim. One use of verbatim in LaTeX is to typeset computer code or the like, and to clearly distinguish the code from the main text it is often typeset within a box. Hence the chapter title.

The class extends the kinds of boxes normally provided, extends the default verbatims, and provides a simple means of writing and reading files.

One problem with verbatims is that they can not be used as part of an argument to a command. For example to typeset something in a framed `minipage` the obvious way is to use the `minipage` as the argument to the `\fbox` macro:

```
\fbox{\begin{minipage}{6cm}
      Contents of framed minipage
      \end{minipage}}
```

This works perfectly well until the contents includes some verbatim material, whereupon you will get nasty error messages. However this particular conundrum is solvable, even if the solution is not particularly obvious. Here it is.

We can put things into a box, declared via `\newsavebox`, and typeset the contents of the box later via `\usebox`. The most common way of putting things into a save box is by the `\sbox` or `\savebox` macros, but as the material for saving is one of the arguments to these macros this approach fails. But, `lrbox` is an environment form of `\sbox`, so it can handle verbatim material. The code below, after getting a new save box, defines a new `framedminipage` environment which is used just like the standard `minipage`. The `framedminipage` starts an `lrbox` environment and then starts a `minipage` environment, after which comes the contents. At the end it closes the two environments and calls `\fbox` with its argument being the contents of the saved box *which have already been typeset*.

```
\newsavebox{\minibox}
\newenvironment{framedminipage}[1]{%
  \begin{lrbox}{\minibox}\begin{minipage}{#1}}%
```

```
{\end{minipage}\end{lrbox}\fbox{\usebox{\minibox}}}
```

Question 1. Can you think of any improvements to the definition of the `framedminipage` environment?

Question 2. An answer to question 1 is at the end of this chapter. Suggest how it was put there.

15.1 BOXES

LaTeX provides some commands to put a box round some text. The class extends the available kinds of boxes.

```
\begin{framed} text \end{framed}
\begin{shaded} text \end{shaded}
\begin{snugshade} text \end{snugshade}
```

The `framed`, `shaded`, and `snugshade` environments, which were created by Donald Arseneau as part of his framed package [Ars07], put their contents into boxes that break across pages. The `framed` environment delineates the box by drawing a rectangular frame. If there is a pagebreak in the middle of the box, frames are drawn on both pages.

The `shaded` environment typesets the box with a shaded or colored background. This requires the use of the color package [Car05], which is one of the required LaTeX packages, or the xcolor package [Ker07]. The shading color is `shadecolor`, which you have to define before using the environment. For example, to have a light gray background:

```
\definecolor{shadecolor}{gray}{0.9}
```

For complete information on this see the documentation for the color or xcolor packages, or one of the LaTeX books like the *Graphics Companion* [GM+07]. In the `snugshaded` environment the box clings more closely to its contents than it does in the `shaded` environment.

Recommended alternative

Since the class was originally written, much have happened in the gfx generating capabilities in LaTeX, especially the popularity of TikZ has provided many more extensive box and graphics generating packages.

As of 2018 one of the most impressive packages for all sorts of boxes is the tcolorbox package by Thomas F. Sturm.

Be aware that the boxes we present in this manual are somewhat delicate; they do not work in all circumstances. For example they will not work with the multicol package [Mit18], and any floats or footnotes in the boxes will disappear.

```
\FrameRule \FrameSep \FrameHeightAdjust
```

The `framed` environment puts the text into an '`\fbox`' with the settings:

```
\setlength{\FrameRule}{\fboxrule}
\setlength{\FrameSep}{3\fboxsep}
```

The macro \FrameHeightAdjust specifies the height of the top of the frame above the baseline at the top of a page; its initial definition is:

```
\providecommand*{\FrameHeightAdjust}{0.6em}
```

\MakeFramed{⟨*settings*⟩} \endMakeFramed \FrameCommand \FrameRestore

Internally, the environments are specified using the MakeFramed environment. The ⟨*setting*⟩ should contain any adjustments to the text width (applied to \hsize and using the \width of the frame itself) and a 'restore' command, which is normally the provided \FrameRestore macro. The frame itself is drawn via the \FrameCommand, which can be changed to obtain other boxing styles. The default definition equates to an \fbox and is:

```
\newcommand*{\FrameCommand}{%
  \setlength{\fboxrule}{\FrameRule}\setlength{\fboxsep}{\FrameSep}%
  \fbox}
```

For example, the framed, shaded and snugshade environments are defined as

```
\newenvironment{framed}{% % uses default \FrameCommand
  \MakeFramed{\advance\hsize -\width \FrameRestore}}%
  {\endMakeFramed}
\newenvironment{shaded}{% % redefines \FrameCommand as \colorbox
  \def\FrameCommand{\fboxsep=\FrameSep \colorbox{shadecolor}}%
  \MakeFramed{\FrameRestore}}%
  {\endMakeFramed}
\newenvironment{snugshade}{% A tight version of shaded
  \def\FrameCommand{\colorbox{shadecolor}}%
  \MakeFramed{\FrameRestore\@setminipage}}%
  {\par\unskip\endMakeFramed}
```

If you wanted a narrow, centered, framed environment you could do something like this:

```
\newenvironment{narrowframed}{%
  \MakeFramed{\setlength{\hsize}{22pc}\FrameRestore}}%
  {\endMakeFramed}
```

where 22pc will be the width of the new framed environment.

\begin{leftbar} text \end{leftbar}

The leftbar environment draws a thick vertical line at the left of the text. It is defined as

```
\newenvironment{leftbar}{%
  \def\FrameCommand{\vrule width 3pt \hspace{10pt}}%
  \MakeFramed{\advance\hsize -\width \FrameRestore}}%
  {\endMakeFramed}
```

By changing the ⟨*setting*⟩ for \MakeFramed and the definition of \FrameCommand you can obtain a variety of framing styles. For instance, to have rounded corners to the frame instead of the normal sharp ones, you can use the fancybox package [Zan98] and the following code:

```
\usepackage{fancybox}
\newenvironment{roundedframe}{%
  \def\FrameCommand{%
    \cornersize*{20pt}%
    \setlength{\fboxsep}{5pt}%
    \ovalbox}%
  \MakeFramed{\advance\hsize-\width \FrameRestore}}%
  {\endMakeFramed}
```

A framed environment is normally used to distinguish its contents from the surrounding text. A title for the environment may be useful, and if there was a pagebreak in the middle, a title on the continuation could be desireable. Doing this takes a bit more work than I have shown so far. This first part was inspired by a posting to CTT by Donald Arseneau.[1].

```
\newcommand{\FrameTitle}[2]{%
  \fboxrule=\FrameRule \fboxsep=\FrameSep
  \fbox{\vbox{\nobreak \vskip -0.7\FrameSep
    \rlap{\strut#1}\nobreak\nointerlineskip% left justified
    \vskip 0.7\FrameSep
    \hbox{#2}}}}
\newenvironment{framewithtitle}[2][\FrameFirst@Lab\ (cont.)]{%
  \def\FrameFirst@Lab{\textbf{#2}}%
  \def\FrameCont@Lab{\textbf{#1}}%
  \def\FrameCommand##1{%
    \FrameTitle{\FrameFirst@Lab}{##1}}%
  \def\FirstFrameCommand##1{%
    \FrameTitle{\FrameFirst@Lab}{##1}}%
  \def\MidFrameCommand##1{%
    \FrameTitle{\FrameCont@Lab}{##1}}%
  \def\LastFrameCommand##1{%
    \FrameTitle{\FrameCont@Lab}{##1}}%
  \MakeFramed{\advance\hsize-\width \FrameRestore}}%
  {\endMakeFramed}
```

The `framewithtitle` environment, which is the end goal of this exercise, acts like the `framed` environment except that it puts a left-justified title just after the top of the frame box and before the regular contents.

```
\begin{framewithtitle}[⟨cont-title⟩]{⟨title⟩} text
\end{framewithtitle}
```

The ⟨*title*⟩ is set in a bold font. If the optional ⟨*cont-title*⟩ argument is given then ⟨*cont-title*⟩ is used as the title on any suceeding pages, otherwise the phrase '⟨*title*⟩ (cont.)' is used for the continuation title.

If you would like the titles centered, replace the line marked 'left justified' in the code for `\FrameTitle` with the line:

1 On 2003/10/24 in the thread *framed.sty w/heading?*. The particulars are no longer applicable as the framing code in question then has since been revised.

```
\rlap{\centerline{\strut#1}}\nobreak\nointerlineskip% centered
```

The code for the `frametitle` environment is not obvious. The difficulty in creating the environment was that the underlying framing code goes through the 'stuff' to be framed by first trying to fit it all onto one page (`\FrameCommand`). If it does not fit, then it takes as much as will fit and typesets that using `\FirstFrameCommand`, then tries to typeset the remainder on the next page. If it all fits then it uses `\LastFrameCommand`. If it doesn't fit, it typesets as much as it can using `\MidFrameCommand`, and then tries to set the remainder on the following page. The process repeats until all has been set.

If you would prefer to have the title at the top outside the frame the above code needs adjusting.

```
\newcommand{\TitleFrame}[2]{%
  \fboxrule=\FrameRule \fboxsep=\FrameSep
  \vbox{\nobreak \vskip -0.7\FrameSep
    \rlap{\strut#1}\nobreak\nointerlineskip% left justified
    \vskip 0.7\FrameSep
    \noindent\fbox{#2}}}
\newenvironment{titledframe}[2][\FrameFirst@Lab\ (cont.)]{%
  \def\FrameFirst@Lab{\textbf{#2}}%
  \def\FrameCont@Lab{\textbf{#1}}%
  \def\FrameCommand##1{%
    \TitleFrame{\FrameFirst@Lab}{##1}}
  \def\FirstFrameCommand##1{%
    \TitleFrame{\FrameFirst@Lab}{##1}}
  \def\MidFrameCommand##1{%
    \TitleFrame{\FrameCont@Lab}{##1}}
  \def\LastFrameCommand##1{%
    \TitleFrame{\FrameCont@Lab}{##1}}
  \MakeFramed{\hsize\textwidth
              \advance\hsize -2\FrameRule
              \advance\hsize -2\FrameSep
              \FrameRestore}}%
  {\endMakeFramed}
```

`\begin{titledframe}[`⟨*cont-title*⟩`]{`⟨*title*⟩`}` text `\end{titledframe}`

The `titledframe` environment is identical to `framewithtitle` except that the title is placed just before the frame. Again, if you would like a centered title, replace the line marked 'left justified' in `\TitleFrame` by

```
\rlap{\centerline{\strut#1}}\nobreak\nointerlineskip% centered
```

You can adjust the code for the `framewithtitle` and `titledframe` environments to suit your own purposes, especially as they are not part of the class so you would have to type them in yourself anyway if you wanted to use them, using whatever names you felt suitable.

The class provides two further environments in addition to those from the framed package.

```
\begin{qframe} text \end{qframe}
\begin{qshade} text \end{qshade}
```

When used within, say, a `quotation` environment, the `framed` and `shaded` environments do not closely box the indented text. The `qframe` and `qshade` environments do provide close boxing.[2] The difference can be seen in the following `quotation`.

> This is the start of a `quotation` environment. It forms the basis showing the difference between the `framed` and `qframe` environments.
>
> This is the second paragraph in the `quotation` environment and in turn it is within the `qframe` environment.
>
> This is the third paragraph in the `quotation` environment and in turn it is within the `framed` environment.
>
> This is the fourth and final paragraph within the `quotation` environment and is not within either a `qfame` or `framed` environment.

If you want to put a frame inside an `adjustwidth` environment then you may well find that `qframe` or `qshade` meet your expections better than `framed` of `shaded`. Of course, it does depend on what your expectations are.

15.2 LONG COMMENTS

The % comment character can be used to comment out (part of) a line of TeX code, but this gets tedious if you need to comment out long chunks of code.

```
\begin{comment} text to be skipped over \end{comment}
```

As an extreme form of font changing, although it doesn't actually work that way, anything in a `comment` environment will not appear in the document; effectively, LaTeX throws it all away. This can be useful to temporarily discard chunks of stuff instead of having to mark each line with the % comment character.

```
\newcomment{⟨name⟩}
\commentsoff{⟨name⟩}
\commentson{⟨name⟩}
```

The class lets you define your own comment environment via the `\newcomment` command which defines a comment environment called ⟨*name*⟩. In fact the class itself uses `\newcomment{comment}` to define the `comment` environment. A comment environment ⟨*name*⟩ may be switched off so that its contents are not ignored by using the `\commentsoff` declaration. It may be switched on later by the `\commentson` declaration. In either case ⟨*name*⟩ must have been previously declared as a comment environment via `\newcomment`.

Suppose, for example, that you are preparing a draft document for review by some others and you want to include some notes for the reviewers. Also, you want to include some private comments in the source for yourself. You could use the `comment` environment for your private comments and create another environment for the notes to the reviewers. These notes should not appear in the final document. Your source might then look like:

2 Donald Arseneau has said that he may put something similar in a later version the `framed` package.

```
\newcomment{review}
\ifdraftdoc\else
  \commentsoff{review}
\fi
...
\begin{comment}
Remember to finagle the wingle!
\end{comment}
...
\begin{review}
\textit{REVIEWERS: Please pay particular attention to this section.}
\end{review}
...
```

Comment environments cannot be nested, nor can they overlap. The environments in the code below will not work in the manner that might be expected:

```
\newcomment{acomment} \newcomment{mycomment}
\begin{comment}
  \begin{acomment} %% comments cannot be nested
  ...
  \end{acomment}
  ...
  \begin{mycomment}
  ...
\end{comment}
...
  \end{mycomment}  %% comments cannot overlap
```

More encompassing `comment` environments are available if you use Victor Eijkhout's comment package [Eij99].

15.3 VERBATIMS

Standard LaTeX defines the `\verb` and `\verb*` commands for typesetting short pieces of text verbatim, short because they cannot include a linebreak. For longer verbatim texts the `verbatim` or `verbatim*` environments can be used. The star forms indicate spaces in the verbatim text by outputing a ␣ mark for each space. The class extends the standard verbatims in various ways.

If you have to write a lot of `\verb` text, as I have had to do for this book, it gets tedious to keep on typing this sort of thing: `\verb!verbatim text!`. Remember that the character immediately after the `\verb`, or `\verb*`, ends the verbatim processing.

`\MakeShortVerb{`⟨*backslash-char*⟩`}`
`\DeleteShortVerb{`⟨*backslash-char*⟩`}`

The `\MakeShortVerb` macro takes a character preceded by a backslash as its argument, say `\!`, and makes that character equivalent to `\verb!`. Using the character a second time will stop the verbatim processing. Doing, for example `\MakeShortVerb{\!}`, lets you then use `!verbatim text!` instead of the longer winded `\verb!verbatim text!`.

You have to pick as the short verb character one that you are unlikely to use; a good choice is often the | bar character as this rarely used in normal text. This choice, though may be unfortunate if you want to have any tabulars with vertical lines, as the bar character is used to specify those. The `\DeleteShortVerb` macro is provided for this contingency; give it the same argument as an earlier `\MakeShortVerb` and it will restore the short verb character to its normal state.

The `\MakeShortVerb` and `\DeleteShortVerb` macros come from the shortvrb package which is part of the LaTeX base system, but I have found them so convenient that I added them to the class.

`\setverbatimfont{`⟨*font-declaration*⟩`}`

The default font for verbatims is the normal sized monospaced font. The declaration `\setverbatimfont` can be used to specify a different font. The class default is

```
\setverbatimfont{\normalfont\ttfamily}
```

To use a smaller version simply say

```
\setverbatimfont{\normalfont\ttfamily\small}
```

A monospaced font is normally chosen as verbatim text is often used to present program code or typewritten text. If you want a more exotic font, try this

```
\setverbatimfont{\fontencoding{T1}\fontfamily{cmss}\selectfont}
```

and your verbatim text will then look like

```
We are no longer using the boring old typewriter font
for verbatim text. We used the T1 encoding
to make sure that characters that are often ligatures
like ‘‘, or ’’, or ---, or <, or >, print as expected.
After this we will switch back to the default verbatim font via
\setverbatimfont{\normalfont\ttfamily}
```

In the normal way of things with an OT1 fontencoding, typesetting the ligatures mentioned above in the sans font produces: ligatures like “, or ”, or —, or ¡, or ¿, which is not what happens in the `\verbatim` environment.

`\begin{verbatim}` anything `\end{verbatim}`
`\begin{verbatim*}` anything `\end{verbatim*}`

In the `verbatim` environment[3] you can write anything you want (except `\end{verbatim}`), and it will be typeset exactly as written. The `verbatim*` environment is similar except, like with `\verb*`, spaces will be indicated with a ␣ mark.

`\tabson[`⟨*number*⟩`]`
`\tabsoff`

The standard `verbatim` environment ignores any TAB characters; with the class's environment after calling the `\tabson` declaration the environment will handle TAB characters. By default 4 spaces are used to represent a TAB; the optional ⟨*number*⟩ argument to the

3 This version of the `verbatim` environment is heavily based on the verbatim package [SRR99] but does provide some extensions.

declaration will set the number of spaces for a TAB to be ⟨*number*⟩. Some folk like to use 8 spaces for a TAB, in which case they would need to declare `\tabson[8]`. Unremarkably, the declaration `\tabsoff` switches off TABs. The class default is `\tabsoff`.

```
\wrappingon
\wrappingoff
\verbatimindent
\verbatimbreakchar{⟨char⟩}
```

As noted, whatever is written in a `verbatim` environment is output just as written, even if lines are too long to fit on the page. The declaration `\wrappingon` lets the environment break lines so that they do not overflow. The declaration `\wrappingoff` restores the normal behaviour.

The following is an example of how a wrapped verbatim line looks. In the source the contents of the `verbatim` was written as a single line.

```
This is an example of line wrapping in the verbatim environment. It %
     is a single line in the source and the \wrappingon %
     declaration has been used.
```

The wrapped portion of verbatim lines are indented from the left margin by the length `\verbatimindent`. The value can be changed by the usual length changing commands. The end of each line that has been wrapped is marked with the ⟨*char*⟩ character of the `\verbatimbreakchar` macro. The class default is `\verbatimbreakchar{\char'\%}`, so that lines are marked with `%`. To put a '`/`' mark at the end of wrapped lines you can do

```
\setverbatimbreak{\char'\/}
```

or similarly if you would like another character. Another possibility is

```
\setverbatimchar{\char'\/\char'\*}
```

which will make '`/*`' the end marker.

15.3.1 Boxed verbatims

Verbatim environments are often used to present program code or, as in this book, LaTeX code. For such applications it can be useful to put the code in a box, or to number the code lines, or perhaps both.

```
\begin{fboxverbatim} anything \end{fboxverbatim}
```

The `fboxverbatim` environment typesets its contents verbatim and puts a tightly fitting frame around the result; in a sense it is similar to the `\fbox` command.

```
\begin{boxedverbatim} anything \end{boxedverbatim}
\begin{boxedverbatim*} anything \end{boxedverbatim*}
```

The `boxedverbatim` and `boxedverbatim*` environments are like the `verbatim` and `verbatim*` environments except that a box, allowing page breaks, may be put around the verbatim text and the lines of text may be numbered. The particular format of the output can be controlled as described below.

```
\bvbox \bvtopandtail \bvsides \nobvbox
\bvboxsep
```

Four styles of boxes are provided and you can extend these. Following the \bvbox declaration, a box is drawn round the verbatim text, breaking at page boundaries if necessary; this is the default style. Conversely, no boxes are drawn after the \nobvbox declaration. With the \bvtopandtail declaration horizontal lines are drawn at the start and end of the verbatim text, and with the \bvsides declarations, vertical lines are drawn at the left and right of the text. The separation between the lines and the text is given by the length \bvboxsep.

The following hooks are provided to set your own boxing style.

\bvtoprulehook \bvtopmidhook \bvendrulehook \bvleftsidehook \bvrightsidehook

The macros \bvtoprulehook and \bvendrulehook are called at the start and end of the boxedverbatim environment, and before and after page breaks. The macros \bvleftsidehook and \bvrightsidehook are called at the start and end of each verbatim line. The macro \bvtopmidhook is called after \bvtoprulehook at the start of the environment. It can be used to add some space if \bvtoprulehook is empty.

\bvperpagetrue \bvperpagefalse \bvtopofpage{⟨*text*⟩} \bvendofpage{⟨*text*⟩}

The command \bvperpagetrue indicates that a box should be visibly broken at a pagebreak, while there should be no visible break for \bvperpagefalse. If the box continues on to another page then it may be advantageous to place some sort of heading before the verbatim continues. Following the declaration \bvperpagetrue the ⟨*text*⟩ argument to \bvtopofpage will be typeset after any pagebreak. For example you could set:

```
\bvtopofpage{continued}
```

to print 'continued' in the normal text font.

By default, the class sets

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

which causes the \hrule to be drawn at the end of a page as the visible break (the rule is 0.4pt thick and the kern backs up that amount after the rule, so it effectively takes no vertical space). This is not always suitable. For instance, if there will be a 'continued' message at the top of the following page it may seem odd to draw a line at the bottom of the previous page. In this case, setting

```
\bvendofpage{}
```

will eliminate the rule.

As examples of the use of these hooks, here is how some of the boxed verbatim styles are defined.

The default style is \bvbox, which puts separate full boxes on each page.

```
\newcommand{\bvbox}{%
  \bvperpagetrue
  \renewcommand{\bvtoprulehook}{\hrule \nobreak \vskip-.1pt}%
  \renewcommand{\bvleftsidehook}{\vrule}%
  \renewcommand{\bvrightsidehook}{\vrule}%
  \renewcommand{\bvendrulehook}{\hrule}%
  \renewcommand{\bvtopmidhook}{\rule{0pt}{2\fboxsep} \hss}%
```

```
}
```

The \nobvbox turns off all boxing, and is defined as

```
\newcommand{\nobvbox}{%
  \bvperpagefalse
  \renewcommand{\bvtoprulehook}{}%
  \renewcommand{\bvleftsidehook}{}%
  \renewcommand{\bvrightsidehook}{}%
  \renewcommand{\bvendrulehook}{}%
  \renewcommand{\bvtopmidhook}{\rule{0pt}{2\fboxsep} \hss}%
}
```

The definitions of the other styles, \bvtopandtail and \bvsides, are intermediate between \bvbox and \nobvbox in the obvious manner.

\linenumberfrequency{⟨*nth*⟩}
\resetbvlinenumber
\setbvlinenums{⟨*first*⟩}{⟨*startat*⟩}
\linenumberfont{⟨*font declaration*⟩}

The command \linenumberfrequency controls the numbering of lines in a boxedverbatim — every ⟨*nth*⟩ line will be numbered. If ⟨*nth*⟩ is 0 or less, then no lines are numbered, if ⟨*nth*⟩ is 1 then each line is numbered, and if ⟨*nth*⟩ is n, where n is 2 or more, then only every nth line is numbered. Line numbering is continuous from one instance of the boxedverbatim environment to the next. Outside the environment the line numbers can be reset at any time by the command \resetbvlinenumber.

The \setbvlinenums macro can be used to specify that the number of the first line of the following boxedverbatim shall be ⟨*first*⟩ and the first printed number shall be ⟨*startat*⟩.

The \linenumberfont declaration sets ⟨*font declaration*⟩ as the font for the line numbers. The default specification for this is:

```
\linenumberfont{\footnotesize\rmfamily}
```

Line numbers are always set at the left of the lines because there is no telling how long a line might be and it might clash with a line number set at the right.

\bvnumbersinside
\bvnumbersoutside

Line numbers are typeset inside the box after the declaration \bvnumberinside and are typeset outside the box after the declaration \bvnumbersoutside. The default is to print the numbers inside the box.

Verbatim tabbing, but not wrapping, applies to the boxedverbatim environment.

Recommended alternative

Again the tcolorbox package offers boxes vs verbatim text.

15.3.2 New verbatims

The class implementation of verbatims lets you define your own kind of verbatim environment. Unfortunately this is not quite as simple as saying

```
\newverbatim{myverbatim}{...}{...}
```

as you can for defining normal environments. Instead, the general scheme is

```
\newenvironment{myverbatim}%
{<non-verbatim stuff> \verbatim <more non-verbatim stuff>}%
{\endverbatim}
```

In particular, you cannot use either the `\begin` or `\end` macros inside the definition of the new verbatim environment. For example, the following code will not work

```
\newenvironment{badverbatim}%
  {NBG\begin{verbatim}}{\end{verbatim}}
```

and this won't work either

```
\newenvironment{badverbatim}%
  {\begin{env}\verbatim}{\endverbatim\end{env}}
```

And, as with the standard `verbatim` environment, you cannot use the new one in the definition of a new command.

For an example of something that does work, this next little piece of typesetting was done in a new verbatim environment I have called `verbexami`, which starts and ends with a horizontal rule, and it shows the definition of `verbexami`.

The verbexami environment

```
\newenvironment{verbexami}%
  {\par\noindent\hrule The verbexami environment \verbatim}%
  {\endverbatim\hrule}
```

And this is a variation on the theme, with the environment again being enclosed by horizontal rules.

Verbexamii

```
Is this fun?
\newenvironment{verbexamii}%
  {\vspace{0.5\baselineskip}\hrule \vspace{0.2\baselineskip}
    Verbexamii \verbatim \textsc{Is this fun?}}%
  {\endverbatim\hrule\vspace{0.3\baselineskip}}
```

As no doubt you agree, these are not memorable examples of the typesetter's art but do indicate that you can define your own verbatim environments and may need to take a bit of care to get something that passes muster.

I will give some more useful examples, but mainly based on environments for writing verbatim files as I think that these provide a broader scope.

15.3.3 Example: the `lcode` environment

In this manual all the example LaTeX code has been typeset in the `lcode` environment; this is a verbatim environment defined especially for the purpose. Below I describe the code for defining my `lcode` environment, but first here is a simple definition of a verbatim environment, which I will call `smallverbatim`, that uses the `\small` font instead of the normalsize font.

```
\newenvironment{smallverbatim}%
  {\setverbatimfont{\normalfont\ttfamily\small}%
   \verbatim}%
  {\endverbatim}
```

The `verbatim` environment is implemented as a kind of `trivlist`, and lists usually have extra vertical space before and after them. For my environment I did not want any extra spacing so I defined the macro `\@zeroseps` to zero the relevant list spacings. I also wanted the code lines to be inset a little, so I defined a new length called `\gparindent` to use as the indentation.

```
\makeatletter
\newcommand{\@zeroseps}{\setlength{\topsep}{\z@}%
                        \setlength{\partopsep}{\z@}%
                        \setlength{\parskip}{\z@}}
\newlength{\gparindent} \setlength{\gparindent}{\parindent}
\setlength{\gparindent}{0.5\parindent}
% Now, the environment itself
\newenvironment{lcode}{\@zeroseps
  \renewcommand{\verbatim@startline}{%
                \verbatim@line{\hskip\gparindent}}
  \small\setlength{\baselineskip}{\onelineskip}\verbatim}%
  {\endverbatim
   \vspace{-\baselineskip}%
  \noindent
 }
\makeatother
```

Unless you are intimately familiar with the inner workings of the `verbatim` processing you deserve an explanation of the `lcode` definition.

Extremely roughly, the code for `\verbatim` looks like this:

```
\def\verbatim{%
  \verbatim@font
  % for each line, until \end{verbatim}
    \verbatim@startline
    % collect the characters in \verbatim@line
    \verbatim@processline{\the\verbatim@line\par}
    % repeat for the next line
}
```

The code first calls `\verbatim@font` to set the font to be used. Then, for each line it does the following:

- Calls the macro `\verbatim@startline` to start off the output version of the line.
- Collects all the characters comprising the line as a single token called `\verbatim@line`.
- If the characters are the string '`\end{verbatim}`' it finishes the verbatim environment.

- Otherwise it calls the macro `\verbatim@processline` whose argument is the characters in the line, treated as a paragraph. It then starts all over again with the next line.

I configured the `\verbatim@startline` macro to indent the line of text using a horizontal skip of `\gparindent`. The rest of the initialisation code, before calling `\verbatim` to do the real processing, just sets up the vertical spacing.

15.4 FILES

LaTeX reads and writes various files as it processes a document. Obviously it reads the document source file, or files, and it writes the `log` file recording what it has done. It also reads and writes the `aux` file, and may read and write other files like a `toc` file.

On occasions it can be useful to get LaTeX to read and/or write other files of your own choosing. Unfortunately standard LaTeX does not provide any easy method for doing this. The memoir class tries to rectify this.

`\jobname`

When you run LaTeX on your source file, say `fred.tex`, LaTeX stores the name of this file (`fred`) in the macro `\jobname`. LaTeX uses this to name the various files that it writes out — the `dvi` or `pdf` file, the `log` file, the `aux` file, etc.

TeX can read from 16 input streams and can write to 16 output streams. Normally an input stream is allocated for each kind of file that will be read and an output stream for each kind of file that will be written. On the input side, then, at least two streams are allocated, one for the source `tex` file and one for the `aux` file. On the output side again at least two streams are allocated, one for the `log` file and one for the `aux` file. When `toc` and other similar files are also part of the LaTeX process you can see that many of the 16 input and output streams may be allocated before you can get to use one yourself.

`\newoutputstream{`⟨*stream*⟩`}`
`\newinputstream{`⟨*stream*⟩`}`

The macros `\newoutputstream` and `\newinputstream` respectively create a new output and input stream called ⟨*stream*⟩, where ⟨*stream*⟩ should be a string of alphabetic characters, like `myout` or `myin`. The ⟨*stream*⟩ names must be unique, you cannot use the same name for two streams even if one is a input stream and the other is an output stream. If all the 16 streams of the given type have already been allocated TeX will issue a message telling you about this, of the form:

```
No room for a new write   % for an output stream
No room for a new read    % for an input stream
```

The two `\new...stream` commands also provide two empty macros called `\atstreamopen<stream>` and `\atstreamclose<stream>`. If these macros already exist then they are left undisturbed. For example if you do:

```
\newcommand{\atstreamopenmyout}{...}
\newoutputstream{myout}
\newinputstream{myin}
```

Then you will find that three new commands have been created like:

```
\newcommand{\atstreamclosemyout}{}
\newcommand{\atstreamopenmyin}{}
\newcommand{\atstreamclosemyin}{}
```

You can use `\renewcommand` to change the definitions of these if you wish.

`\IfStreamOpen{`⟨*stream*⟩`}{`⟨*true-code*⟩`}{`⟨*false-code*⟩`}`

The macro `\IfStreamOpen` checks whether or not the ⟨*stream*⟩ stream is open. If it is then the ⟨*true-code*⟩ argument is processed, while when it is not open the ⟨*false-code*⟩ argument is processed.

15.4.1 Writing to a file

One stream may be used for writing to several different files, although not simultaneously.

`\openoutputfile{`⟨*filename*⟩`}{`⟨*stream*⟩`}`
`\closeoutputstream{`⟨*stream*⟩`}`

The command `\openoutputfile` opens the file called ⟨*filename*⟩, either creating it if it does not exist, or emptying it if it already exists. It then attaches the file to the output stream called ⟨*stream*⟩ so that it can be written to, and then finally calls the macro named `\atstreamopen<stream>`.

The command `\closeoutputstream` firstly calls the macro named `\atstreamclose<stream>` then closes the output stream ⟨*stream*⟩, and finally detaches and closes the associated file.

`\addtostream{`⟨*stream*⟩`}{`⟨*text*⟩`}`

The `\addtostream` command writes ⟨*text*⟩ to the output stream ⟨*stream*⟩, and hence to whatever file is currently attached to the stream. The ⟨*stream*⟩ must be open. Any commands within the ⟨*text*⟩ argument will be processed before being written. To prevent command expansion, precede the command in question with `\protect`.

Writing verbatim text to a file is treated specially as it is likely to be the most common usage.

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

The text within a `verbatimoutput` environment is written verbatim to the ⟨*file*⟩ file. Alternatively, the contents of the `writeverbatim` environment are written verbatim to the ⟨*stream*⟩ stream.

Specifically, `verbatimoutput` opens the specified file, writes to it, and then closes the file. This means that if `verbatimoutput` is used more than once to write to a given file, then only the contents of the last of these outputs is captured in the file. On the other hand, you can use `writeverbatim` several times to write to the file attached to the stream and, providing the stream has not been closed in the meantime, all will be captured.

15.4.2 Reading from a file

One stream may be used for reading from several files, although not simultaneously.

`\openinputfile{`⟨*filename*⟩`}{`⟨*stream*⟩`}`
`\closeinputstream{`⟨*stream*⟩`}`

The command `\openinputfile` opens the file called ⟨*filename*⟩ and attaches it to the input stream called ⟨*stream*⟩ so that it can be read from. Finally it calls the macro named `\atstreamopen<stream>`. It is an error if ⟨*filename*⟩ can not be found.

The command `\closeinputstream` calls the macro named `\atstreamclose<stream>`, closes the output stream ⟨*stream*⟩, and then detaches and closes the associated file.

`\readstream{`⟨*stream*⟩`}`

The command `\readstream` reads the entire contents of the file currently associated with the input stream ⟨*stream*⟩. This provides the same functionality as `\input{`⟨*filename*⟩`}`.

`\readaline{`⟨*stream*⟩`}`

The `\readaline` reads what TeX considers to be one line from the file that is currently associated with the input stream ⟨*stream*⟩.

Multiple lines can be read by calling `\readaline` multiple times. A warning is issued if there are no more lines to be read (i.e., the end of the file has been reached).

Just as for writing, reading files verbatim is treated specially.

`\verbatiminput{`⟨*file*⟩`}` `\verbatiminput*{`⟨*file*⟩`}`
`\boxedverbatiminput{`⟨*file*⟩`}` `\boxedverbatiminput*{`⟨*file*⟩`}`
`\readverbatim{`⟨*stream*⟩`}` `\readverbatim*{`⟨*stream*⟩`}`
`\readboxedverbatim{`⟨*stream*⟩`}` `\readboxedverbatim*{`⟨*stream*⟩`}`

The commands `\verbatiminput` and `\boxedverbatiminput`, and their starred versions, act like the `verbatim` and `boxedverbatim` environments, except that they get their text from the ⟨*file*⟩ file. It is an error if ⟨*file*⟩ cannot be found. Similarly, `\readverbatim` and `\readboxedverbatim` get their text from the file currently attached to the ⟨*stream*⟩ input stream. It is an error if ⟨*stream*⟩ is not open for input.

15.4.3 Example: endnotes

In an earier version of the manual, this section contained an example as to how one could make endnotes. The example is now irrelevant, since memoir contain something similar to end notes called page notes, see section 12.6 on page 251.

Those interested in the code from the old example, can find it in the manual source (it has just been commented out).

15.4.4 Example: end floats

There are some documents where all figures are required to be grouped in one place, for instance at the end of the document or perhaps at the end of each chapter. Grouping at the end of a document with chapters is harder, so we'll tackle that one.

The basic idea is to write out verbatim each figure environment and then read them all back in at the end. We will use a stream, let's call it `tryout`, and call our file for figures `tryout.fig`.

```
\newoutputstream{tryout}
\openoutputfile{tryout.fig}{tryout}
```

If all were simple, in the document we could then just do

```
\begin{writeverbatim}{tryout}
\begin{figure} ... \end{figure}
\end{writeverbatim}
```

```
...
\closeoutputstream{tryout}
\input{tryout.fig}
```

So, what's the problem?

By default figure captions are numbered per chapter, and are preceeded by the chapter number; more precisely, the definition of a figure number is

```
\thechapter.\arabic{figure}
```

If we simply lump all the figures at the end, then they will all be numbered as if they were in the final chapter. For the sake of argument assume that the last chapter is number 10. The nth figure will then be numbered 10.n. One thing that we can do rather simply is to change the definition of the figure by using another counter, let's call it `pseudo`, instead of the chapter.

```
\newcounter{pseudo}
  \renewcommand{\thepseudo}{\arabic{pseudo}}
\renewcommand{\thefigure}{\thepseudo.\arabic{figure}}
```

Now, all we should have to do is arrange that the proper value of `pseudo` is available before each figure is typeset at the end. The code around the `figure` environments might then look like this

```
\addtostream{tryout}{\protect\setcounter{pseudo}{\thechapter}}
\begin{writeverbatim}{tryout}
\begin{figure}...
```

and a part of the file might then look like

```
...
\setcounter{pseudo}{4}
\begin{figure}...
```

The `\protect` before the `\setcounter` command will stop it from expanding before it is written to the file, while the `\thechapter` command *will* be expanded to give the actual number of the current chapter. This looks better as now at least the figure will be numbered 4.n instead of 10.n.

There is one last snag — figure numbers are reset at the start of each chapter — but if we just dump the figures at the end of the document then although the chapter part of the number will alter appropriately because of the `pseudo` process, the second part of the number will just increase continuously. It looks as though we should write out a change to the chapter counter at the start of each chapter. If we do that, then we should be able to get rid of the `pseudo` counter, which sounds good. But, and this is almost the last but, what if there are chapters after we have read in the figure file? To cater for this the chapter number of the last chapter before the file must be saved, and then restored after the figures have been processed.

Finally, wouldn't it be much better for the user if everything was wrapped up in an environment that handled all the messy stuff?

Here is the final code that I am going to produce which, by the way, is displayed in the `boxedverbatim` environment with line numbers and the following settings, just in case there is a page break in the middle of the box.

```
\nobvbox
\bvperpagetrue
\bvtopofpage{\begin{center}\normalfont%
             (Continued from previous page)\end{center}}
\bvendofpage{}
\resetbvlinenumber
\linenumberfrequency{1}
\bvnumbersoutside
\linenumberfont{\footnotesize\rmfamily}
\begin{boxedverbatim}
...
```

```
1  \newoutputstream{tryout}
2  \openoutputfile{\jobname.fig}{tryout}
3  \newcounter{pseudo}
4  \renewcommand{\thefigure}{\thepseudo.\arabic{figure}}
5  \newenvironment{writefigure}{%
6    \ifnum\value{chapter}=\value{pseudo}\else
7      \setcounter{pseudo}{\value{chapter}}
8      \addtostream{tryout}{\protect\stepcounter{chapter}}
9      \addtostream{tryout}{\protect\addtocounter{chapter}{-1}}
10     \addtostream{tryout}{%
11       \protect\setcounter{pseudo}{\thechapter}}
12   \fi
13   \addtostream{tryout}{\protect\begin{figure}}
14   \writeverbatim{tryout}}%
15  {\endwriteverbatim\finishwritefigure}
16 \newcommand{\finishwritefigure}{%
17   \addtostream{tryout}{\protect\end{figure}}}
18 \newcommand{\printfigures}{%
19   \closeoutputstream{tryout}%
20   \input{\jobname.fig}%
21 }
```

The above code should be either put in the preamble or in a separate package file.

The first four lines of the code perform the initial setup described earlier. Lines 1 and 2 set up for outputting to a file `\jobname.fig`, which is where the figures will be collected. Lines 3 and 4 create the new counter we need and change the construction of the figure number. The rest of the code defines a new environment `writefigure` which is to be used instead of the `figure` environment. It writes its content out to the `tryout` stream.

In line 6 a check is made to see if the current values of the `chapter` and `pseudo` counters are the same; nothing is done if they are. If they are different, it means that this is the first figure in the chapter and we have to put appropriate information into the figure file. Line 7 sets the `pseudo` counter to the value of the `chapter` counter (if there is another `writefigure` in the chapter it will then skip over the code in lines 7 to 11). The next lines put (where N is the number of the current chapter):

```
\stepcounter{chapter}
\addtocounter{chapter}{-1}
\setcounter{pseudo}{N}
```

into the figure file. Stepping the chapter number (by one) resets the following figure number, and then subtracting one from the stepped number returns the chapter number to its original value. Finally the counter `pseudo` is set to the number of the current chapter.

Line 13 puts

```
\begin{figure}
```

into the figure file, and line 14 starts the `writeverbatim` environment.

For the end of the `writefigure` environment (line 15), the `writeverbatim` environment is ended and after that the `\finishwritefigure` macro is called. This is defined in lines 16 and 17, and simply writes

```
\end{figure}
```

out to the figure file. The `\endwriteverbatim`, and any other kind of `\end...verbatim`, command is very sensitive to anything that follows it, and in this case did not like to be immediately followed by an `\addtostream{...}`, but did not mind it being wrapped up in the `\finishwritefigure` macro.

The `\printfigures` macro defined in the last three lines of the code simply closes the output stream and then inputs the figures file.

As an example of how this works, if we have the following source code:

```
\chapter{The fifth chapter}
...
\begin{writefigure}
%% illustration and caption
\end{writefigure}
...
\begin{writefigure}
%% another illustration and caption
\end{writefigure}
```

then the figure file will contain the following (shown verbatim in the `fboxverbatim` environment).

```
\stepcounter{chapter}
\addtocounter{chapter}{-1}
\setcounter{pseudo}{5}
\begin{figure}
%% illustration and caption
\end{figure}
\begin{figure}
%% another illustration and caption
\end{figure}
```

15.4.5 Example: questions and answers

Text books often have questions at the end of a chapter. Sometimes answers are also provided at the end of the book, or in a separate teachers guide. During the draft stages of

such a book it is useful to keep the questions and answers together in the source and paper drafts, only removing or repositioning the answers towards the end of the writing process.

This example provides an outline for meeting these desires. For pedagogical purposes I use a `\label` and `\ref` technique although there are better methods. The example also shows that not everything works as expected — it is a reasonably accurate rendition of the process that I actually went through in designing it.

First we need a counter for the questions and we'll use an environment for questions as these may be of any complexity. The environment takes one argument — a unique key to be used in a `\label`.

```
\newcounter{question} \setcounter{question}{0}
\renewcommand{\thequestion}{\arabic{question}}
\newenvironment{question}[1]%
  {\refstepcounter{question}
   \par\noindent\textbf{Question \thequestion:}\label{#1}}%
  {\par}
```

I have used `\refstepcounter` to increment the counter so that the `\label` will refer to it, and not some external counter.

We will use a file, called `\jobname.ans` to collect the answers and this will be written to by a stream. There is also a convenience macro, `\printanswers`, for the user to call to print the answers.

```
\newoutputstream{ansout}
```

A matching environment for answers is required. The argument to the environment is the key of the question.

In draft mode it is simple, just typeset the answer and no need to bother with any file printing (remember that `\ifdraftdoc` is `true` for a draft mode document).

```
\ifdraftdoc                         % when in draft mode
\newenvironment{answer}[1]%
  {\par\noindent\textbf{Answer \ref{#1}:}}%
  {\par}
\newcommand{\printanswers}{}
\else                               % when not in draft mode
```

In final mode the `answer` environment must write its contents verbatim to the `ans` file for printing by `\printanswers`. Dealing with these in reverse order, this is the definition of `\printanswer` when not in draft mode.

```
\newcommand{\printanswers}{%
  \closeoutputstream{ansout}
  \input{\jobname.ans}}
```

Now for the tricky bit, the `answer` environment. First define an environment that makes sure our output stream is open, and which then writes the answer title to the stream.

```
\newenvironment{@nswer}[1]{\@bsphack
  \IfStreamOpen{ansout}{}{%
    \openoutputfile{\jobname.ans}{ansout}%
  }%
```

```
  \addtostream{ansout}{\par\noindent\textbf{Answer \ref{#1}:}}%
  }{\@esphack}
```

The macros `\@bsphack` and `\@esphack` are LaTeX kernel macros that will gobble extraneous spaces around the environment. In other words, this environment will take no space in the typeset result. The `\IfStreamOpen` macro is used to test whether or not the stream is open, and if it isn't then it opens it. The answer title is then written out to the stream. Now we can define the `answer` environment so that its contents get written out verbatim.

```
\newenvironment{answer}[1]%
  {\@bsphack\@nswer{#1}\writeverbatim{ansout}}%
  {\par\endwriteverbatim\end@nswer\@esphack}
\fi                                % end of \ifdraftdoc ...\else ...
```

When I was testing this code I had a surprise as I got nasty error messages from LaTeX the first time around, but it worked fine when I processed the source a second time! The problem lies in the code line

```
\addtostream{ansout}{\par\noindent\textbf{Answer \ref{#1}:}}%
```

The first time around, LaTeX processed the `\ref` command and of course it was undefined. In this case `\ref` gets replaced by the code to print the error message, which involves macros that have `@` in their names, which LaTeX only understands under special circumstances. The second time around `\ref` gets replaced by the question number and all is well. I then remembered that some commands need protecting when they are written out, so I tried (I've wrapped the line to fit)

```
\addtostream{ansout}{\par\noindent
  \protect\makeatletter\textbf{Answer
  \protect\ref{#1}:}\protect\makeatother}%
```

which did work but seemed very clumsy.

I then took another line of attack, and looked at the definition of `\ref` to see if I could come up with something that didn't expand into `@` names. The result of this was

```
\addtostream{ansout}{\par\noindent\textbf{Answer
                                        \quietref{#1}:}}%
```

In the kernel file `ltxref.dtx` I found the definition of `\ref` and it used a macro `\@setref` (shown below) to do its work. My `\quietref` locally changes the definition of `\@setref` and then calls `\ref`, which will then use the modified `\@setref`.

```
\def\@setref#1#2#3{%        %% kernel definition
  \ifx#1\relax
    \protect\G@refundefinedtrue
    \nfss@text{\reset@font\bfseries ??}%
    \@latex@warning{Reference `#3' on page \thepage \space
                    undefined}%
  \else
    \expandafter#2#1\null
  \fi}
```

```
\DeclareRobustCommand{\quietref}[1]{\begingroup
  \def\@setref##1##2##3{%
    \ifx##1\relax ??\else
      \expandafter##2##1\null
    \fi
  \ref{#1}\endgroup}
```

Having gone all round the houses, the simplest solution was actually one that I had skipped over

```
\addtostream{ansout}{\par\noindent\textbf{Answer
                                          \protect\ref{#1}:}}%
```

The advantage of using the `\label` and `\ref` mechanism is that a question and its answer need not be adjacent in the source; I think that you have seen some of the disadvantages. Another disadvantage is that it is difficult to use, although not impossible, if you want the answers in a separate document.

The real answer to all the problems is force an answer to come immediately after the question in the source and to use the `question` counter directly, as in the endnotes example. In the traditional manner, this is left as an exercise for the reader.

15.5 ANSWERS

Question 1. As a convenience, the argument to the environment could be made optional, defaulting, say, to the current line width. If the default width is used the frame will be wider than the line width, so we really ought to make the width argument specify the width of the frame instead of the minipage. This means calculating a reduced width for the minipage based on the values of `\fboxsep` and `\fboxrule`.

```
\newsavebox{\minibox}
\newlength{\minilength}
\newenvironment{framedminipage}[1][\linewidth]{%
  \setlength{\minilength}{#1}
  \addtolength{\minilength}{-2\fboxsep}
  \addtolength{\minilength}{-2\fboxrule}
  \begin{lrbox}{\minibox}\begin{minipage}{\minilength}}%
  {\end{minipage}\end{lrbox}\fbox{\usebox{\minibox}}}
```

Question 2. There are at least three reasonable answers. In increasing or decreasing order of probability (your choice) they are:

- I took Sherlock Holmes' advice and followed the methods outlined in the chapter;
- I used a package, such as the answer package which is designed for the purpose;
- I just wrote the answers here.

Sixteen

Cross referencing

A significant aspect of LaTeX is that it provides a variety of cross referencing methods, many of which are automatic. An example of an automatic cross reference is the way in which a `\chapter` command automatically adds its title and page number to the ToC, or where a `\caption` adds itself to a 'List of...'.

Some cross references have to be specifically specified, such as a reference in the text to a particular chapter number, and for these LaTeX provides a general mechanism that does not require you to remember the particular number and more usefully does not require you to remember to change the reference if the chapter number is later changed.

16.1 Labels and references

The general LaTeX cross reference method uses a pair of macros.

```
\label{⟨labstr⟩}
\ref{⟨labstr⟩} \pageref{⟨labstr⟩}
```

You can put a `\label` command where you want to label some numbered element in case you might want to refer to the number from elsewhere in the document. The ⟨*labstr*⟩ argument is a sequence of letters, digits, and punctuation symbols; upper and lower case letters are different. The `\ref` macro prints the number associated with ⟨*labstr*⟩. The `\pageref` macro prints the number of the page where the `\label` specifying the ⟨*labstr*⟩ was placed.

The `\label` and `\ref` mechanism is simple to use and works well but on occasions you might be surprised at what `\ref` prints.

LaTeX maintains a current *ref* value which is set by the `\refstepcounter` command. This command is used by the sectioning commands, by `\caption`, by numbered environments like `equation`, by `\item` in an `enumerate` environment, and so on. The `\label` command writes an entry in the `aux` file consisting of the ⟨*labstr*⟩, the current *ref* value and the curent page number. A `\ref` command picks up the *ref* value for ⟨*labstr*⟩ and prints it. Similarly `\pageref` prints the page number for ⟨*labstr*⟩.

The critical point is that the `\label` command picks up the value set by the *most recent* visible[1] `\refstepcounter`.

- A `\label` after a `\section` picks up the `\section` number, not the `\chapter` number.
- A `\label` after a `\caption` picks up the caption number.
- A `\label` *before* a `\caption` picks up the surrounding sectional number.

1 Remember, a change within a group, such as an environment, is not visible outside the group.

If you are defining your own macro that sets a counter, the counter value will be invisible to any `\label` unless it is set using `\refstepcounter`.

`\fref{`⟨*labstr*⟩`}` `\figurerefname`
`\tref{`⟨*labstr*⟩`}` `\tablerefname`
`\pref{`⟨*labstr*⟩`}` `\pagerefname`

The class provides these more particular named references to a figure, table or page. For example the default definitions of `\fref` and `\pref` are

```
\newcommand{\fref}[1]{\figurerefname~\ref{#1}}
\newcommand{\pref}[1]{\pagerefname~\pageref{#1}}
```

and can be used as

```
\ldots footnote parameters are shown in~\fref{fig:fn}
on~\pref{fig:fn}.
```

which in this document prints as:

...footnote parameters are shown in Figure 12.1 on page 241.

`\Aref{`⟨*labstr*⟩`}` `\appendixrefname`
`\Bref{`⟨*labstr*⟩`}` `\bookrefname`
`\Pref{`⟨*labstr*⟩`}` `\partrefname`
`\Cref{`⟨*labstr*⟩`}` `\chapterrefname`
`\Sref{`⟨*labstr*⟩`}` `\sectionrefname`

Similarly, specific commands are supplied for referencing sectional divisions; `\Aref` for Appendix , `\Bref` for Book , `\Pref` for Part , `\Cref` for Chapter , and `\Sref` for divisions below Chapter . For example:

```
This is \Sref{sec:lab&ref} in \Cref{chap:xref}.
```

This is §16.1 in Chapter 16.

16.2 REFERENCE BY NAME

In technical works it is common to reference a chapter, say, by its number. In non-technical works such cross references are likely to be rare, and when they are given it is more likely that the chapter title would be used instead of the number, as in:

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

The chapter *An example book design* describes ...

There are two packages, nameref [Rahtz01] and titleref [Ars01a], that let you refer to things by name instead of number.

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

`\titleref{`⟨*labstr*⟩`}`

Typeset example 16.1: Named references should be to titled elements

Labels may be applied to:

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

Item 1 in section *Reference by name* mentions sections while item Named references should be to titled elements, on page 303 in the same section, mentions things like items in enumerated lists that should not be referenced by `\titleref`.

The macro `\titleref` is a class addition to the usual set of cross referencing commands. Instead of printing a number it typesets the title associated with the labelled number. This is really only useful if there *is* a title, such as from a `\caption` or `\section` command. For example, look at this code and its result.

Source for example 16.1

```
Labels may be applied to:
\begin{enumerate}
\item Chapters, sections, etc.            \label{sec:nxref:1}
...
\item Items in numbered lists, etc. \ldots \label{sec:nxref:5}
\end{enumerate}
Item \ref{sec:nxref:1} in section \textit{\titleref{sec:nxref}}
mentions sections while item \titleref{sec:nxref:5}, on page
\pageref{sec:nxref:5} in the same section, mentions things like
items in enumerated lists that should not be referenced
by \verb?\titleref?.
```

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

`\headnameref` `\tocnameref`

There can be three possibilities for the name of a sectional division; the full name, the name in the ToC, and the name in the page header. As far as `\titleref` is concerned it does not use the fullname, and so the choice simplifies to the ToC or page header. Following the declaration `\headnameref` it uses the page header name. Following the opposite declaration `\tocnameref`, which is the default, it uses the ToC name.

NOTE: Specifically with the memoir class, do not put a `\label` command inside an

Typeset example 16.2: Current title

This sentence in the section titled 'Current title' is an example of the use of the command \currenttitle.

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

\currenttitle

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

Source for example 16.2

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

\theTitleReference{⟨*num*⟩}{⟨*text*⟩}

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

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

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

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

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

The commands \titleref, \theTitleReference and \currenttitle are direct equivalents of those in the titleref package [Ars01a].

\namerefon \namerefoff

The capability for referencing by name has one potentially unfortunate side effect — it causes some arguments, such as that for \legend, to be moving arguments and hence any fragile command in the argument will need \protecting. However, not every document will require the use of \titleref and so the declaration \namerefoff is provided to switch it off (the argument to \legend would then not be moving). The declaration \namerefon, which is the default, enables name referencing.

Seventeen

Back matter

The back matter consists of reference and supportive elements for the main matter; things like bibliographies, indexes, glossaries, endnotes, and other material. The class provides additional elements and features of such matter that are not in the standard classes.

17.1 BIBLIOGRAPHY

Just as a reminder the bibliography is typeset within the `thebibliography` environment.

```
\bibname
\begin{thebibliography}{⟨exlabel⟩}
\bibitem ...
\end{thebibliography}
```

The environment takes one required argument, ⟨*exlabel*⟩, which is a piece of text as wide as the widest label in the bibliography. The value of `\bibname` (default 'Bibliography') is used as the title.

```
\bibintoc \nobibintoc
```

The declaration `\bibintoc` will cause the `thebibliography` environment to add the title to the ToC, while the declaration `\nobibintoc` ensures that the title is not added to the ToC. The default is `\bibintoc`.

```
\cite[⟨detail⟩]{⟨labstr-list⟩}
```

Within the text you call out a bibliographic reference using the `\cite` command, where ⟨*labstr-list*⟩ is a comma-separated list of identifiers for the cited works; there must be no spaces in this list. The optional ⟨*detail*⟩ argument is for any additional information regarding the citation such as a chapter or page number; this is printed after the main reference.

Various aspects of a bibliography can be changed and at this point it may be helpful to look at some of the internals of the `thebibliography` environment, which is defined like this

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

The bibliographic entries are typeset as a list, the `bibitemlist`.

```
\bibsection
```

The macro \bibsection defines the heading for the thebibliography environment; that is, everything before the actual list of items. It is effectively defined as

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

If you want to change the heading, redefine \bibsection. For example, to have the bibliography as a numbered section instead of an unnumbered chapter, redefine it like

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

If you use the natbib [Dal99a] and/or the chapterbib [Ars01b] packages with the sectionbib option, then they change \bibsection appropriately to typeset the heading as a numbered section.

\bibmark

\bibmark may be used in pagestyles for page headers in a bibliography. Its default definition is:

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

but could be redefined like, say,

```
\renewcommand*{\bibmark}{\markboth{\bibname}{}}
```

\prebibhook \postbibhook

The commands \prebibhook and postbibhook are called after typesetting the title of the bibliography and after typesetting the list of entries, respectively. By default, they are defined to do nothing. You may wish to use one or other of these to provide some general information about the bibliography. For example:

```
\renewcommand{\prebibhook}{%
CTAN is the Comprehensive \tx\ Archive Network and URLS for the
several CTAN mirrors can be found at \url{http://www.tug.org}.}
```

\biblistextra

Just at the end of setting up the bibitemlist the \biblistextra command is called. By default this does nothing but you may change it to do something useful. For instance, it can be used to change the list parameters so that the entries are typeset flushleft.

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

`\setbiblabel{`⟨*style*⟩`}`

The style of the labels marking the bibliographic entries can be set via `\setbiblabel`. The default definition is

```
\setbiblabel{[#1]\hfill}
```

where `#1` is the citation mark position, which generates flushleft marks enclosed in square brackets. To have marks just followed by a dot

```
\setbiblabel{#1.\hfill}
```

`\bibitem[`⟨*label*⟩`]{`⟨*labstr*⟩`}`
`\newblock`

Within the `bibitemlist` environment the entries are introduced by `\bibitem` instead of the more normal `\item` macro. The required ⟨*labstr*⟩ argument is the identifier for the citation and corresponds to a ⟨*labstr*⟩ for `\cite`. The items in the list are normally labelled numerically but this can be overriden by using the optional ⟨*label*⟩ argument. The `\newblock` command can be used at appropriate places in the entry for encouraging a linebreak (this is used by the openbib option).

`\bibitemsep`

In the listing the vertical space between entries is controlled by the length `\bibitemsep`, which by default is set to the normal `\itemsep` value. The vertical space is `(\bibitemsep + \parsep)`. If you wish to eliminate the space between items do

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

17.1.1 BibTex

Often the BibTeX program [Pat88a] is used to generate the bibliography list from database(s) of bibliographic data. For BibTeX a bibliographic data base is a `bib` file containing information necessary to produce entries in a bibliography. BibTeX extracts the raw data from the files for each citation in the text and formats it for typesetting according to a particular style.

`\bibliography{`⟨*bibfile-list*⟩`}`

The bibliography will be printed at the location of the `\bibliography` command. Its argument is a comma-separated list of BibTeX `bib` files which will be searched by BibTeX to generate the bibliography. Only the file name(s) should be supplied, the extension must not be given.

`\nocite{`⟨*labstr*⟩`}` `\nocite{*}`

The command `\nocite` causes BibTeX to make an entry in the bibliography but no citation will appear in the text. The special case `\nocite{*}` causes *every* entry in the database to be listed in the bibliography.

`\bibliographystyle{`⟨*style*⟩`}`

Many different BibTeX styles are available and the particular one to be used is specified by calling `\bibliographystyle` before the bibliography itself. The 'standard' bibliography ⟨*style*⟩s follow the general schemes for mathematically oriented papers and are:

`plain` The entry format is similar to one suggested by Mary-Claire van Leunen [Leu92], and entries are sorted alphabetically and labelled with numbers.

`unsrt` The format is the same as `plain` but that entries are ordered by the citation order in the text.

`alpha` The same as `plain` but entries are labelled like 'Wil89', formed from the author and publication date.

`abbrv` The same as `plain` except that some elements, like month names, are abbreviated.

There are many other styles available, some of which are used in collaboration with a package, one popular one being Patrick Daly's `natbib` [Dal99a] package for the kinds of author-year citation styles used in the natural sciences. Another package is `jurabib` [Ber02] for citation styles used in legal documents where the references are often given in footnotes rather than listed at the end of the document.

I assume you know how to generate a bibliography using BibTeX, so this is just a quick reminder. You first run LaTeX on your document, having specified the bibliography style, cited your reference material and listed the relevant BibTeX database(s). You then run BibTeX, and after running LaTeX twice more the bibliography should be complete. After a change to your citations you have to run LaTeX once, BibTeX once, and then LaTeX twice more to get an updated set of references.

The format and potential contents of a BibTeX database file (a `bib` file) are specified in detail in Lamport [Lam94] and the first of the *Companions* [MG+04]. Alternatively there is the documentation by Oren Patashnik [Pat88a] who wrote the BibTeX program.

A BibTeX style, specified in a `bst` file, is written using an anonymous stack based language created specifically for this purpose. If you can't find a BibTeX style that provides what you want you can either use the `makebst` [Dal99b] package which leads you through creating your own style via a question and answer session, or you can directly write your own. If you choose the latter approach then Patashnik's *Designing BibTeX files* [Pat88b] is essential reading. As he says, it is best to take an existing style and modify it rather than starting from scratch.

17.2 INDEX

It is time to take a closer look at indexing. The class allows multiple indexes and an index may be typeset as a single or a double column.

The general process is to put indexing commands into your source text, and LaTeX will write this raw indexing data to an `idx` file. The raw index data is then processed, not by LaTeX but by yourself if you have plenty of spare time on your hands, or more usually by a separate program, to create a sorted list of indexed items in a second file (usually an `ind` file). This can then be input to LaTeX to print the sorted index data.

17.2.1 Printing an index

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

In order to make LaTeX collect indexing information the declaration `\makeindex` must be put in the preamble. By default the raw index data is put into the `jobname.idx` file. If the optional ⟨*file*⟩ argument is given then index data can be output to `file.idx`. Several `\makeindex` declarations can be used provided they each call for a different file.

The \printindex command will print an index where by default the indexed items are assumed to be in a file called jobname.ind. If the optional ⟨*file*⟩ argument is given then the indexed items are read from the file called file.ind.

\begin{theindex} entries \end{theindex}
\onecolindex \twocolindex
\indexname

The index entries are typeset within the theindex environment. By default it is typeset with two columns but following the \onecolindex declaration the environment uses a single column. The default two column behaviour is restored after the \twocolindex declaration. The index title is given by the current value of \indexname (default 'Index').

\indexintoc \noindexintoc

The declaration \indexintoc will cause the theindex environment to add the title to the ToC, while the declaration \noindexintoc ensures that the title is not added to the ToC. The default is \indexintoc.

\indexcolsep
\indexrule

The length \indexcolsep is the width of the gutter between the two index columns The length \indexrule, default value 0pt, is the thickness of a vertical rule separating the two columns.

\preindexhook

The macro \preindexhook is called after the title is typeset and before the index listing starts. By default it does nothing but can be changed. For example

```
\renewcommand{\preindexhook}{Bold page numbers are used
  to indicate the main reference for an entry.}
```

\indexmark

\indexmark may be used in pagestyles for page headers in an index. Its default definition is:

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

but could be redefined like, say,

```
\renewcommand*{\indexmark}{\markboth{\indexname}{\indexname}}
```

\ignorenoidxfile
\reportnoidxfile

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

17.2.2 Preparing an index

It it is easy for a computer to provide a list of all the words you have used, and where they were used. This is called a concordance. Preparing an index, though, is not merely a gathering of words but is an intellectual process that involves recognising and naming

concepts, constructing a logical hierarchy of these and providing links between related concepts. No computer can do that for you though it can help with some tasks, such as sorting things into alphabetical order, eliminating duplicates, and so forth.

Several iterations may be required before you have an acceptable index. Generally you pick out the important words or phrases used on the first pass. Part of the skill of indexing is finding appropriate words to describe things that may not be obvious from the text. If there are several ways of describing something they may all be included using a 'see' reference to the most obvious of the terms, alternatively you could use 'see also' references between the items. Entries should be broken down into subcategories so that any particular item will not have a long string of page numbers and your reader is more likely to quickly find the relevant place. After having got the first index you will most probably have to go back and correct all the sins of ommission and commission, and start the cycle again.

I found that indexing this manual was the most difficult part of preparing it. It was easy to index the names of all the macros, environments, and so on as I had commands that would simultaneously print and index these. It was the concepts that was difficult. I inserted `\index` commands as I went along at what seemed to be appropriate places but turned out not to be. I would use slightly different words for the same thing, and what was worse the same word for different things. It took a long time to improve it to its present rather pitiful state.

`\index[`⟨*file*⟩`]{`⟨*stuff*⟩`}`

The `\index` macro specifies that ⟨*stuff*⟩ is to appear in an index. By default the raw index data — the ⟨*stuff*⟩ and the page number — will be output to the `jobname.idx` file, but if the optional ⟨*file*⟩ argument is given then output will be to the `file.idx` file.

This book has two indexes. The main index uses the default indexing commands, while the second index does not. They are set up like this:

```
% in preamble
\makeindex
\makeindex[lines]
% in body
...\index{main} ...\index[lines]{First line} ...
...
% at the end
\clearpage
% main index
\pagestyle{Index}
\renewcommand{\preindexhook}{%
The first page number is usually, but not always,
the primary reference to the indexed topic.\vskip\onelineskip}
\printindex

% second index
\clearpage
\pagestyle{ruled}
\renewcommand{\preindexhook}{}
\renewcommand{\indexname}{Index of first lines}
```

```
\onecolindex
\printindex[lines]
```

`\specialindex{`⟨*file*⟩`}{`⟨*counter*⟩`}{`⟨*stuff*⟩`}`

The `\index` command uses the page number as the reference for the indexed item. In contrast the `\specialindex` command uses the ⟨*counter*⟩ as the reference for the indexed ⟨*stuff*⟩. It writes ⟨*stuff*⟩ to the `file.idx` file, and also writes the page number (in parentheses) and the value of the ⟨*counter*⟩. This means that indexing can be with respect to something other than page numbers. However, if the hyperref package is used the special index links will be to pages even though they will appear to be with respect to the ⟨*counter*⟩; for example, if figure numbers are used as the index reference the hyperref link will be to the page where the figure caption appears and not to the figure itself.

`\showindexmarks` `\hideindexmarks`
`\indexmarkstyle`

The declaration `\showindexmarks` causes the argument to practically any `\index` and `\specialindex` to be printed in the margin of the page where the indexing command was issued. The argument is printed using the `\indexmarkstyle` which is initially specified as

```
\indexmarkstyle{\normalfont\footnotesize\ttfamily}
```

For reasons I don't fully understand, spaces in the argument are displayed as though it was typeset using the starred version of `\verb`. The `\hideindexmarks`, which is the default, turns off `\showindexmarks`.

The standard classes just provide the plain `\index` command with no optional ⟨*file*⟩ argument. In those classes the contents of the `jobname.idx` file is limited to the index entries actually seen in the document. In particular, if you are using `\include` for some parts of the document and `\includeonly` to exclude one or more files, then any `\index` entries in an excluded file will not appear in the `jobname.idx` file. The new implementation of indexing eliminates that potential problem.

`\item` `\subitem` `\subsubitem`

The `theindex` environment supports three levels of entries. A `\item` command flags a main entry; a subentry of a main entry is indicated by `\subitem` and a subentry of a subentry is flagged by `\subsubitem`. For example a portion of an index might look like:

```
\item bridge, 2,3,7
\subitem railway, 24
\subsubitem Tay, 37
```

bridge, 2, 3, 7
railway, 24
Tay, 37

if the Tay Bridge[1] was mentioned on page 37.

1 A railway (railroad) bridge in Scotland that collapsed in 1879 killing 90 people. The disaster lives for ever in the poem *The Tay Bridge Disaster* by William McGonagall (1830–?), the first verse of which goes:

> Beautiful Railway Bridge of the Silv'ry Tay!
> Alas! I am very sorry to say
> That ninety lives have been taken away
> On the last Sabbath day of 1879,
> Which will be remember'd for a very long time.

p. v:	`\index{Alf}`	`\indexentry{Alf}{v}`
p. 1:	`\index{Alf}`	`\indexentry{Alf}{1}`
p. 2:	`\index{Alf}`	`\indexentry{Alf}{2}`
p. 3:	`\index{Alf}`	`\indexentry{Alf}{3}`
p. 5:	`\index{Alfabet|see{Bet}}`	`\indexentry{Alfabet|see{Bet}}{5}`
p. 7:	`\index{Alf@\textit{Alf}}`	`\indexentry{Alf@\textit{Alf}}{7}`
	`\index{Bet|textbf}`	`\indexentry{Bet|textbf}{7}`
p. 8:	`\index{Alf!Bet!Con}`	`\indexentry{Alf!Bet!Con}{8}`
p. 9:	`\index{Alf!Dan}`	`\indexentry{Alf!Dan}{9}`

Figure 17.1: Raw indexing: (left) index commands in the source text; (right) `idx` file entries

17.2.3 MakeIndex

It is possible, but time consuming and error prone, to create your index by hand from the output of the `\index` commands you have scattered throughout the text. Most use the MakeIndex program to do this for them; there is also the xindy program [Keh98] but this is much less known.

```
\xindyindex
```

It turns out that xindy cannot handle a memoir hyperindex (which can be obtained with the aid of the hyperref package), although MakeIndex can do so.[2] If you are going to use xindy to process the raw index data put `\xindyindex` in the preamble, which will prevent hyperindexing.

MakeIndex reads an `idx` file containg the raw index data (which may include some commands to MakeIndex itself), sorts the data, and then outputs an `ind` file containing the sorted data, perhaps with some LaTeX commands to control the printing. MakeIndex was created as a general purpose index processing program and its operation can be controlled by a 'makeindex configuration file' (by default this is an `ist` file). Such a file consists of two parts. The first part specifies MakeIndex commands that can be included in the ⟨*stuff*⟩ argument to `\index`. The second part controls how the sorted index data is to be output.

I will only describe the most common elements of what you can put in an `ist` file; consult the MakeIndex manual [CH88], or the *Companion* [MG+04], for all the details.

You can embed commands, in the form of single characters, in the argument to `\index` that guide MakeIndex in converting the raw `idx` file into an `ind` file for final typesetting. The complete set of these is given in Table 17.1. They all have defaults and you can modify these via a MakeIndex configuration file.

In the simplest case you just use the name of the index item as the argument to the `\index` command. However, spaces are significant as far as MakeIndex is concerned. The following three uses of `\index` will result in three different entries in the final index
`\index{␣entry}` `\index{entry}` `\index{entry␣}`

The `!` character

The `level` specifier starts a new minor level, or subitem, with a maximum of two sub-levels. The default `level` specifier is the special character `!`. For example:

```
\index{item!sub item!sub sub item}
```

2 This deficiency in xindy was discovered by Frederic Connes, who also provided the `\xindyindex` command.

Table 17.1: MakeIndex configuration file input parameters

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

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

```
\begin{theindex}
\item Alf, v, 1-3
  \subitem Bet
    \subsubitem Con, 8
  \subitem Dan, 9
\item \textit{Alf}, 7
\item Alfabet, \see{Bet}{5}
\indexspace
\item Bet, \textbf{7}
\end{theindex}
```

Alf, v, 1-3
    Bet
        Con, 8
    Dan, 9
Alf, 7
Alfabet, *see* Bet

Bet, **7**

Figure 17.2: Processed index: (left) alphabeticized `ind` file; (right) typeset index

The @ character

An indexable item may be represented in two portions, separated by the `actual` specifier, which by default is the `@` character. The portion before the `@` is used when MakeIndex sorts the raw index data, and the portion after the `@` is used as the entry text. For example:

```
\index{MakeIndex@\textit{MakeIndex}}
```

will result in the final index entry of `\textit{MakeIndex}` in the alphabetic position accorded to `MakeIndex`. The same treatment can be applied for subitems:

```
\index{program!MakeIndex@\textit{MakeIndex}!commands}
```

The | character

Anything after the `encap` specifier, which by default is the `|` character, is treated as applying to the page number. In general

```
\index{...|str}
```

will produce a page number, say n, in the form

```
\str{n}
```

For example, if you wanted the page number of one particular entry to be in a bold font, say to indicate that this is where the entry is defined, you would do

```
\index{entry|textbf}
```

As a special case, if you want an index item to have a page range put the two characters `|(` at the end of the argument on the first page, and the character pair `|)` at the end of the argument on the last page. For example:

```
... \index{range|(} pages about range  \index{range|)} ...
```

The two arguments must match exactly except for the final `(` and `)`. You can also do

```
\index{...|(str}
```

which will produce a page range of the form

```
\str{n-m}
```

In this case, if the range is only a single page, the result is simply

```
\str{n}
```

`\see{`⟨*text*⟩`}{`⟨*page*⟩`}` `seename`
`\seealso{`⟨*text*⟩`}{`⟨*page*⟩`}` `alsoname`

The macros `\see` and `\seealso` are specifically for use in an `\index` command after the `|`. The `\see` command replaces the page number by the phrase '*see* ⟨*text*⟩', while the `\seealso` command adds '*see also* ⟨*text*⟩' to the entry. For example, in the source for this manual I have

```
\index{chapter!style|see{chapterstyle}}
\index{figure|seealso{float}}
```

A \see or \seealso should be used once only for a particular entry. The 'see' texts for \see and \seealso are stored in \seename and \alsoname, whose default definitions are:

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

The " and \ characters

If, for some reason, you want to index something that includes one of the !, @, | or " characters there is the difficulty of persuading MakeIndex to ignore the special meaning. This is solved by the quote specifier, which is normally the " character. The character immediately after " is treated as non-special. For example, if you needed to index the @ and ! characters:

```
\index{"@ (commercial at)}
\index{"! (exclamation)}
```

The leading " is stripped off before entries are alphabetized.

The escape specifier is used to strip the special meaning from the quote specifier. This is usually the \ character. So, to index the double quote character itself:

```
\index{\" (double quote)}
```

Example of using the special characters

Here is a short example of indexing the special characters. Given an input like this in the document

```
\index{exclamation mark ("!)}
\index{vicious|see{circle}}
\index{atsign@\texttt{"@} sign|\textbf}
\index{quote!double ("")}
\index{circle|see{vicious}}
```

then an index could eventually be produced that looks like:

@ sign, **30**
circle, *see* vicious
exclamation mark (!), 21
quote
    double ("), 47
vicious, *see* circle

17.2.4 Controlling MakeIndex output

Table 17.2 lists the parameters that control MakeIndex's output, except for the keywords that control the setting of page numbers. The special characters and strings are not fixed within the MakeIndex program. The program will read an ist file in which you can redefine all of MakeIndex's defaults.

I have used a file called memman.ist for configuring MakeIndex for this manual. Here it is:

Table 17.2: MakeIndex configuration file output parameters

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

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

```
% MakeIndex style file  memman.ist

% @ is a valid character in some entries, use ? instead
actual '?'

% output main entry <entry> as: \item \idxmark{<entry>},
item_0  "\n\\item \\idxmark{"
delim_0 "}, "
% not forgetting the subitem case
item_x1 "} \n \\subitem "

% Wrap and uppercase head letters
headings_flag 1
heading_prefix "\\doidxbookmark{"
heading_suffix "}"
```

Many items that I need to index include `@` as part of their names, which is one of the special characters. The `actual` line says that the character `?` performs the same function as the default `@` (and by implication, `@` is not a special character as far as MakeIndex is concerned).

The `item_0` line says that a main entry in the generated index starts

```
\item \idxmark{
```

and the `delim_0` and `item_x1` lines say that the main entry ends

```
}, % or
}
    \subitem
```

Thus, main entries will appear in an `ind` file like

```
\item \idxmark{a main entry}, <list of page numbers>
\item \idxmark{a main entry with no page numbers}
    \subitem subitem, <list of page numbers>
```

Read the MakeIndex manual [CH88] for full details on how to get MakeIndex to do what you want.

The `\doidxbookmark` that is used to wrap around the letter group headers, can be used to not only style the group header, but can also be used to add the headers in the bookmarks list. This can be done using

```
\newcommand{\doidxbookmark}[1]{{\def\@tempa{Symbols}\def\@tempb{#1}%
  \centering\bfseries \ifx\@tempa\@tempb %
  Analphabetics
  \phantomsection%
  \pdfbookmark[0]{Analphabetics}{Analphabetics-idx}%
%  \label{AnalphabeticsAnalphabeticsAnalphabetics-idx}%
  \else
  #1%
  \phantomsection%
  \pdfbookmark[0]{#1}{#1-idx}%
```

```
%  \label{#1#1#1-idx}%
  \fi%
  \vskip\onelineskip\par}}
```

The labels are generally not needed but can be used to add a visual representation of the index bookmarks into the index itself.

17.2.5 Indexing and the natbib package

The natbib package [Dal99a] will make an index of citations if `\citeindextrue` is put in the preamble after the natbib package is called for.

`\citeindexfile`

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

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

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

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

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

```
\makeindex[\citeindexfile]
```

before

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

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

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

or along the lines of

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

17.3 GLOSSARIES

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

`\makeglossary[`⟨*file*⟩`]`

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

`\printglossary[`⟨*file*⟩`]`

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

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

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

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

A simple glossary entry might be:

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

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

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

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

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

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

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

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

17.3.1 Controlling the glossary

Setting up makeindex

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

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

For other jobs just change the file names appropriately.

So, what is in a gst file? The potential contents were described earlier, but at a minimum you need this:

```
%%% basic.gst basic makindex glossary style file
%%% Output style parameters
preamble "\\begin{theglossary}"
postamble "\n\\end{theglossary}\n"
item_0    "\n\\glossitem"
delim_0   "{\\memglonum{"
encap_suffix "}}}"
headings_flag 1
heading_prefix "\\doglobookmark{"
heading_suffix "}"
%%% Input style parameters
keyword "\\glossaryentry"
```

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

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

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

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

A complete listing of the possible entries in a configuration file, also called a style file, for MakeIndex is in Table 17.1 and 17.2 with the exception of the output file page number setting keywords.

The \doglobookmark macro can be used to add bookmarks for the letter groups. In the case of this manual we do not write anything, just add the letters to the glossary entry in the bookmark list. In memsty \doglobookmark is defined as

```
\newcommand\doglobookmark[1]{%
  \def\@tempa{Symbols}\def\@tempb{#1}%
  \ifx\@tempa\@tempb %
  \phantomsection\pdfbookmark[0]{Analphabetics}{Analphabetics-glo}%
  \else%
  \phantomsection\pdfbookmark[0]{#1}{#1-glo}%
  \fi%
}
```

MakeIndex uses the word 'Symbols' to specify the group that does not start with a letter.

Raw input data

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

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

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

which MakeIndex then effectively converts into

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

The listing

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

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

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

The title for the glossary is `\glossaryname` whose initial definition is

```
\newcommand*{\glossaryname}{Glossary}
```

`\glossarymark`, which by default does nothing, can be redefined to set marks for headers. The glossary title will be added to the ToC if the `\glossaryintoc` declaration is in force, but will not be added to the ToC following the `\noglossaryintoc`.

```
\preglossaryhook
```

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

```
\onecolglossary \twocolglossary
\glossarycolsep \glossaryrule
```

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

```
\begintheglossaryhook
\atendtheglossaryhook
```

The last thing that `\begin{theglossary}` does is call `\begintheglossaryhook`. Similarly, the first thing that is done at the end of the environment is to call

\atendtheglossaryhook. By default these macros do nothing but you can redefine them.

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

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

The glossary for this document

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

This is the code in the sty file that I used.

```
\makeglossary
\changeglossactual{?}
\changeglossnum{\thepage}
\changeglossnumformat{|hyperpage}%% for hyperlinks
\renewcommand*{\glossaryname}{Command summary}
\renewcommand*{\glossarymark}{\markboth{\glossaryname}{\glossaryname}}
\renewcommand{\glossitem}[4]{%
  \sbox\@tempboxa{#1 \space #2 #3 \makebox[2em]{#4}}%
  \par\hangindent 2em
  \ifdim\wd\@tempboxa<0.8\linewidth
    #1 \space #2 #3 \dotfill \makebox[2em][r]{#4}\relax
  \else
    #1 \dotfill \makebox[2em][r]{#4}\\
    #2 #3
  \fi}
```

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

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

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

```
%%% memman.gst makindex glossary style file for memman and friends
%%% Output style parameters
preamble "\\begin{theglossary}"
postamble "\n\\end{theglossary}\n"
group_skip "\n\\glossaryspace\n"
item_0    "\n\\glossitem"
delim_0   "{\\memglonum{"
```

```
encap_suffix "}}}"
indent_space "\t"
indent_length 2
%%% Input style parameters
keyword "\\glossaryentry"
actual '?'
page_compositor "."
```

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

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

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

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

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

In particular, for this document, which also includes an index so that can be processed when the glossary is processed.

```
pdflatex memman
makeindex -s memman.gst -o memman.gls memman.glo
makeindex -s memman.ist memman     %%% for the index
makeindex lines                    %%% for the index of first lines
pdflatex memman
```

Eighteen

Miscellaneous

In which we talk of many things, but not shoes or ships or sealing wax, nor cabbages and kings.

This chapter started with the `\chapterprecis` command to add the above text, which is also added to the ToC.

The class provides a miscellany of minor facilities, which are described here.

18.1 Draft documents

The draft option marks any overfull lines with black rectangles, otherwise the appearance is the same as for a final document.

`\ifdraftdoc`

The `\ifdraftdoc` macro is provided so that you can add extra things that you might want to happen when processing a draft; for example you might want to have each page header (or footer) include the word 'DRAFT'. The code to do this can be put into a construct like the following:

```
\ifdraftdoc
  % special things for a draft document
\else
  % perhaps special things for a non-draft document
\fi
```

18.2 Change marks

When preparing a manuscript it normally goes through several iterations. The macros described in this section can be used to identify changes made to a document throughout its lifecycle.

The commands below implement a simplified version of the change marking in the iso class [Wil00b].

`\changemarks` `\nochangemarks`

The change marking macros only work properly when the draft class option is used. Additionally, the marks are only printed when the `\changemarks` declaration is in effect. Using `\nochangemarks` switches off any marking.

`\added{`⟨*change-id*⟩`}`
`\deleted{`⟨*change-id*⟩`}`
`\changed{`⟨*change-id*⟩`}`

Each of these macros puts a symbol and ⟨*change-id*⟩ into the margin near where the command is given. The `\added` macro indicates that something has been added to the manuscript and uses ⊕ as its symbol. The `\deleted` macro is for indicating that something has been deleted and uses the ≠ symbol. The macro `\changed` uses the ⇔ symbol to indicate that something has been changed.

These marking commands should be attached to some word or punctuation mark in the text otherwise extraneous spaces may creep into the typeset document.

18.3 TRIM MARKS

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

```
\showtrimsoff \showtrimson
```

If the showtrims class option has been used then the `\showtrimsoff` declaration switches off the trim marks; the `\showtrimson` declaration, which is the default, switches on the trim marks. These declarations do nothing if the showtrims option has not been used.

> ***Changed.*** From the LaTeX format dated 2020/10/01 we changed the way trimmarks are added to the pages. It will now go through the »`shipout/background`« hook instead of changing the `\shipout` routine directly. This also have the advantage that the hooks can be reordered if say you want something to be added to `shipout/background` *before* the trimmarks are added. For more details about the hooks see [Mit2020b, Mit2020c].
>
> The trimmarks will have the hook label »`memoir/trimmarks`«.

> ***Caveat.*** Most modern LaTeX editors make use of the *synctex* features in, say, pdfLaTeX to enable *reverse search* from the PDF previewer back to the source code in the editor. That is, click in a certain manner in the PDF previewer and you will be taken to the corresponding (or there about) line in the source code.
>
> Apparently the *synctex* feature does not like the trimmarks the class provide, or the *showlocs* page style. The code line one is referred back to may be off by tens of lines.
>
> Currently, there is no known workarounds for this deficiency.
>
> NB: We have yet to see if the 2020/10/01 format update and change to the trimmarks implementation will fix this issue.

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

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

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

coinciding with the page boundaries. The declaration \trimNone disables all kinds of trim marking. All three plus \quarkmarks (described below) is visibly shown on Figure 18.1.

The macro \trimmarkscolor is despite its name a normal (empty) macro. By redefining it one can change the color of the trimmarks, handy for example if the document has a dark background color. To make them blue use

```
\newcommand*{\trimmarkscolor}{\color{blue}}
```

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

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

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

\trimmark

The declaration \trimXmarks uses \trimmark for the corner crosses. This is defined as

```
\newcommand{\trimmark}{%
  \begin{picture}(0,0)
    \setlength{\unitlength}{1cm}\thicklines
    \put(-2,0){\line(1,0){4}}
    \put(0,-2){\line(0,1){4}}
  \end{picture}}
```

which produces a zero-sized picture of a 4cm cross. Then \trimXmarks is defined as:

```
\newcommand*{\trimXmarks}{%
  \let\tmarktl\trimmark
  \let\tmarktr\trimmark
  \let\tmarkbr\trimmark
  \let\tmarkbl\trimmark}
```

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

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

Thin horizontal lines of length 10mm will be drawn at the middle left and middle right of the page, starting 2mm outside the page boundary. This is what we do (now) by default for all four middle parts.

```
\quarkmarks
\registrationColour{⟨mark⟩}
```

Following the declaration `\quarkmarks` and trim marks will be in the style of Quark Xpress registration marks.[1] The marks will be typeset using `\registrationColour`. The default definition is simply

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

but you can change that to, say, print the marks in particular color. See Figure 18.1.

18.4 SHEET NUMBERING

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

```
sheetsequence
\thesheetsequence
```

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

```
lastsheet
lastpage
```

The counter `lastsheet` holds the number of sheets processed during the *previous* run of LaTeX. Similarly, the counter `lastpage` holds the number of the last page processed during the previous run. Note that the last page number is not necessarily the same as the last sheet number. For example:
In this document this is sheet 366 of 615 sheets, and page 328 of 577.

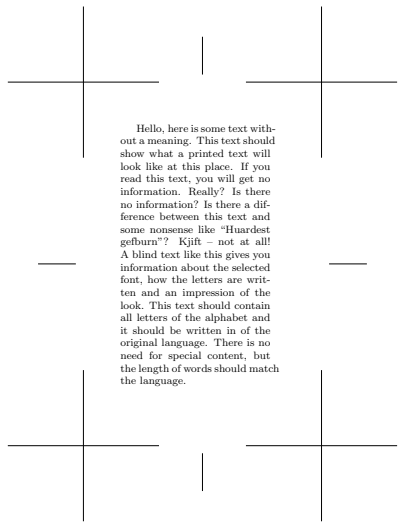
The previous sentence was the result of processing the following code

```
\textit{In this document this is
        sheet \thesheetsequence\ of \thelastsheet\ sheets,
        and page \thepage\ of \thelastpage.}
```

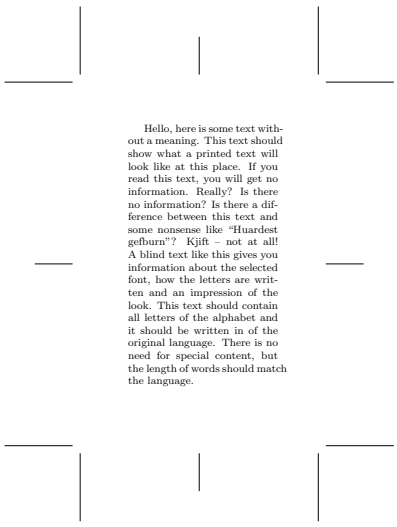
You may wish to use the sheet and/or page numbers as part of some trim marks. The following will note the sheet numbers above the page.

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

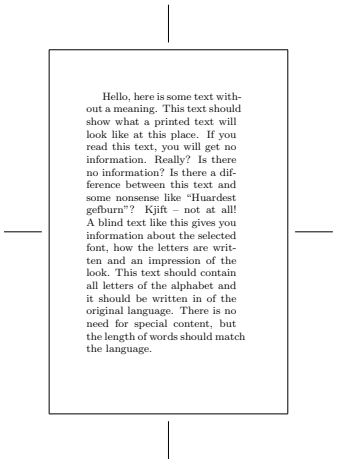
1 The code for this was donated by William Adams.



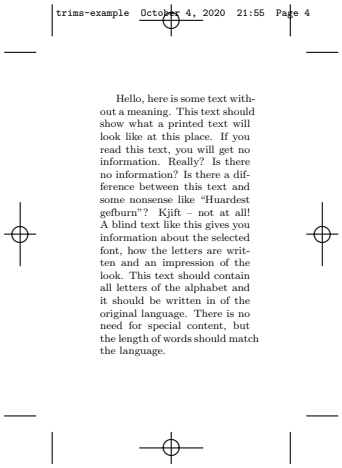
(a) `\trimXmarks` (default)



(b) `\trimLmarks`



(c) `\trimFrame`



(d) `\quarkmarks`

Figure 18.1: The four trimmark types

18.5 GATHERINGS OR SIGNATURES

Sometimes publishers request that they be supplied with a total number of pages that meet their planned *gatherings*.[2] For instance a gathering may consist of 8 leaves, and as there are two pages to a leaf this is equivalent to 16 pages. To meet this particular requirement there must be a total of $8N$ leaves, or equivalently $16N$ pages, where N will be the number of gatherings.

`\leavespergathering{`⟨*num*⟩`}`

The command `\leavespergathering` ensures that there will be exactly the right number of pages output to make a complete set of gatherings of ⟨*num*⟩ leaves (2⟨*num*⟩ pages) each — if necessary blank pages will be output at the end to make up the correct tally. If ⟨*num*⟩ is less than two (the default) then no additional pages will be output.

18.6 TIME

`\printtime` `\printtime*`
`\hmpunct` `\amname` `\pmname`

The `\printtime` command[3] prints the time of day when the document is processed using the 24 hour clock while `\printtime*` uses a 12 hour clock. For example, the effect of the next piece of code is shown below.

```
This document was processed on: \today\ at \printtime\ (\printtime*).
```

This document was processed on: October 4, 2020 at 21:55 (9:55 pm).

The punctuation between the hours and minutes is `\hmpunct` which defaults to a colon (:). The macros `\amname` and `\pmnane` hold the abbreviations for *ante meridiem* and *post meridiem*, respecitively; the defaults are 'am' and 'pm'.

According to the *Chicago Manual of Style* [Chi93] there should be no punctuation between the hours and minutes in the 24 hour system. For the 12 hour system it recommends that small caps be used for the divisions of the day (e.g., A.M. and P.M.) and also that the American practice is to use a colon as the separator between hours and minutes whereas the English practice is to use a period (known to the English as a 'full stop'). I don't know what the traditions are in other orthographies.

The `\quarkmarks` declaration uses `\printtime`, so be careful if you change it.

Nicola Talbot's datetime package [Tal06] provides a much more comprehensive collection of styles for printing the time; also for dates.

18.7 PAGE BREAKS BEFORE LISTS

A sentence or two may be used to introduce a list (e.g., `itemize`) and it can be annoying if there is a page break between the introductory words and the first item.

`\noprelistbreak`

Putting `\noprelistbreak` immediately before the `\begin{itemize}` should prevent a page break. Ideally, there should be no blank lines between the end of the introduction and the start of the list.

2 There was a thread on CTT, *pagenumber mod 4?* about this in 2008.
3 I based the code on a similar macro in *TeX for the Impatient* [AHK90].

18.8 CHANGING COUNTERS

This is effectively a bundling of the chngcntr package [Wil01e].

```
\newcounter{⟨ctr⟩}[⟨within⟩]
\thectr
```

In LaTeX a new counter called, say `ctr`, is created by the `\newcounter` command as `\newcounter{ctr}`. If the optional ⟨*within*⟩ argument is given, the counter `ctr` is reset to zero each time the counter called `within` is changed; the `within` counter must exist before it is used as the optional argument. The command `\thectr` typesets the value of the counter `ctr`. This is automatically defined for you by the `\newcounter` command to typeset arabic numerals.

```
\counterwithin{⟨ctr⟩}{⟨within⟩}
\counterwithin*{⟨ctr⟩}{⟨within⟩}
```

The `\counterwithin` macro makes a ⟨*ctr*⟩ that has been initially defined by `\newcounter{ctr}` act as though it had been defined by `\newcounter{ctr}[within]`. It also redefines the `\thectr` command to be `\thewithin.\arabic{ctr}`. The starred version of the command does nothing to the original definition of `\thectr`.

```
\counterwithout{⟨ctr⟩}{⟨within⟩}
\counterwithout*{⟨ctr⟩}{⟨within⟩}
```

The `\counterwithout` macro makes the `ctr` counter that has been initially defined by `\newcounter{ctr}[within]` act as though it had been defined by `\newcounter{ctr}`. It also redefines the `\thectr` command to be `\arabic{ctr}`. The starred version of the command does nothing to the original definition of `\thectr`.

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 these commands. If you want to change the value use `\setcounter`, and to change the typesetting style use `\renewcommand` on `\thectr`.

Caveat. As of 2018 `\counterwithout` and `\counterwithin` will be provided by the LaTeX-kernel, thus we only provide it if it does not already exist.

```
\letcountercounter{⟨counterA⟩}{⟨counterB⟩}
\unletcounter{⟨counterA⟩}
```

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*⟩`}` `\let`s (make the same) ⟨*counterA*⟩ to ⟨*counterB*⟩. The original of ⟨*counterA*⟩ is kept, such that you can unlet it later.

`\unletcounter{`⟨*counterA*⟩`}` restores ⟨*counterA*⟩ to its un`\let` condition.

This feature can be quite handy. Say for instance you want figures and tables to counter within the same counter (say table), then we need each change to the `figure` counter to actually act on the `table` counter. `\letcountercounter{figure}{table}` solves the problem.

18.9 NEW NEW AND PROVIDE COMMANDS

```
\newloglike{⟨cmd⟩}{⟨string⟩}
\newloglike*{⟨cmd⟩}{⟨string⟩}
```

The class provides means of creating new math log-like functions. For example you might want to do

```
\newloglike{\Ei}{Ei}
```

if you are using the exponential integral function in your work. The starred version of the command creates a function that takes limits (like the `\max` function).

The LaTeX kernel defines the `\providecommand` macro that acts like `\newcommand` if the designated macro has not been defined previously, otherwise it does nothing. The class adds to that limited repertoire.

```
\provideenvironment{⟨name⟩}[⟨numargs⟩][⟨optarg⟩]{⟨begindef⟩}{⟨enddef⟩}
\providelength{⟨cmd⟩}
\providecounter{⟨ctr⟩}[⟨within⟩]
\provideloglike{⟨cmd⟩}{⟨string⟩}
\provideloglike*{⟨cmd⟩}{⟨string⟩}
```

The macros `\provideenvironment`, `\providelength` and `\providecounter` take the same arguments as their `\new...` counterparts. If the environment, length or counter has not been defined then it is defined accordingly, otherwise the macros do nothing except produce a warning message for information purposes.

The `\provideloglike` commands are for math log-like functions, but they do not produce any warning messages.

18.10 CHANGING MACROS

Commands are provided for extending simple macro definitions.

```
\addtodef{⟨macro⟩}{⟨prepend⟩}{⟨append⟩}
\addtoiargdef{⟨macro⟩}{⟨prepend⟩}{⟨append⟩}
```

The macro `\addtodef` inserts ⟨*prepend*⟩ at the start of the current definition of ⟨*macro*⟩ and puts ⟨*append*⟩ at the end, where ⟨*macro*⟩ is the name of a macro (including the backslash) which takes no arguments. The `\addtoiargdef` macro is similar except that ⟨*macro*⟩ is the name of a macro that takes one argument.

For example the following are two equivalent definitions of `\mymacro`:

```
\newcommand{\mymacro}[1]{#1 is a violinist in spite of being tone deaf}
```

and

```
\newcommand{\mymacro}[1]{#1 is a violinist}
\addtoiargdef{\mymacro}{}{ in spite of being tone deaf}
```

The `\addtoiargdef` (and `\addtodef`) commands can be applied several times to the same ⟨*macro*⟩. Revising the previous example we could have

```
\newcommand{\mymacro}[1]{#1 is a violinist}
\addtoiargdef{\mymacro}{Although somewhat elderly, }%
                       { in spite of being tone deaf}
```

```
\addtoiargdef{\mymacro}{}{ and a bagpiper}
```

which is equivalent to

```
\newcommand{\mymacro}[1]{%
  Although somewhat elderly, #1 is a violinist
  in spite of being tone deaf and a bagpiper}
```

The ⟨*prepend*⟩ and ⟨*append*⟩ arguments may include LaTeX code, as shown in this extract from the class code:

```
\addtoiargdef{\date}{}{%
  \begingroup
    \renewcommand{\thanks}[1]{}
    \renewcommand{\thanksmark}[1]{}
    \renewcommand{\thanksgap}[1]{}
    \protected@xdef\thedate{#1}
  \endgroup}
```

Note that in the case of `\addtoiargdef` an argument can also refer to the original argument of the ⟨*macro*⟩.

`\addtodef*{`⟨*macro*⟩`}{`⟨*prepend*⟩`}{`⟨*append*⟩`}`
`\addtoiargdef*{`⟨*macro*⟩`}{`⟨*prepend*⟩`}{`⟨*append*⟩`}`

These starred versions are for use when the original ⟨*macro*⟩ was defined via `\newcommand*`. Using the starred versions is like using `\renewcommand*` and the unstarred versions are like having used `\renewcommand`. It is the version (starred or unstarred) of a sequence of `\addto...` commands that counts when determining whether the equivalent `\renew...` is treated as starred or unstarred.

The `\addto...` macros cannot be used to delete any code from ⟨*macro*⟩ nor to add anything except at the start and end. Also, in general, they cannot be used to change the definition of a macro that takes an optional argument, or a starred macro.

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

The `\patchcommand` is from the late Michael Downes' `patchcmd` package [Dow00]. It inserts the ⟨*start-code*⟩ at the start of the current definition of the macro ⟨*macro*⟩, and inserts ⟨*end-code*⟩ at the end of its current definition. The ⟨*macro*⟩ can have zero to nine parameters. If ⟨*macro*⟩ uses `\futurelet` (e.g., it is a starred command or takes an optional argument) only ⟨*start-code*⟩ is useful — ⟨*end-code*⟩ must be empty otherwise things get messed up. If ⟨*macro*⟩ has any delimited arguments then `\patchcommand` cannot be used.

18.11 String arguments

In the code for the class I have sometimes used macro arguments that consist of a 'string', like the `*` arguments in the page layout macros (e.g., `\settypeblocksize`), or the `flushleft`, `center` and `flushright` strings for the `\makeheadposition` macro.

`\nametest{`⟨*str1*⟩`}{`⟨*str2*⟩`}`
`\ifsamename`

The macro `\nametest` takes two strings as the arguments ⟨*str1*⟩ and ⟨*str2*⟩. It sets `\ifsamename true` if ⟨*str1*⟩ is the same as ⟨*str2*⟩, otherwise it sets `\ifsamename false`.

For the purposes of \nametest, a string is a sequence of characters which may include spaces and may include the \ backslash character; strings are equal if and only if their character sequences are identical.

Typically, I have used it within macros for checking on argument values. For example:

```
\newcommand{\amacro}[1]{%
  \nametest{#1}{green}
  \ifsamename
%    code for green
  \fi
  \nametest{#1}{red}
  \ifsamename
%    code for red
  \fi
  ...
}
```

18.12 ODD/EVEN PAGE CHECKING

It is difficult to check robustly if the current page is odd or even but the class does provide a robust method based on writing out a label and then checking the page reference for the label. This requires at least two LaTeX runs to stabilise. This has been extracted from the original chngpage package (which is no longer available). (The class code and chngpage code is similar but not identical. There is a later package, changepage [Wil08a] which contains code that is identical to the class.)

```
\checkoddpage
\ifoddpage
\strictpagecheck \easypagecheck
```

The macro \checkoddpage sets \ifoddpage to true if the current page number is odd, otherwise it sets it to false (the page number is even). The robust checking method involves writing and reading labels, which is what is done after the command \strictpagecheck is issued; it may take more than one run before everything settles down. The simple method is just to check the current page number which, because of TeX's asynchronous page breaking algorithm, may not correspond to the actual page number where the \checkoddpage command was issued. The simple, but faster, page checking method is used after the \easypagecheck command is issued.

```
\cplabel
```

When strict page checking is used the labels consist of a number preceded by the value of \cplabel, whose default definition is ^_ (e.g., a label may consist of the characters ^_21). If this might clash with any of your labels, change \cplabel with \renewcommand, but the definition of \cplabel must be constant for any given document.

18.13 MOVING TO ANOTHER PAGE

Standard LaTeX provides the `\newpage`, `\clearpage` and `\cleardoublepage` commands for discontinuing the current page and starting a new one. The following is a bundling of the nextpage package [Wil00c].

`\needspace{`⟨*length*⟩`}`

This macro decides if there is ⟨*length*⟩ space at the bottom of the current page. If there is, it does nothing, otherwise it starts a new page. This is useful if ⟨*length*⟩ amount of material is to be kept together on one page. The `\needspace` macro depends on penalties for deciding what to do which means that the reserved space is an approximation. However, except for the odd occasion, the macro gives adequate results.

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

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

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

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

`\movetoevenpage[`⟨*text*⟩`]`
`\cleartoevenpage[`⟨*text*⟩`]`

The `\movetoevenpage` stops the current page and starts typesetting on the next even numbered page. The `\clear...` version flushes out all floats before going to the next even page. The optional ⟨*text*⟩ is put on the skipped page (if there is one).

`\movetooddpage[`⟨*text*⟩`]`
`\cleartooddpage[`⟨*text*⟩`]`

These macros are similar to the `\...evenpage` ones except that they jump to the next odd numbered page.

A likely example for the optional ⟨*text*⟩ argument is

```
\cleartooddpage[\vspace*{\fill}THIS PAGE LEFT BLANK\vspace*{\fill}]
```

which will put 'THIS PAGE LEFT BLANK' in the centre of any potential skipped (empty) even numbered page.

`\cleartorecto` `\cleartoverso`

These are slightly simpler forms[4] of `\cleartooddpage` and `\cleartoevenpage`. For example, if you wanted the ToC to start on a verso page, like in *The TeXbook* [Knu84], then do this:

4 Perhaps more robust.

```
\cleartoverso
\tableofcontents
```

18.14 NUMBER FORMATTING

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

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

- Some digits, e.g., `\ordinal{123}` -> 123rd
- A macro expanding to digits, e.g., `\def\temp{3}\ordinal{\temp}` -> 3rd
- The value of a counter, e.g., `\ordinal{\value{page}}` -> 336th
- The arabic representation of a counter, e.g., `\ordinal{\thepage}` -> 336th
  However, if the representation of a counter is not completely in arabic digits, such as `\thesection` which here prints as 18.14, it will produce odd errors or peculiar results if it is used as the argument. For instance:
  `\ordinal{\thesection}` -> .1418th

18.14.1 Numeric numbers

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

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

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

Here are some examples:
`\fcardinal{12}` -> 12
`\fcardinal{1234}` -> 1,234
`\fcardinal{1234567}` -> 1,234,567
`\renewcommand*{\fnumbersep}{\:}\fcardinal{12345678}` -> 12 345 678
`\renewcommand*{\fnumbersep}{,\:}`

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

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

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

Some examples are:
`\fordinal{3}` -> 3rd
`\fordinal{12341}` -> 12, 341st

Typeset example 18.1: TeX's minimum number in words (English style)

The minimum number in TeX is minus two billion, one hundred and forty-seven million, four hundred and eighty-three thousand, six hundred and forty-seven (i.e., -2, 147, 483, 647)

```
\renewcommand{\ordscript}[1]{\textsuperscript{#1}}
\fordinal{1234567} -> 1, 234, 567th
\ordinal{1234567} -> 1234567th
This is the \ordinal{\value{chapter}} chapter. -> This is the 18th chapter.
```

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

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

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

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

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

18.14.2 Named numbers

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

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

As examples:

```
\numtoname{12345} -> twelve thousand, three hundred and forty-five
\numtoName{12345} -> Twelve thousand, three hundred and forty-five
\NumToName{12345} -> Twelve Thousand, Three Hundred and Forty-Five
```

Source for example 18.1

```
The minimum number in TeX is \numtoname{-2147483647}
(i.e., \fcardinal{-2147483647})
```

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

Typeset example 18.2: TeX's maximum number in words (American style)

The maximum number in TeX is two billion one hundred forty-seven million four hundred eighty-three thousand six hundred forty-seven (i.e., 2147483647).

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

For example:
`This is the \ordinaltoname{\value{chapter}} chapter. ->` This is the eighteenth chapter.

`\namenumberand` `\namenumbercomma` `\tensunitsep`

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

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

The next example shows how to achieve American typesetting.

Source for example 18.2

```
\renewcommand*{\namenumberand}{ }
\renewcommand*{\namenumbercomma}{ }
The maximum number in TeX is \numtoname{2147483647}
(i.e., \cardinal{2147483647}).
```

`\minusname` `\lcminusname` `\ucminusname`

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

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

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

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

Typeset example 18.3: Varieties of fractions in text

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

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

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

If you want to express numbers that fall outside TeX's range, Edward Reingold has produced a set of macros that can write out in words any number within the range $-10^{66} to 10^{66}$, that is, up to a thousand vigintillion [Rei07].

18.14.3 Fractions

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

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

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

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

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

Source for example 18.3

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

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

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

Typeset example 18.4: Super- and subscripts in text

In normal text you can typeset superscripts like H^+ and subscripts like H_2O without going into math mode.

class also provides the `\textsubscript` macro for typesetting its argument like a subscript.

Source for example 18.4

```
In normal text you can typeset superscripts like H\textsuperscript{+} and
subscripts like H\textsubscript{2}O without going into math mode.
```

18.15 AN ARRAY DATA STRUCTURE

The class includes some macros for supporting the `patverse` environment which may be more generally useful.

`\newarray{`⟨*arrayname*⟩`}{`⟨*low*⟩`}{`⟨*high*⟩`}`

`\newarray` defines the ⟨*arrayname*⟩ array, where ⟨*arrayname*⟩ is a name like `MyArray`. The lowest and highest array indices are set to ⟨*low*⟩ and ⟨*high*⟩ respectively, where both are integer numbers.

`\setarrayelement{`⟨*arrayname*⟩`}{`⟨*index*⟩`}{`⟨*text*⟩`}`
`\getarrayelement{`⟨*arrayname*⟩`}{`⟨*index*⟩`}{`⟨*result*⟩`}`

The `\setarrayelement` macro sets the ⟨*index*⟩ location in the ⟨*arrayname*⟩ array to be ⟨*text*⟩. Conversely, `\getarrayelement` sets the parameterless macro ⟨*result*⟩ to the contents of the ⟨*index*⟩ location in the ⟨*arrayname*⟩ array. For example:

```
\setarrayelement{MyArray}{23}{$2^{23}$}
\getarrayelement{MyArray}{23}{\result}
```

will result in `\result` being defined as `\def\result{$2^{23}$}`.

`\checkarrayindex{`⟨*arrayname*⟩`}{`⟨*index*⟩`}`
`\ifbounderror`

`\checkarrayindex` checks if ⟨*arrayname*⟩ is an array and if ⟨*index*⟩ is a valid index for the array. If both conditions hold then `\ifbounderror` is set `false`, but if either ⟨*arrayname*⟩ is not an array or, if it is, ⟨*index*⟩ is out of range then `\ifbounderror` is set `true`.

`\stringtoarray{`⟨*arrayname*⟩`}{`⟨*string*⟩`}`
`\arraytostring{`⟨*arrayname*⟩`}{`⟨*result*⟩`}`

The macro `\stringtoarray` puts each character from ⟨*string*⟩ sequentially into the ⟨*arrayname*⟩ array, starting at index 1. The macro `\arraytostring` assumes that

⟨*arrayname*⟩ is an array of characters, and defines the macro ⟨*result*⟩ to be that sequence of characters. For example:

```
\stringtoarray{MyArray}{Chars}
\arraytostring{MyArray}{\MyString}
```

is equivalent to

```
\def\MyString{Chars}
```

`\checkifinteger{`⟨*num*⟩`}` `\ifinteger`

The command `\checkifinteger` ckecks if ⟨*num*⟩ is an integer (not less than zero). If it is then `\ifinteger` is set `true`, otherwise it is set `false`.

Note. *Please note that* `\checkifinteger` *may only work on simple input.*

18.16 CHECKING THE PROCESSOR

18.16.1 Checking for pdfLaTeX

Both LaTeX and pdfLaTeX can be run on the same document. LaTeX produces a `.dvi` file as output, while pdfLaTeX can produce either a `.dvi` or a `.pdf` file. On modern systems pdfLaTeX produces a `pdf` file by default.

If you want a `dvi` file output use LaTeX and if you want a `pdf` file use pdfLaTeX.

`\ifpdf` ... `\fi`

The class provides `\ifpdf` (by autoloading the iftex package) which is `true` when the document is being processed by pdfLaTeX and `false` otherwise. You can use it like this:

```
\ifpdf
  code for pdfLaTeX only
\else
  code for LaTeX only
\fi
```

If there is no LaTeX specific code then don't write the `\else` part.

18.16.2 Checking for etex

Modern LaTeX distributions use `etex`, which is an extended version of TeX, as the underlying engine. `etex` provides some more primitives than TeX, which may be useful, but not everybody has `etex` available (Though, as of 2018, this is *very* rare).

`\ifetex`

`\ifetex` can be used to determine if `etex` is being used as the underlying engine; it is analagous to `\ifpdf` which tests for pdfLaTeX (provided by autoloading the iftex package). For example:

```
\ifetex
  %%% code only processible by etex
\else
  \typeout{etex is not available}
\fi
```

18.16.3 Checking for XeTeX

You have been able to use `\ifpdf` to check if PDF output is being used to process the document.

`\ifxetex`

In a similar manner you can use `\ifxetex` to check if the document is being processed by XeTeX (provided by autoloading the iftex package).

`\RequireXeTeX`

The iftex package also provides `\RequireXeTeX`, which generates an error if XeTeX is not being used to process the document. This can be useful if you make your own class building upon memoir.

18.16.4 Checking for LuaTeX

Similarly you can use

`\ifluatex`

to check if the doc is being process by LuaTeX. Again provided by loading iftex.

18.17 LEADING

LaTeX automatically uses different leading for different font sizes.

`\baselineskip` `\onelineskip`

At any point in a document the standard LaTeX `\baselineskip` length contains the current value of the leading[5]. The class provides the length `\onelineskip` which contains the initial leading for the normal font. This value is useful if you are wanting to specify length values in terms of numbers of lines of text.

18.18 MINOR SPACE ADJUSTMENT

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

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

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

5 This statement ignores any attempts to stretch the baseline.

both math and text mode to insert a medium space. Similarly, the class version of `\!` can be used to insert a negative thin space in both text and math mode.

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

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

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

18.19 Adding a period

Much earlier, when showing the code for the sectional division styles for this document, I used the macro `\addperiod`.

`\addperiod{`⟨*text*⟩`}`

This puts a period (a full stop) at the end of ⟨*text*⟩. I used it to add a period at the end of the `\paragraph` and `\subparagaph` titles. When sectional titles, like `\paragraph` are run-in, it is customary to end them with a period (or occasionally a colon).

18.20 Words and phrases

The class provides several macros that expand into English words or phrases. To typeset in another language these need to be changed, or an author or publisher may want some changes made to the English versions. Table 18.1 lists the macros, their default values, and where they used.

Most, if not all, of the tabulated definitions are simple — for example

```
\newcommand*{\partname}{Part}
\newcommand*{\partrefname}{Part~}
```

and so can be also changed simply.

The definitions of the macros for the names of numbers are more complex — for example for the number 11 (eleven)

```
\newcommand*{\nNamexi}{\iflowertonumname e\else E\fi leven}
```

That is, each definition includes both a lowercase and an uppercase initial letter, so a bit more care has to be taken when changing these. For specifics read the documentation of the class code.

18.21 Symbols

LaTeX lets you typeset an enormous variety of symbols. The class adds nothing to the standard LaTeX capabilities in this respect. If you want to see what symbols are available then get a copy of Scott Pakin's *The Comprehensive LaTeX Symbol List* [Pak01]. You may have to do a little experimentation to get what you want, though.

For example, the `\texttrademark` command produces the trademark symbol™, but the `\textregistered` command produces®. When I wanted to use the registered trademark symbol it needed to be typeset like a superscript instead of on the baseline. The `\textsuperscipt` macro typesets its argument like a superscript, so using

```
\textsuperscript{\textregistered}
```

gave the required result®.

Table 18.1: Defined words and phrases

Macro	Default	Usage
\abstractname	Abstract	title for abstract environment
\alsoname	see also	used by \seealso
\amname	am	used in printing time of day
\appendixname	Appendix	name for an appendix heading
\appendixpagename	Appendices	name for an \appendixpage
\appendixtocname	Appendices	ToC entry announcing appendices
\bibname	Bibliography	title for \thebibliography
\bookname	Book	name for \book heading
\bookrefname	Book	used by \Bref
\chaptername	Chapter	name for \chapter heading
\chapterrefname	Chapter	used by \Cref
\contentsname	Contents	title for \tableofcontents
\figurename	Figure	name for figure \caption
\figurerefname	Figure	used by \fref
\glossaryname	Glossary	title for \theglossary
\indexname	Index	title for \theindex
\lcminusname	minus	used in named number formatting
\listfigurename	List of Figures	title for \listoffigugres
\listtablename	List of Tables	title for \listoftables
\minusname	minus	used in named number formatting
\namenumberand	and	used in named number formatting
\namenumbercomma	,	used in named number formatting
\notesname	Notes	title of \notedivision
\pagename	page	for your use
\pagerefname	page	used by \pref
\partname	Part	name for \part heading
\partrefname	Part	used by \Pref
\pmnane	pm	used in printing time of day
\sectionrefname	§	used by \Sref
\seename	see	used by \see
\tablename	Table	name for table \caption
\tablerefname	Table	used by \tref
\ucminusname	Minus	used in named number formatting

18.22 TWO SIMPLE MACROS

There are two trivial macros that can be generally useful.

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

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

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

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

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

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

18.23 VERTICAL CENTERING

Earlier there was a description of a method for centering text vertically. The vplace environment provides a simpler and more general way.

\begin{vplace}[⟨*num*⟩] text \end{vplace}

The contents of the vplace environment are vertically centered. The optional ⟨*num*⟩ argument can be used to specify the ratio of the upper space to the lower space. You can put other text on the page above or below the centered text. The environment may be useful for title pages.

18.24 FOR PACKAGE WRITERS

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

18.24.1 Emulating packages

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

These commands are for package writers; they are based on a conversation with Donald Arseneau on CTT. They fool LaTeX into thinking that the ⟨*package*⟩ has already been loaded so it won't try loading it again. These are probably only useful if your package includes the actual code for ⟨*package*⟩.

memoir does include code from several packages and uses a similar internal command to ensure that the packages are not loaded following some later \usepackage command. The names of the emulated packages are written to the log file. At the time of writing the emulated packages are: abstract, appendix, array, booktabs, ccaption, chngcntr, crop, dcolumn, delarray, enumerate, epigraph, ifmtarg, ifpdf, index, makeidx, moreverb, needspace, newfile, nextpage, pagenote, patchcmd, parskip, setspace, shortvrb, showidx, tabularx, titleref, tocbibind, tocloft, verbatim, and verse. As well as the emulated packages memoir provides functions equivalent to those in the following packages, although the class does not prevent you from using them: fancyhdr, framed, geometry, sidecap, subfigure, and titlesec.

\DisemulatePackage{⟨*package*⟩}

This command undoes any prior \EmulatedPackage or \EmulatedPackageWithOptions for the ⟨*package*⟩ package. For example, if you wish to use the index package instead of memoir's emulation then put

```
\DisemulatePackage{index}
\usepackage{index}
```

in your preamble.

18.24.2 Inserting code before and after a file, package or class

> ***Changed.*** As of LaTeX format dated 2020/10/01 these \At... macros described below will be implemented using kernel hooks as layed out via [Mit2020a]. Which also means the order of the hooks can actually be changed. See [Mit2020b] for general information about the hooking feature.

> ***Caveat.*** In the {⟨*code*⟩} part of the hooks, any use of # needs to be doubled. Thus if you want to redefine the macro \foo which takes an argument at the end of package bar, you need to use
>
> ```
> \AtEndPackage{bar}{
>   \renewcommand\foo[1]{something something ##1}
> }
> ```
>
> In the format coming after 2020/10/01 we actually only needed a single #, but then our code is not backwards compatible, so the # is now forced even in the new emplementation.

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

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

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

\AtBeginFile{⟨*filename*⟩}{⟨*code*⟩} \AtEndFile{⟨*filename*⟩}{⟨*code*⟩}

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

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

Caveat. As of LaTeX format 2020/10/01 this is **no longer true!** You will now have to provide the `.tex` extension if it is a `.tex` file you want to hook into, as in:

```
\AtEndFile{test.tex}{\typeout{test.tex is done}}
```

Also note that the 2020/10/01 format will have some specific hooks what apply to `\include`. The class does not provide an interface for this and instead refers to [Mit2020a].

Note that `\At...File` actually did not work correctly for `\include` in the old implementation, it still needed the extension! In this new implementation both `\input` and `\include` trigger.

Changed. By the 2020/10/01 format this is now implemented via the `file/before/filename` and `file/after/filename` hooks. We are not adding any hook label.

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

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

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

Changed. By the 2020/10/01 format this is now implemented via the `package/before/pack` and `package/after/pack` hooks. We are not adding any hook label.

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

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

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

Changed. By the 2020/10/01 format this is now implemented via the `class/before/pack` and `class/after/pack` hooks. We are not adding any hook label.

The 2020/10/01 format hooking features will solve a lot of issues we have had with file hooking, especially when things are delayed via `\AtBeginDocument` and you want to have your code to run after someone else's code. The hooking feature now offers users to add rules to reorder how the hooks are executed. See [Mit2020b] for details.

18.25 HEADING HOOKS

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

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

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

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

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

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

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

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

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

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

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

Hooks for appendix-like page headings.

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

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

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

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

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

As an example of how one of these macros might be used, just before the start of this section I put

```
\renewcommand{\memsecinfo}[5]{\edef\Margi{#1}\edef\Margii{#2}%
                              \edef\Margiii{#3}\edef\Margiv{#4}%
                              \edef\Margv{#5}}
```

and now I'm putting

```
The arguments are: (1) `\Margi', (2) `\Margii', (3) `\Margiii',
                   (4) `\Margiv', (5) `\Margv'.
```

The arguments are: (1) 'section', (2) '18.25', (3) 'Heading hooks', (4) 'Heading hooks', (5) 'Heading hooks'.

Warning: Be very careful with the fifth argument of this macro when using hyperref, this argument will then contain a hyper link anchor, whih may cause problems when used out of context.

18.26 Documenting LaTeX commands

The class provides a few macros to help you if you want to describe LaTeX commands.

```
\bs \cs{⟨name⟩} \cmdprint{⟨cmd⟩} \cmd{⟨cmd⟩}
```

The macro `\bs` simply prints the '\' backslash.

The macro `\cs` prints its argument, putting a backslash in front of it. For example `\cs{name}` prints `\name`.

The argument to `\cmdprint` should be the name of a macro, including the backslash. It is then printed as is. For instance `\cmdprint{\amacro}` prints `\amacro`.

The argument to `\cmd` should be the name of a macro, including the backslash. It is then printed, using `\cmdprint`, and also added to the index file with the assumption that `?` will be used as the 'actual' character (the default is `@` which is not of much use if you are trying to index macro names that have `@` as part of their names).

```
\meta{⟨arg⟩} \marg{⟨arg⟩} \oarg{⟨arg⟩} \parg{⟨arg⟩}
```

The macro `\meta{`⟨*arg*⟩`}` prints ⟨*arg*⟩ for an argument to a macro.
The macro `\marg{`⟨*arg*⟩`}` prints `{`⟨*arg*⟩`}` for a required argument.
The macro `\oarg{`⟨*arg*⟩`}` prints `[`⟨*arg*⟩`]` for an optional argument.
The macro `\parg{`⟨*arg*⟩`}` prints `(`⟨*arg*⟩`)` for a parenthesized argument.

Nineteen

For package users

Many packages work just as well with memoir as with any of the standard classes. In some instances, though, there may be a problem. In other instances the class and package have been designed to work together but you have to be careful about any code that you might write.

19.1 Class/package name clash

A typical indication of a problem is an error message saying that a command has already been defined (see page 465).

When the class and a package both use the same name for a macro that they define something has to give. For the sake of an example, assume that memoir has defined a macro called `\foo` and a package pack that is used in the document also defines `\foo`. There are several options that you can choose which might resolve the difficulty.

1. Just keep the class definition:

```
\documentclass{...}
\let\classfoo\foo   % save the class' definition
\let\foo\relax      % 'undefine' \foo
\usepackage{pack}   % defines \foo
\let\foo\classfoo   % restore \foo to the class definition
...
\foo...             % the class \foo
```

2. Just keep the package definition:

```
\documentclass{...}
\let\foo\relax      % 'undefine' \foo
\usepackage{pack}   % defines \foo
...
\foo...             % the package \foo
```

3. Keep the class definition and rename the package definition:

```
\documentclass{...}
\let\classfoo\foo   % save the class' definition
\let\foo\relax      % 'undefine' \foo
\usepackage{pack}   % defines \foo
\let\packfoo\foo    % rename pack's \foo
```

```
\let\foo\classfoo   % restore \foo to the class definition
...
\foo...             % the class \foo
\packfoo            % the package \foo
```

4. Keep the package definition and rename the class definition:

```
\documentclass{...}
\let\classfoo\foo   % save the class' definition
\let\foo\relax      % 'undefine' \foo
\usepackage{pack}   % defines \foo
...
\foo...             % the package \foo
\classfoo...        % the class \foo
```

A potential problem with these options can occur after the package is loaded and there are class or package commands that you use that, knowingly or not, call `\foo` expecting to get the class or the package definition, one of which is now not available (except under a different name).

The memoir class has been available since 2001. It seems likely that if older packages clash with memoir then, as eight years have gone by, the authors are unlikely to do anything about it. If a newer package clashes then contact the author of the package.

If all else fails, ask for help on the CTT newsgroup.

19.2 SUPPORT FOR BIDIDIRECTIONAL TYPESETTING

The bidi system [Kha10] provides means of bidirectional typesetting. The class has built in support for bidi but this means that if you are defining your own macros there are some things you need to be aware of.

When dealing with bidirectional texts the left-to-right (LTR) direction is the familiar one and LaTeX is set up for this. When typesetting right-to-left (RTL) bidi interchanges left and right. The support in memoir consists of replacing many, but not all, of the right- and left-specific constructs. The replacement macros are:

```
\memRTLleftskip \memRTLrightskip
\memRTLvleftskip \memRTLvrightskip
\memRTLraggedright \memRTLraggedleft
```

In certain places, but not everywhere:
`\memRTLleftskip` is used instead of `\leftskip`
`\memRTLrightskip` is used instead of `\rightskip`
`\memRTLvleftskip` is used instead of `\vleftskip`
`\memRTLvrightskip` is used instead of `\vrightskip`
`\memRTLraggedleft` is used instead of `\raggedleft`
`\memRTLraggedright` is used instead of `\raggedright`

So, if you are defining any macros that use the `\...skip` or `\ragged...` macros you may have to use the `\memRTL...` version instead. The memoir definitions of these macros are simply:

```
\newcommand*{\memRTLleftskip}{\leftskip}
\newcommand*{\memRTLrightskip}{\rightskip}
...
\newcommand*{\memRTLraggedleft}{\raggedleft}
```

The bidi system redefines them to suit its purposes. If your work will only be set LTR there is no need to use the `\memRTL...` macros in any of your code. If you might ever be producing a bidirectional document then you may have to use the `\memRTL...` versions of the standard commands in your code. To determine where you might have to use them you will have to consult the bidi documentation as not every use of the standard commands needs to be replaced.

20.1 INTRODUCTION

In this chapter I will work through a reasonably complete design exercise. Rather than trying to invent something myself I am taking the design of Bringhurst's *The Elements of Typographic Style* [Bri99] as the basis of the exercise. This is sufficiently different from the normal LaTeX appearance to demonstrate most of the class capabilities, and also it is a design by a leading proponent of good typography.

As much as possible, this chapter is typeset according to the results of the exercise to provide both a coding and a graphic example.

20.2 DESIGN REQUIREMENTS

The *Elements of Typographic Style* is typeset using Minion as the text font and Syntax (a sans font) for the captions.

The trimmed page size is 23 by 13.3cm. The foredge is 3.1cm and the top margin is 1.9cm.

As already noted, the font for the main text is Minion, with 12pt leading on a 21pc measure with 42 lines per page. For the purposes of this exercise I will assume that Minion can be replaced by the font used for this manual. The captions to figures and tables are unnamed and unnumbered and typeset in Syntax. The captions give the appearance of being in a smaller font size than the main text, which is often the case. I'll assume that the `\small\sfseries` font will reasonably do for the captions.

The footer is the same width as the typeblock and the folio is placed in the footer at the fore-edge. There are two blank lines between the bottom of the typeblock and the folio.

There is no header in the usually accepted sense of the term but the chapter title is put on recto pages and section titles are on verso pages. The running titles are placed in the fore-edge margin level with the seventh line of the text in the typeblock. The recto headers are typeset raggedright and the verso ones raggedleft.

Bringhurst also uses many marginal notes, their maximum width being about 51pt, and typeset raggedright in a smaller version of the textfont.

Chapter titles are in small caps, lowercase, in a larger font than for the main text, and a rule is placed between the title and the typeblock. The total vertical space used by a chapter title is three text lines. Chapters are not numbered in the text but are in the ToC.

Section titles are again in small caps, lowercase, in the same size as the text font. The titles are numbered, with both the chapter and section number.

A subsection title, which is the lowest subdivision in the book, is in the italic form of the textfont and is typeset as a numbered non-indented paragraph. These are usually multiline as Bringhurst sometimes uses them like an enumerated list, so on occasion there is a subsection title with no following text.

Only chapter titles are put into the ToC, and these are set raggedright with the page numbers immediately after the titles. There is no LoF or LoT.

Note that unlike the normal LaTeX use of fonts, essentially only three sizes of fonts are used — the textfont size, one a bit larger for the chapter titles, and one a bit smaller for marginal notes and captions. Also, bold fonts are not used except on special occasions,

such as when he is comparing font families and uses large bold versions to make the differences easier to see.

20.3 SPECIFYING THE PAGE AND TYPEBLOCK

Specifying the page and typeblock

The first and second things to do are to specify the sizes of the page after trimming and the typeblock. The trimmed size is easy as we have the dimensions.

```
\settrimmedsize{23cm}{13.3cm}{*}
```

We want 42 lines of text, so that's what we set as the height of the typeblock; however, we have to remember to ask for `lines` as the optional ⟨*algorithm*⟩ argument when we finally call `\checkandfixthelayout`.

```
\settypeblocksize{42\onelineskip}{21pc}{*}
```

To make life easier, we'll do no trimming of the top of the stock

```
\setlength{\trimtop}{0pt}
```

but will trim the fore-edge. The next set of calculations first sets the value of the `\trimedge` to be the `\stockwidth`; subtracting the trimmed `\paperwidth` then results in `\trimedge` being the amount to trim off the foredge.

```
\setlength{\trimedge}{\stockwidth}
\addtolength{\trimedge}{-\paperwidth}
```

The sizes of the trimmed page and the typeblock have now been specified. The typeblock is now positioned on the page. The sideways positioning is easy as we know the fore-edge margin to be 3.1cm.

```
\setlrmargins{*}{3.1cm}{*}
```

The top margin is specified as 1.9cm, which is very close to four and a half lines of text. Just in case someone might want to use a different font size, I'll specify the top margin so that it is dependent on the font size. The `\footskip` can be specified now as well (it doesn't particularly matter what we do about the header-related lengths as there isn't anything above the typeblock).

```
\setulmargins{4.5\onelineskip}{*}{*}
\setheadfoot{\onelineskip}{3\onelineskip}
\setheaderspaces{\onelineskip}{*}{*}
```

Lastly define the dimensions for any marginal notes.

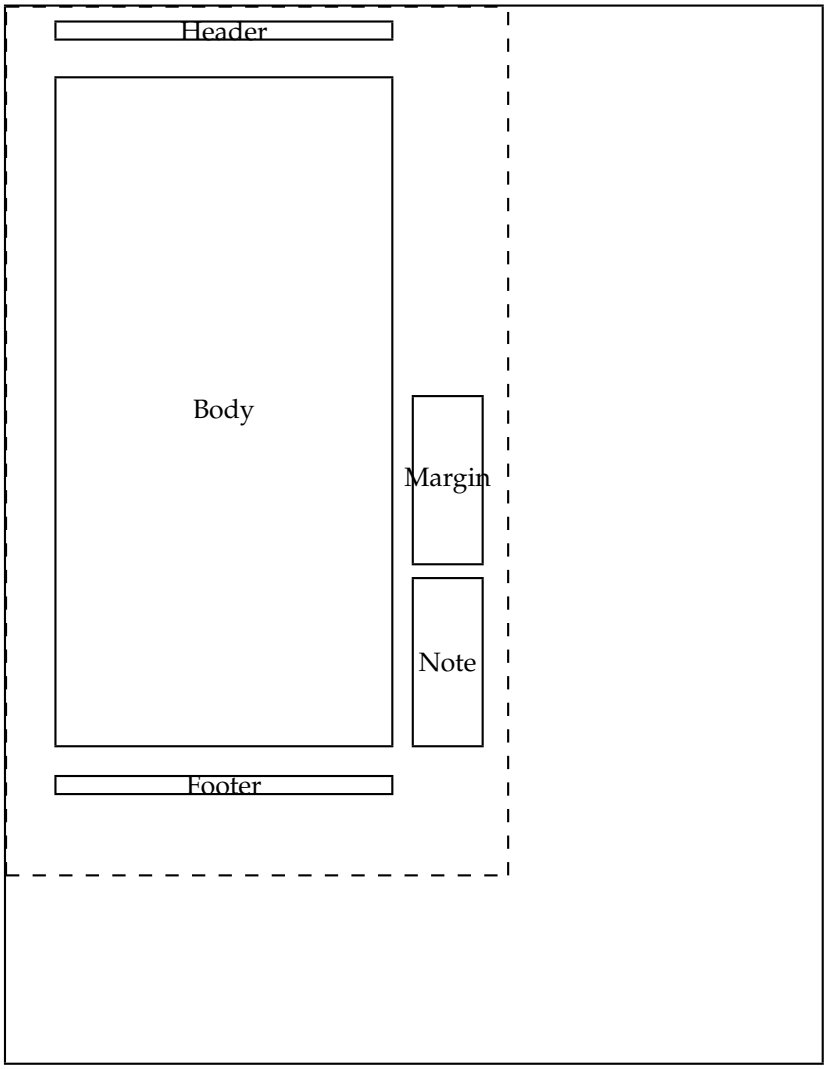
```
\setmarginnotes{17pt}{51pt}{\onelineskip}
```

If this was for real, the page layout would have to be checked and implemented.

```
\checkandfixthelayout[lines]
```

It is possible to implement this layout just for this chapter but I'm not going to tell you either how to do it, or demonstrate it. Except under exceptional circumstances it is not good to make such drastic changes to the page layout in the middle of a document. However, the picture on page 357 illustrates how this layout would look on US letterpaper stock. Looking at the illustration suggests that the layout would look rather odd unless the stock was trimmed down to the page size — another reason for not switching the layout here.

Dashed lines represent the actual page size after trimming the stock.



An example book design

Lengths are to the nearest pt.

`\stockheight = 795pt`	`\stockwidth = 614pt`
`\pageheight = 654pt`	`\pagewidth = 378pt`
`\textheight = 502pt`	`\textwidth = 252pt`
`\trimtop = 0pt`	`\trimedge = 236pt`
`\uppermargin = 54pt`	`\spinemargin = 38pt`
`\headheight = 12pt`	`\headsep = 30pt`
`\footskip = 36pt`	`\marginparsep = 17pt`
`\marginparpush = 12pt`	`\columnsep = 10pt`
`\columnseprule = 0.0pt`	

An illustration of Bringhurst's page layout style when printed on US letter paper stock. Also shown are the values used for the page layout parameters for this design.

20.4.1 The chapter style

Recapping, chapter titles are in small caps, lowercase, in a larger font than for the main text, and a rule is placed between the title and the typeblock. The total vertical space used by a chapter title is three text lines. Chapters are not numbered in the text but are in the ToC. Titles in the ToC are in mixed case.

Specifying the sectional titling styles

The definition of the chapterstyle is remarkably simple, as shown below.

```
%% Bringhurst chapter style
\makechapterstyle{bringhurst}{%
  \renewcommand{\chapterheadstart}{}
  \renewcommand{\printchaptername}{}
  \renewcommand{\chapternamenum}{}
  \renewcommand{\printchapternum}{}
  \renewcommand{\afterchapternum}{}
  \renewcommand{\printchaptertitle}[1]{%
    \raggedright\Large\scshape\MakeLowercase{##1}}
  \renewcommand{\afterchaptertitle}{%
    \vskip\onelineskip \hrule\vskip\onelineskip}
}
```

Most of the specification consists of nulling the majority of the normal LaTeX specification, and modifying just two elements.

The chapter title (via `\printchaptertitle`) is typeset raggedright using the `\Large` smallcaps fonts. The `\MakeLowercase` macro is used to ensure that the entire title is lowercase before typesetting it. Titles are input in mixed case.

After the title is typeset the `\afterchaptertitle` macro specifies that one line is skipped, a horizontal rule is drawn and then another line is skipped.

20.4.2 Lower level divisions

Section titles are in small caps, lowercase, in the same size as the text font. The titles are numbered, with both the chapter and section number.

The specification is:

```
\setsecheadstyle{\raggedright\scshape\MakeLowercase}
  \setbeforesecskip{-\onelineskip}
  \setaftersecskip{\onelineskip}
```

The macro `\setsecheadstyle` lowercases the title and typesets it small caps.

The default skips before and after titles are rubber lengths but this does not bode well if we are trying to line something up with a particular line of text — the presence of section titles may make slight vertical adjustments to the text lines because of the flexible spacing. So, we have to try and have fixed spacings. A single blank line is used before (`\setbeforesecskip`) and after (`\setaftersecskip`) the title text.

A subsection title, which is the lowest subdivision in the book, is in the italic form of the textfont and is typeset as a numbered non-indented paragraph. The code for this is below.

```
\setsubsecheadstyle{\sethangfrom{\noindent ##1}\raggedright\itshape}
  \setbeforesubsecskip{-\onelineskip}
  \setaftersubsecskip{\onelineskip}
```

As in the redefinition of the `\section` style, there are fixed spaces before and after the title text. The title is typeset (`\setsubsecheadstyle`) raggedright in a normal sized italic font. The macro `\sethangfrom` is used to to redefine the internal `\@hangfrom` macro so that the title and number are typeset as a block paragraph instead of the default hanging paragraph style. Note the use of the double `##` mark for denoting the position of the argument to `\@hangfrom`.

An example book design

20.5 SPECIFYING THE PAGESTYLE

The pagestyle is perhaps the most interesting aspect of the exercise. Instead of the chapter and section titles being put at the top of the pages they are put in the margin starting about seven lines below the top of the typeblock. The folios are put at the bottom of the page aligned with the outside of the typeblock.

As the folios are easy, we'll deal with those first.

```
%% Bringhurst page style
\makepagestyle{bringhurst}
\makeevenfoot{bringhurst}{\thepage}{}{}
\makeoddfoot{bringhurst}{}{}{\thepage}
```

Putting text at a fixed point on a page is typically done by first putting the text into a zero width picture (which as far as LaTeX is concerned takes up zero space) and then placing the picture at the required point on the page. This can be done by hanging it from the header.

We might as well treat the titles so that they will align with any marginal notes, which are `\marginparsep` (17pt) into the margin and `\marginparwidth` (51pt) wide. Earlier in the manual I defined two lengths called `\pwlayi` and `\pwlayii` which are no longer used. I will use these as scratch lengths in performing some of the necessary calculations.

For the recto page headers the picture will be the ⟨*right*⟩ part of the header and for the verso pages the picture will be the ⟨*left*⟩ part of the header, all other parts being empty.

For the picture on the ⟨*right*⟩ the text must be 17pt to the right of the origin, and some distance below the origin. From some experiments, this distance turns out to be the `\headsep` plus the `\topskip` plus 7.3 lines, which is calculated as follows:

```
\setlength{\pwlayi}{\headsep}
\addtolength{\pwlayi}{\topskip}
\addtolength{\pwlayi}{7.3\onelineskip}
```

There is a nifty internal LaTeX macro called `\strip@pt` which you probably haven't heard about, and I have only recently come across. What it does is strip the 'pt' from a following length, reducing it to a plain real number. Remembering that the default `\unitlength` is 1pt we can do the following, while making sure that the current `\unitlength` *is* 1pt:

```
\makeatletter
\newcommand{\bringpicr}[1]{%
  \setlength{\unitlength}{1pt}
```

```
  \begin{picture}(0,0)
    \put(\strip@pt\marginparsep, -\strip@pt\pwlayi){%
      \begin{minipage}[t]{\marginparwidth}
        \raggedright\itshape #1
      \end{minipage}}
  \end{picture}
}
\makeatother
```

Specifying the pagestyle

The new macro `\bringpicr{`⟨*text*⟩`}` puts ⟨*text*⟩ into a `minipage` of width `\marginparwidth`, typeset raggedright in an italic font, and puts the top left of the minipage at the position (`\marginparsep`, `-\pwlayi`) in a zero width picture.

We need a different picture for the ⟨*left*⟩ as the text needs to be typeset raggedleft with the right of the text 17pt from the left of the typeblock. I will use the length `\pwlayii` to calculate the sum of `\marginparsep` and `\marginparwidth`. Hence:

```
\makeatletter
\setlength{\pwlayii}{\marginparsep}
\addtolength{\pwlayii}{\marginparwidth}
\newcommand{\bringpicl}[1]{%
  \setlength{\unitlength}{1pt}
  \begin{picture}(0,0)
    \put(-\strip@pt\pwlayii, -\strip@pt\pwlayi){%
      \begin{minipage}[t]{\marginparwidth}
        \raggedleft\itshape #1
      \end{minipage}}
  \end{picture}
}
\makeatother
```

The new macro `\bringpicl{`⟨*text*⟩`}` puts ⟨*text*⟩ into a `minipage` of width `\marginparwidth`, typeset raggedleft in an italic font, and puts the top left of the minipage at the position (`-(\marginparsep + \marginparwidth)`, `-\pwlayi`) in a zero width picture.

Now we can proceed with the remainder of the pagestyle specification. The next bit puts the chapter and section titles into the `\...mark` macros.

```
\makeatletter
\makepsmarks{bringhurst}{%
  \def\chaptermark##1{\markboth{##1}{##1}}
  \def\sectionmark##1{\markright{##1}}
}
\makeatother
```

Finally, specify the evenhead using `\bringpicl` with the section title as its argument, and the oddhead using `\bringpicr` with the chapter title as its argument.

```
\makeevenhead{bringhurst}{\bringpicl{\rightmark}}{}{}
\makeoddhead{bringhurst}{}{}{\bringpicr{\leftmark}}
```

20.6 CAPTIONS AND THE TOC

The captions to figures and tables are set in a small sans font and are neither named nor numbered, and there is no LoF or LoT. Setting the caption titles in the desired font is simple:

```
\captiontitlefont{\small\sffamily}
```

An example book design

There are two options regarding table and figure captioning: either use the `\legend` command (which produces an anonymous unnumbered title) instead of the `\caption` command, or use the `\caption` command with a modified definition. Just in case the design might change at a later date to required numbered captions, it's probably best to use a modified version of `\caption`. In this case this is simple, just give the `\caption` command the same definition as the `\legend` command.

```
\let\caption\legend
```

An aside: I initially used the default caption style (block paragraph) for the diagram on page 357, but this looked unbalanced so now it has the last line centered. As a float environment, like any other environment, forms a group, you can make local changes within the float. I actually did it like this:

```
\begin{figure}
\captiontitlefont{\small\sffamily}
\captionstyle{\centerlastline}
...
\legend{...} \label{...}
\end{figure}
```

For fine typesetting you may wish to change the style of particular captions. The default style for a single line caption works well, but for a caption with two or three lines either the `centering` or `centerlastline` style might look better. A very long caption is again probably best done in a block paragraph style.

Only chapter titles are included in the ToC. To specify this we use the `\settocdepth` command.

```
\settocdepth{chapter}
```

The ToC is typeset raggedright with no leaders and the page numbers coming immediately after the chapter title. This is specified via:

```
\renewcommand{\cftchapterfont}{\normalfont}
\renewcommand{\cftchapterpagefont}{\normalfont}
\renewcommand{\cftchapterpresnum}{\bfseries}
\renewcommand{\cftchapterleader}{}
\renewcommand{\cftchapterafterpnum}{\cftparfillskip}
```

20.7 PREAMBLE OR PACKAGE?

When making changes to the document style, or just defining a new macro or two, there is the question of where to put the changes — in the preamble of the particular document or into a separate package?

If the same changes/macros are likely to be used in more than one document then I suggest that they be put into a package. If just for the single document then the choice remains open.

I have presented the code in this chapter as though it would be put into the preamble, hence the use of `\makeatletter` and `\makeatother` to surround macros that include the `@` character (see §E.4). The code could just as easily be put into a package called, say, bringhurst. That is, by putting all the code, except for the `\makeatletter` and `\makeatother` commands, into a file called `bringhurst.sty`. It is a good idea also to end the code in the file with `\endinput`; LaTeX stops reading the file at that point and will ignore any possible garbage after `\endinput`.

Preamble or package?

You then use the bringhurst package just like any other by putting

```
\usepackage{bringhurst}
```

in your document's preamble.

Twenty-one

An example thesis design

Many universities in the United States have strict regulations about the typography of theses. The title and administrative pages are inherently specific to a particular university, but often the design for the body of the thesis clashes with normally accepted typographic practice. This chapter presents fairly typical guidelines for some US universities and code intended to meet them. Let's call the university in question the *Archibald Smythe University*, or ASU for short.

The requirements that are listed below are not from any single university but I have, over the years, seen each one of them. In reality there are likely to be many more nit-picking rules than I have shown.

It amuses me that I have never seen a printed set of requirements that followed the rules laid down therein.

Universities outside the US tend to be more relaxed with the result that theses from these establishments are very often more attractive (certainly less bulky) and more readable. The ASU requirements lead to an exceptionally dull and unattractive appearance.

21.1 Example US thesis typographic requirements

21.1.1 General

Paper size The thesis shall be printed on 8.5 by 11 inch plain white paper.

Single-sided The thesis shall be printed single-sided; that is, one side of each sheet of paper will be blank.

Margins Every page of the document shall meet the requirements of a 1.5 inch margin on the left and a 1 inch margin at the top, right, and bottom of the page. Nothing shall appear in any margin.

Fonts The thesis may be set in 10, 11 or 12pt Arial, Century, Garamond, Lucida Bright (10pt only), Tahoma, Times, or Verdana. The same font and size shall be used throughout the thesis.

There shall be no bold type.

Italic type (or underlining) is limited to the names of species, genera, book titles, musical compositions, or foreign words.

Line Spacing All text shall be double-spaced, except material in tables and the optional biographical sketch (which must be single-spaced). You shall single-space individual footnotes and endnotes with a double space between each entry.

21.1.2 Preliminary matter

The preliminary matter consists of the following pages in this order:

1. Title page
2. Approval page
3. Abstract
4. Dedication (optional)
5. Acknowledgements (optional)
6. Table of contents
7. List of tables (if there are any tables)
8. List of figures (if there are any figures)
9. Other lists (e.g., nomenclature, definitions, glossary of terms, etc.)
10. Preface (optional but must be less than ten pages)
11. Under special circumstances further sections may be allowed

The heading for each preliminary page (except the Dedication which shall not have a heading) is centered between the margins, in all capital letters, double-spaced and begin on the first line below the top margin.

The title and approval page are counted as pages one and two, but no page numbers shall appear on them. All subsequent preliminary pages are paginated with lowercase Roman numerals. Starting with 'iii' on the abstract page, place all page numbers at the bottom of the page, centered between the left and right margins and upon the 1 inch bottom margin. Continue numbering consecutively on the subsequent pages up to the first page of the main text.

Title page

1. All text shall be centered between the side margins.
2. Set the title in all capital letters, double-spaced, starting at the top of the page (but below the top margin).
3. On a new line (double-spaced) type 'by' in lowercase letters.
4. On a new line (double-spaced) type your full legal name.
5. At the center of the page type the appropriate description for your degree with the exact wording and line breaks as shown, and single-spaced:

   A _____ Presented in Partial Fulfillment
   of the Requirements for the Degree

   _____

   Replace the blanks with the appropriate wording: Thesis and Master of Arts or Dissertation and Doctor of Philosophy.
6. At the bottom of the page type 'ARCHIBALD SMYTHE UNIVERSITY' in all capitals.
7. Type the month and year of the date you will graduate, with the month in title case and no comma between the month and year.
8. The space between your name and the degree description should equal the space between the degree description and the name of the University.

Approval page

1. All text shall be centered between the side margins.
2. Set the title in all capital letters, double-spaced, starting at the top of the page (but below the top margin).
3. On a new line (double-spaced) type 'by' in lowercase letters.
4. On a new line (double-spaced) type your full legal name in title-cased letters.
5. Add two double-spaced lines (four single-spaced lines) and type 'has been aproved' in lowercase
6. Add a double-space.
7. Type the month and year of your oral defense, with the month in title case and no comma between the month and year.
8. At about the center of the page type 'Graduate Supervisory Committee:'
9. A blank (double-spaced) line
10. Type the members' names, without titles, one per line, single-spaced, as follows:
    a) If you have one chair, type: the chair's name, comma, space 'Chair'
    b) If two chairs, type: comma, space, 'Co-Chair' after the first two names
    c) Follow with the other members' names.
11. At the bottom of the page, type 'ACCEPTED BY THE GRADUATE COLLEGE'
12. The space between the date and 'Graduate Supervisory Committe' lines should equal the space between the last member's name and the 'ACCEPTED…' line.

Abstract page

Center the title 'ABSTRACT' at the top of the page. Number the page at the bottom, centered with the Roman numeral 'iii'. If there is a second page, number it similarly with 'iv'.

Dedication and Acknowledgements (optional)

- The dedication and acknowledgements together must not exceed three pages.
- The dedication page is not titled and the text should be centered both vertically and horizontally.
- The heading for the acknowledgements page is 'ACKNOWLEDGEMENTS', centered and at the top of the page.
- Continue the page numbering in lowercase Roman, at the bottom and centered.

21.1.3 Table of contents

1. Type 'TABLE OF CONTENTS' centered at the top of the page.
2. On the next line type the word 'Page' right justified
3. Begin listing any preliminary pages that follow the table of contents (e.g., lists) in ALL CAPS. The title is left justified, the page number is right justified and a dotted line fills the gap between.

4. Double space between entries.
5. Chapter headings and subheadings to three levels shall be listed, with a lower level being indented with respect to a higher level.
6. The wording of headings shall correspond exactly to those in the main body.
7. The page number is centered at the bottom of the page.
8. If the listing continues for more than one page, subsequent pages shall be headed with one line consisting of 'Chapter' left justified and 'Page' right justified.

21.1.4 Lists

For a given kind of list (often figures or tables) called, say, 'things':

1. Type 'LIST OF THINGS' centered at the top of the page.
2. On the next line type 'Thing' left justified and 'Page' right justified.
3. List, double-spaced, the caption or title of the thing left justified and the page number right justified, with a dotted line between them.
4. Use Roman lowercase to number the page(s) at the bottom, centered.
5. If the listing continues for more than one page, subsequent pages shall be headed with one line consisting of 'Thing' left justified and 'Page' right justified.

21.1.5 Main text

Nothing shall appear in the margins.

The top line on a page is the line immediately below the top margin. The top text line is the one following that (i.e. the second line below the margin).

Page numbering

All pages are counted, but the first page of each chapter is not numbered (paginated); other pages are paginated. The first page of the main text is counted as number 1. Numbered pages have the number right justified on the top line.

Headings

Chapter headings shall be centered. On the top line type 'Chapter' followed by the number. On the top text line type the heading in all uppercase. Type the text on the subsequent lines.

Subheadings, consisting of the number and title (not in all caps), shall be centered, with one blank line before and after.

Captions

Table captions, which are left justified, shall be put before the table itself. The first line consists of 'Table' followed by the number; the caption wording commences on the next line.

Captions for figures are similar, except that they shall be put below the figure and 'Table' replaced by 'Figure'.

Tables and figures shall be single-spaced.

Notes

Notes may be placed at the bottom of the page (i.e., footnotes), or grouped in the backmatter (i.e., endnotes) before the reference list.

All notes shall be introduced by a superior number in the text, with the same number used for the text of the note. Notes should be single spaced, with double spacing between them.

21.1.6 Backmatter

The backmatter consists of the following pages, in order (all of which are optional).

1. Notes (if you are using endnotes and grouping them at the end)
2. References (AKA 'Bibliography' or 'Works Cited')
3. Appendices
4. Biographical sketch (optional)

Pagination continues from the main text; but as with chapters, the first pages of any notes, references, or appendices are not numbered. A biographical sketch, if it is included, is the last page and is neither counted nor paginated.

Headings for the backmatter sections shall be in uppercase, centered, and on the top line.

Notes

The section for endnotes should begin on a new, unnumbered page. Subsequent pages should be numbered.

Use 'NOTES', centered and at the top, as the heading for the notes section.

References

Use the reference heading appropriate for your discipline, in uppercase, centered and at the top of the page. Individual refences should be single-spaced with the second and later lines of a multiline reference indented with respect to the first line. There should be double-spacing between references.

Appendices

The heading for an appendix consists of the word 'APPENDIX' followed by the uppercase letter signifying its position in the sequence of appendices (e.g., A or B or C or ...). This shall be centered on the top line. The title of the appendix, in uppercase, is centered on the following line. This page is not numbered. Subsequent pages are numbered and the text commences on the top text line of the following page.

Biography

The title for the optional biographical page is 'BIOGRAPHICAL SKETCH', in the usual position. The text shall not exceed the one page.

21.2 CODE

Given the above set of requirements we can produce code that, hopefully, will generate documents that will not fall foul of the inspectorate. For simplicity I'll do the code in

the form of a package called `pwasu.sty`. I will be using some LaTeX kernel commands that you won't normally come across. Some of the macros include `@` as part of their name but this is safe as they are in a package, otherwise they would have to be within a `\makeatletter` ... `\makeatother` grouping (see §E.4).

21.2.1 Initialisation

First, identify the package and its basic requirements.

```
%%%% file pwasu.sty
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{pwasu}[2009/04/18 v0.3 ASU thesis]
```

This is only going to work with memoir, so check if the class is being used, and if not then give an error message and stop reading the package code.

```
%% Only works with the memoir class!!!!!!!!!!!!!!!
\@ifclassloaded{memoir}{\let\endpwasu\relax}{\let\endpwasu\endinput
  \PackageError{pwasu}{The pwasu package only works with the memoir class}%
              {\@ehd}}
\endpwasu
```

ASU is very strict about only using a single font in a single size. It is probable that at least one reference will be made to a location on the web. With LaTeX such references are set using the url package, which defaults to using a monospaced font for urls. For ASU we have to make sure that they will be made using the body font (and the same applies to any verbatim text, such as computer code listings, for which we can use the `\setverbatimfont` command).

```
\usepackage{url}
\urlstyle{same}
\setverbatimfont{\normalfont\rmfamily}% make verbatims use the body font
```

Ensuring that footnotes use the body font is a simple matter of redefining `\foottextfont`.

```
\renewcommand*{\foottextfont}{\normalfont\normalsize}
```

Noticing that the requirements can involve switching between double and single spacing, some of which will be done internally in the package, give the user a chance to change the default double spacing. The argument to `\setasuspacing` can be either `\OnehalfSpacing` or `\DoubleSpacing`. The result sets `\AsuSpacing` to be one of these commands.

```
% To enable spacing to be changed if neccesary by the user
\newcommand*{\setasuspacing}[1]{%
  \let\AsuSpacing#1
  \AsuSpacing}
\setasuspacing{\DoubleSpacing}
```

21.2.2 Page layout

As this is for an American University on letterpaper sized paper we'll assume letterpaper stock, so no cropping will be needed. Setting the side margins is easy:

```
% left, right margins and textwidth
\setlrmarginsandblock{1.5in}{1in}{*}
```

Setting the top and bottom margins requires more thought. LaTeX provides the header and footer areas for page numbers. However, the requirements state that page numbers must be either on the top or bottom line (of the textblock) with the text extending down to the bottom line or up to the top line (of the textblock). I'll organise it according to the layout for the main body. Here, the top of the header is 1 inch below the top of the paper and the bottom of the text is 1 inch above the bottom of the paper (with the footer below that).

```
%% for main body, bottom of text at 1in, footer below
%% top of header at 1in, first text line double spaced
%% below base of header
\newlength{\linespace}
\setlength{\linespace}{\baselineskip} % current equivalent of \onelineskip
\setlength{\headheight}{\onelineskip}
\setlength{\headsep}{\linespace}
\addtolength{\headsep}{-\topskip}
\setlength{\uppermargin}{1in}
\addtolength{\uppermargin}{\headheight}
\addtolength{\uppermargin}{\headsep}
```

And for the bottom margin:

```
%% and for the bottom
\setlength{\lowermargin}{1in}
\setlength{\textheight}{\paperheight}
\addtolength{\textheight}{-\uppermargin}
\addtolength{\textheight}{-\lowermargin}
```

And finally for footnotes:

```
%% footnote settings
\setlength{\footskip}{\onelineskip}
\setlength{\footnotesep}{\onelineskip}
```

The layout on the preliminary pages is different, so I'll need some handy lengths in order to change the layout as appropriate. The user can also make adjustments with these if necessary.

```
%% the fiddle lengths (..ta.. for title/approval page, others for prelims)
\newlength{\toptafiddle} \setlength{\toptafiddle}{2\linespace}
\newlength{\bottafiddle} \setlength{\bottafiddle}{0pt}
\newlength{\topfiddle}   \setlength{\topfiddle}{\toptafiddle}
\newlength{\botfiddle}   \setlength{\botfiddle}{\onelineskip}
```

That's it for the general layout, except for increasing the paragraph indentation as the line spacing is larger than normal.

```
\setlength{\parindent}{2em}
\checkandfixthelayout[nearest]
```

As the layout is set up, the bottom of the text is one inch above the bottom of the paper. This is fine for the main text and the Title and Approval pages but the text height must be decreased, temporarily, for the other pages in the prelims. Changes to the page layout may be accomplished by the following sneaky procedure. First change from one- to two-column (or vice-versa), which starts a new page, make the changes, then change from two- to one-column (or vice-versa) which starts the same new page again but with the layout changes implemented. The following code implements this for the `\textheight` in a one column document, which is what we are dealing with here.

```
\newcommand*{\addtotextheight}[1]{%
  \twocolumn
  \addtolength{\textheight}{#1}%
  \onecolumn}
```

21.2.3 Page styles

Next I'll tackle the page styles. The style for the main body is simple, with the page number top right, and the *empty* style is the one for chapter pages. The page number for the preliminary pages is centered at the bottom, and the *plain* page style provides that. For the main text define the *asu* page style.

```
%%%% pagestyles
%% the main text
\makepagestyle{asu}
  \makeevenhead{asu}{\thepage}{}{}
  \makeoddhead{asu}{}{}{\thepage}
```

Any 'continuation' pages for the ToC, etc., have a header that consists of a name at the left and the word 'Page' at the right. We need a header for each kind of listing.

```
%% for continuation pages of the ToC, LoF, LoT
\makepagestyle{toc}
  \makeevenfoot{toc}{}{\thepage}{}
  \makeoddfoot{toc}{}{\thepage}{}
  \makeevenhead{toc}{Chapter}{}{Page}
  \makeoddhead{toc}{Chapter}{}{Page}
\makepagestyle{lof}
  \makeevenfoot{lof}{}{\thepage}{}
  \makeoddfoot{lof}{}{\thepage}{}
  \makeevenhead{lof}{Figure}{}{Page}
  \makeoddhead{lof}{Figure}{}{Page}
\makepagestyle{lot}
  \makeevenfoot{lot}{}{\thepage}{}
  \makeoddfoot{lot}{}{\thepage}{}
  \makeevenhead{lot}{Table}{}{Page}
  \makeoddhead{lot}{Table}{}{Page}
```

21.2.4 The ToC and friends

While we're at it, do the code for the LoF and LoT, which is simpler than that needed for the ToC. We have to specify our new pagestyles which can be done by extending the `\listof...` macros. The `\addtodef` macro makes this rather easy.

```
%%%% The LoF
\renewcommand{\listfigurename}{LIST OF FIGURES}
\addtodef{\listoffigures}{\clearpage\pagestyle{lof}}{}
```

For the titles, these have to be moved up into the header area, so that they come just below the top margin, and set the initial pagestyle as *plain*. After the title we can insert the relevant column headers.

```
\renewcommand*{\lofheadstart}{\vspace*{-\topfiddle}}
\renewcommand*{\afterloftitle}{\thispagestyle{plain}%
  \par\nobreak {\normalfont Figure \hfill Page}\par\nobreak}
```

And the same for the LoT.

```
%%% The LoT
\renewcommand{\listtablename}{LIST OF TABLES}
\addtodef{\listoftables}{\clearpage\pagestyle{lot}}{}
\renewcommand*{\lotheadstart}{\vspace*{-\topfiddle}}
\renewcommand*{\afterlottitle}{\thispagestyle{plain}%
  \par\nobreak {\normalfont Table \hfill Page}\par\nobreak}
```

The ToC is similar but we also have to deal with the entries themselves.

```
%%%% Do the ToC
\renewcommand{\contentsname}{TABLE OF CONTENTS}
\addtodef{\tableofcontents}{\clearpage\pagestyle{toc}}{}
\renewcommand*{\tocheadstart}{\vspace*{-\topfiddle}}
\renewcommand*{\aftertoctitle}{\thispagestyle{plain}%
  \par\nobreak \mbox{}\hfill{\normalfont Page}\par\nobreak}
```

And the changes to the entries, all of which are set in the normal font and with dotted leaders, with no extra space between any of the entries.

```
\renewcommand*{\cftchapterfont}{\normalfont}
\renewcommand*{\cftchapterpagefont}{\normalfont}
\renewcommand*{\cftchapterleader}{%
  \cftchapterfont\cftdotfill{\cftchapterdotsep}}
\renewcommand*{\cftchapterdotsep}{\cftdotsep}
```

Unlike the typical LaTeX ToC there must be no additional space before chapter entries; also, there should be no additional space inserted by the chapters in the LoF or LoT which just requires a simple redefinition of `\insertchapterspace`..

```
%%% no extra space before the entry, or in the LoF/LoT
\setlength{\cftbeforechapterskip}{0pt plus 0pt}
\renewcommand*{\insertchapterspace}{}
```

21.2.5 Chapter styling

Moving on to styling the chapter titles, the first line must be moved up into the header area, and other spacings set to give blank lines. The fonts are just the regular body font. Call the chapterstyle *asu* and make sure that the *empty* pagestyle is used for it.

```
%% chapter style
\makechapterstyle{asu}{%
  \setlength{\beforechapskip}{-\topfiddle}
  \setlength{\midchapskip}{1.0\onelineskip}
  \setlength{\afterchapskip}{1.0\onelineskip}
  \renewcommand*{\chapnamefont}{\normalfont}
  \renewcommand*{\chapnumfont}{\chapnamefont}
  \renewcommand*{\printchapternum}{\centering\chapnumfont \thechapter}
  \renewcommand*{\chaptitlefont}{\normalfont\centering}
  \renewcommand*{\printchapternonum}{}}
\aliaspagestyle{chapter}{empty}
```

The chapterstyle for any appendices is slightly different as the title, all in uppercase, is on one page by itself and the text starts on the following page. Call this the *asuappendix* chapterstyle.

```
%% chapter style for appendices, text comes on following page
\makechapterstyle{asuappendix}{%
  \setlength{\beforechapskip}{-\topfiddle}
  \setlength{\midchapskip}{1.0\onelineskip}
  \setlength{\afterchapskip}{1.0\onelineskip}
  \renewcommand*{\chapnamefont}{\normalfont}
  \renewcommand*{\chapnumfont}{\chapnamefont}
  \renewcommand*{\printchaptername}{%
    \chapnamefont\MakeUppercase{\@chapapp}}
  \renewcommand*{\printchapternum}{\centering\chapnumfont \thechapter}
  \renewcommand*{\chaptitlefont}{\normalfont\centering}
  \renewcommand*{\printchapternonum}{}
  \renewcommand*{\afterchaptertitle}{\clearpage}}
```

We have to extend the `\appendix` command to use the new chapter style, and also to ensure that double spacing will be used (certain elements that come before the appendices are single spaced).

```
%%% different chapter style for appendices, (and double spaced)
\addtodef{\appendix}{}{\chapterstyle{asuappendix}\AsuSpacing}
```

21.2.6 Section, etc., styling

Set up the section headings so that they are centered, use the normal font, and have a blank line before and after.

```
%%% (subsub)section styles
\setsecheadstyle{\centering\normalfont}
```

```
\setbeforesecskip{-1\onelineskip plus -1ex minus -.2ex}
\setaftersecskip{1\onelineskip plus .2ex}
\setsubsecheadstyle{\centering\normalfont}
\setbeforesubsecskip{-1\onelineskip plus -1ex minus -.2ex}
\setaftersubsecskip{1\onelineskip plus .2ex}
```

21.2.7 Captions

The captions are set flushleft and raggedright with the name and number on one line and the title on the following line. Fortunately floats are automatically set single spaced, which is what the requirements specify.

```
%% Captions
\captiontitlefont{\normalfont}% title font
\precaption{\raggedright}% for Caption N
\captiondelim{\newline}% newline
\captionstyle{\raggedright}% for title
\setlength{\belowcaptionskip}{\onelineskip}
```

21.2.8 The bibliography

The requirements imply that the title is likely to be 'REFERENCES'. The bibliography is set single spaced but with a blank line between the entries.

```
%% for REFERENCE section
\renewcommand*{\bibname}{REFERENCES}
\setlength{\bibitemsep}{\onelineskip}
```

The second and later lines of any entry are to be indented. We use the `\biblistextra` hook for setting this up.

```
\renewcommand*{\biblistextra}{%
  \setlength{\itemsep}{\bibitemsep}
  \setlength{\labelwidth}{0pt}
  \setlength{\leftmargin}{3em}% hanging indent
  \setlength{\itemindent}{-\leftmargin}}
```

The title for the bibliography is set via the `\bibsection` macro. The heading is unnumbered but is added to the ToC. To get the spacing right the heading, set as a `\chapter*`, which must be called double spaced, and then single spacing is called for after that.

```
\renewcommand*{\bibsection}{%
  \AsuSpacing
  \chapter*{\bibname}\addcontentsline{toc}{chapter}{\bibname}
  \SingleSpacing}
```

21.2.9 End notes

The heading for the Notes section is similar to the bibliography heading.

```
%% endnotes
\renewcommand*{\notesname}{NOTES}
\renewcommand*{\notedivision}{%
```

```
\AsuSpacing
\chapter*{\notesname}
\addcontentsline{toc}{chapter}{\notesname}
\SingleSpacing}
```

The rest of the code for endnotes ensures that they are numbered continuously throughout the text, the number is set as a superscript, that there is a blank line between each entry, and that there are no subdivisions within the listing.

```
\continuousnotenums
\renewcommand*{\notenuminnotes}[1]{\textsuperscript{#1}\space}
\renewcommand{\noteinnotes}[1]{#1\\}
\renewcommand*{\pagenotesubhead}[3]{}% no subheads
```

21.2.10 Preliminary headings

There can be any number of sections in the prelims. The titles for these are located in the LaTeX header area. Here's a general macro for setting these.

```
%%% general macro for Abstract, etc., headings
\newcommand*{\pretoctitle}[1]{{\clearpage\centering
  \vspace*{-\topfiddle}#1\par}}
%%% Start the ACKNOWLEDGEMENTS
\newcommand{\asuacknowledgements}{\pretoctitle{ACKNOWLEDGEMENTS}}
```

The Abstract is the first section after the title and approval pages. At this point we must reduce the textheight in order to raise the footer area.

```
%%% Start the ABSTRACT
\newcommand{\asuabstract}{%
  \addtotextheight{-\botfiddle}%
  \pretoctitle{ABSTRACT}}
```

While we are at this, the textheight must be reset to its default value just before the first chapter in the main matter. A simple addition to `\mainmatter` handles this.

```
\addtodef{\mainmatter}{\addtotextheight{\botfiddle}{}
```

The dedication, if any, does not have a heading and the text is centered horizontally and vertically.

```
%% make it easy to center any dedication
\newcommand{\asudedication}[1]{%
  {\clearpage\mbox{}\vfill\centering #1 \par\vfill\clearpage}}
```

There may be sections in the prelims that come after the ToC, and the titles of these are added to the ToC.

```
%% for any headings after the tocloft and before the main body
\newcommand{\prelimtitle}[1]{%
  \pretoctitle{#1}\addcontentsline{toc}{chapter}{#1}}
```

21.2.11 Components of the title and approval pages

There are several items that are set on the title and approval pages. In order to separate the information from the particular layout, I've defined a macro for defining each item.

```
%%%% for the title page and approval page.
% your title
\newcommand{\settitle}[1]{\def\asutitle{#1}}
% you
\newcommand{\setauthor}[1]{\def\asuauthor{#1}}
% document type (e.g., thesis)
\newcommand{\setdoctype}[1]{\def\asudoctype{#1}}
% possible degree
\newcommand{\masters}{\def\asudegree{Master of Arts}}
\newcommand{\doctors}{\def\asudegree{Doctor of Philosophy}}
% defence date
\newcommand{\setdefdate}[1]{\def\asudefdate{#1}}
% graduation date
\newcommand{\setgraddate}[1]{\def\asugraddate{#1}}
% committe chair
\newcommand{\setchair}[1]{\def\asuchair{#1, Chair}}
% committe co-chairs
\newcommand{\setchairs}[2]{%
  \def\asuchair{#1, Co-chair \\ #2, Co-chair}}
% other members (separated by \\s)
\newcommand{\setmembers}[1]{\def\asumembers{#1\par}}
```

Just for fun, create some default settings for these. The successful user will have changed them all!

```
%% Use them like this, and if you don't change them you will
%% get unacceptable title and/or approval pages
\settitle{AN INCREDIBLE PIECE OF WORK OVER WHICH I HAVE STRUGGLED
DAY AND NIGHT FOR FAR TOO LONG AND NOW IT IS OVER}
\setauthor{A. N. Author}
\setdoctype{Polemic}
\masters  % going for a Masters degree
%% \doctors % going for a PhD
\setdefdate{April 2018}
\setgraddate{May 2021}
% \setchair{A. Jones}           % this one
\setchairs{A. Jones}{B. Doe}  % or this one
\setmembers{C. Smith \\ D. Somebody \\ Some One Else \\ Yet Another}
```

21.2.12 The title and approval pages

An example of a title page is shown in Figure 21.1 and an example of the corresponding approval page is in Figure 21.2.

CONTINUING CONUNDRUMS IN THE TYPOGRAPHIC

REQUIREMENTS FOR THESES

by

Ima A. Student

A Polemic Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Arts

ARCHIBALD SMYTHE UNIVERSITY

May 2021

Figure 21.1: Example Archibald Smythe University title page

CONTINUING CONUNDRUMS IN THE TYPOGRAPHIC

REQUIREMENTS FOR THESES

by

Ima A. Student

has been approved

April 2018

Graduate Supervisory Committee:

S. Holmes, Co-Chair
J. Moriarty, Co-Chair
G. E. Challenger
A. Quartermain
J. H. Watson

ACCEPTED BY THE GRADUATE COLLEGE

Figure 21.2: Example Archibald Smythe University approval page

Now we can set up the layouts for the title and approval pages. The information typeset on these pages is obtained from the previous set of commands. Note that the last line on each of these pages has to be set upon the bottom margin. The ASU inspectorate is likely to be very keen on this, perhaps using a ruler to measure the actual distance from the bottom of the page to ensure that it is the magic 1 inch. I have included an `\enlargethispage` by the amount `\bottafiddle` so the user can make any fine adjustment that might be requested. Similarly, the length `\toptafiddle` may be altered to fine tune the position of the title. Hopefully, neither of these tweaks will be needed, but if so, then use `\addtolength` instead of `\setlength` to make an adjustement.

```
%%% typesets the TITLE page
\newcommand{\thetitlepage}{{%
  \clearpage
  \thispagestyle{empty}
  \centering
  \vspace*{-\toptafiddle}
  \asutitle \\ by \\ \asuauthor
  \vfill

  {\SingleSpace
  A \asudoctype\ Presented in Partial Fulfillment \\
  of the Requirements for the Degree \\
  \asudegree\par}

  \vfill
  ARCHIBALD SMYTHE UNIVERSITY \\
  \asugraddate
  \par
  \enlargethispage{\bottafiddle}
  \clearpage}}
```

And similary for the approval page.

```
%%% typesets the APPROVAL page
\newcommand{\approvalpage}{{%
  \thispagestyle{empty}
  \centering
  \vspace*{-\toptafiddle}
  \asutitle \\ by \\ \asuauthor \\[3\onelineskip]
  has been approved \\
  \asudefdate

  \vfill

  Graduate Supervisory Committee: \\[-0.5\onelineskip]
  {\SingleSpacing
  \asuchair \\
  \asumembers}
```

```
\vfill
ACCEPTED BY THE GRADUATE COLLEGE
\par
\enlargethispage{\bottafiddle}
\clearpage}}
```

21.2.13 The last bits

The biographical sketch has a title (which is not added to the ToC), the text is single spaced and there is no page number. It is easiest to provide this as an environment.

```
%%% put your biographical text in this environment
%% \begin{biosketch} I'm a person who has accomplished .... \end{biosketch}
\newenvironment{biosketch}{%
  \pretoctitle{BIOGRAPHICAL SKETCH}\thispagestyle{empty}\SingleSpacing}%
  {}
```

Make sure that the requisite initial page style and appropriate chapter style is used.

```
%% use the asu chapterstyle and plain pagestyle
\chapterstyle{asu}
\pagestyle{plain}

%%%%%%%%%%%%%%% end of *.sty file
\endinput
%%%%%%%%%%%%%%%
```

21.3 USAGE

This is a sketch of how an ASU thesis could be written.

With the wide textblock, 12pt is too small for reading ease, so best not to use 11pt or 10pt.

Times Roman comes with LaTeX, but you are effectively writing a book, not a newspaper column. If you have Garamond or Lucida Bright then seriously consider using one or other of them.[1] Lucida Bright is probably more appropriate if your thesis includes mathematics while Garamond is perhaps more in keeping if your thesis topic falls into the humanities area. If the requirements did not limit your choices then there are other fonts that might better express your work. In any case I suggest that you do not use a sans font (e.g., Arial, Tahoma or Verdana from the ASU list).

```
\documentclass[oneside,12pt]{memoir}
\usepackage{mathptmx}  % Times New Roman
\usepackage{pwasu}     % the package
```

1 Garamond is a commercial font and, for example, is available along with many other fonts from FontSite (http://www.fontsite.com) with LaTeX support from http://contrapunctus.net/fs500tex. Lucida Bright, another commercial font, is available from TUG at http://tug.org/store/lucida and is supported by several LaTeX packages.

The general sequence in your document, after you have set the data for the TITLE and APPROVAL pages and any other specifics and packages in the preamble, is:

```
%  if you can get away without the default \DoubleSpacing, then
%\setasuspacing{\OnehalfSpacing}
%% if you use endnotes, then
   \makepagenote
\begin{document}
\maxtocdepth{subsection} % put 3 levels into the ToC
\frontmatter
\thetitlepage
\approvalpage
\asuabstract
   abstract text
%% if you have any acknowledgements, then
  \asuacknowledgements
    acknowledgements text
% \asudedication{ text } % if you want a dedication
\tableofcontents
% \listoffigures  % if you have any figures
% \listoftables   % if you have any tables
%%% if you have more prelim sections, then
%%% \clearpage
%%% \pagestyle{plain}
%%% \prelimtitle{title} text % for each section before main text
\mainmatter
\pagestyle{asu}
\chapter{...} % start of your main text
... report on lots of incredible work, now you are on your
own until...

%% if endnotes then
   \printpagenotes
%% if a bibliography then
   \begin{thebibliography}...\end{thebibliography}
%% if appendices, then
  \appendix
  \chapter{...}
  ...
%% if Biographical sketch then
   \begin{biosketch} ... \end{biosketch}
\end{document}
```

If you actually try any of the above code and it does not work, then I may have made a typo or two, or maybe you have, or perhaps we both have. In any event, the code is more of a sketch of what might be needed than a prescription of how it must be done.

21.4 COMMENTS

Having read through the requirements you will have realised that whatever committees set them had not advanced beyond the 19th century technology of the typewriter.[2] When I wrote my thesis some forty years ago it was, of necessity, single sided so that carbon copies could be made by the typist (who would have objected strongly to having to type the hundred and fifty or so pages six times). I must admit, though, that the sixth copy was almost too faint and blurry to be read comfortably even though the typist had used thin paper and kept replacing the carbon paper. In this day of double sided printers and double sided copiers I see no reason except inertia to keep a single sided requirement. Many students, and faculty members, have beaten their heads against the diehards and very rarely have they managed to prevail.

In contrast to the ASU style I have a copy of a doctoral thesis [Sne04] for Vrije Universiteit, Amsterdam. This is a professionally printed 100 page, double sided, glossy paperbound book with an attractive coloured photograph of a sunset on the front and rear covers. The page size is 40.5pc by 57pc with spine and top margins of 5pc and foredge and bottom margins of 7pc.[3] The textblock, then, is 28.5 by 45pc set with 45 lines of a 10pt Lucida Bright seriffed font. Chapter and section heads are in a sans font, with the chapter heads larger than the section heads. Caption names are a bold sans with the caption text in an italic. Headers on the verso pages are the chapter title with the section title as the recto header. The page numbers are in the footers by the foredge margin. Altogether, a much more appealing production than Archibald Smythe University will permit.

2 Remington sold their first commercial typewriter in 1873 which even then had the QWERTY keyboard layout. By 1910 typewriter designs were pretty well standardised.

3 Professional printers use points and picas for their measurements.

Appendices

A

Packages and macros

The memoir class does not provide for everything that you have seen in the manual. I have used some packages that you are very likely to have in your LaTeX distribution, and have supplemented these with some additional macros, some of which I will show you.

A.1 Packages

The packages that I have used that you are likely to have, and if you do not have them please consider getting them, are:

- etex lets you take advantage of eTeX's extended support for counters and such.
  Note that from 2015 and onwards, the allocation of extra registers have now been build into the LaTeX kernel. Thus in most cases the etex package is no longer necessary. There are how ever extra very special features left in etex that *some* users may need. In that case please remember to load etex by placing `\RequirePackage{etex}` *before* `\documentclass`!
- url [Ars99] is for typesetting URL's without worrying about special characters or line breaking.
- fixltx2e [MC00] eliminates some infelicities of the original LaTeX kernel. In particular it maintains the order of floats on a twocolumn page and ensures the correct marking on a twocolumn page.
  Note that as of 2015, the functionality of this package has been merged into the LaTeX kernel. Loading this package does nothing.
- alltt [Bra97] is a basic package which provides a verbatim-like environment but \, {, and } have their usual meanings (i.e., LaTeX commands are not disabled).
- graphicx [CR99] is a required package for performing various kinds of graphical functions.
- color [Car05] is a required package for using color, or xcolor [Ker07] is an enhanced version of color.
- latexsym gives access to some extra symbols.
- amsmath for when you are doing anything except the simplest kind of maths typeseting.
- fontenc for using fonts with anything other than the original OT1 encoding (i.e., for practically any font).
- pifont for typesetting Pifonts (i.e., Symbol and Zapf Dingbats)

Apart from the packages that are supplied as part of the memoir distribution, the packages that I used and you most likely do not have are:

- layouts [Wil04c]. I used it for all the layout diagrams. For example, Figure 6.2 and Figure 6.3 were drawn simply by:

```
\begin{figure}
\centering
\setlayoutscale{1}
\drawparameterstrue
\drawheading{}
\caption{Displayed sectional headings} \label{fig:displaysechead}
\end{figure}

\begin{figure}
\centering
\setlayoutscale{1}
\drawparameterstrue
\runinheadtrue
\drawheading{}
\caption{Run-in sectional headings} \label{fig:runsechead}
\end{figure}
```

  The package also lets you try experimenting with different layout parameters and draw diagrams showing what the results would be in a document.

  The version of layouts used for this manual is v2.4 dated 2001/04/30. Earlier versions will fail when attempting to draw some figures (e.g., to draw Figure 2.3).

- fonttable [Wil09a]. I used this for the font tables (e.g., Table 3.2). You must have at least version 1.3 dated April 2009 for processing the manual (earlier versions are likely to produce errors in the number formatting area with minor, but odd looking, effect on the printed result).

A.2 MACROS

Originally the preamble of the manual contained many macro definitions, probably more than most documents would because:

- I am having to typeset many LaTeX commands, which require some sort of special processing;
- I have tried to minimize the number of external packages needed to LaTeX this manual satisfactorily, and so have copied various macros from elsewhere;
- I wanted to do some automatic indexing;
- I wanted to set off the syntax specifications and the code examples from the main text.

I have since put the majority of these into a package file called `memsty.sty`. To get the whole glory you will have to read the preamble, and the memsty package file but I show a few of the macros below as they may be of more general interest.

`\Ppstyle{`⟨*pagestyle*⟩`}` `\pstyle{`⟨*pagestyle*⟩`}`

The command \Ppstyle prints its argument in the font used to indicate pagestyles and the command \pstyle prints its pagestyle argument and also makes a pagestyle entry in the index. Its definition is

```
\newcommand*{\pstyle}[1]{\Ppstyle{#1}%
  \index{#1 pages?\Ppstyle{#1} (pagestyle)}%
  \index{pagestyle!#1?\Ppstyle{#1}}}
```

The first part prints the argument in the text and the second adds two entries to the idx file. The fragment #1 pages is what the MakeIndex program will use for sorting entries, and the fragment following the ? character is what will be put into the index.

\Pcstyle{⟨*chapterstyle*⟩} \cstyle{⟨*chapterstyle*⟩}

The command \Pcstyle prints its argument in the font used to indicate chapterstyles and \cstyle prints its chapterstyle argument and also makes a chapterstyle entry in the index. Its definition is

```
\newcommand*{\cstyle}[1]{\Pcstyle{#1}%
  \index{#1 chaps?\Pcstyle{#1} (chapterstyle)}%
  \index{chapterstyle!#1?\Pcstyle{#1}}}
```

which is almost identical to \pstyle.

There is both a *companion* chapterstyle and a *companion* pagestyle. The strings used for sorting the index entries for these are companion chaps and companion pages respectively, so the chapterstyle will come before the pagestyle in the index. The reason for distinguishing between the string used for sorting and the actual entry is partly to distinguish between different kinds of entries for a single name and partly to avoid any formatting commands messing up the sorted order.

\begin{syntax} syntax \end{syntax}

The syntax environment is for specifying command and environment syntax. Its definition is

```
\newcommand*{\tightcenter}{%
  \topsep=0.25\onelineskip\trivlist \centering\item\relax}
\def\endtigthcenter{\endtrivlist}
\newenvironment{syntax}{\begin{tightcenter}
                         \begin{tabular}{|p{0.9\linewidth}|} \hline}%
                        {\hline
                         \end{tabular}
                         \end{tightcenter}}
```

It is implemented in terms of the tabular environment, centered within the typeblock, which forms a box that will not be broken across a pagebreak. The box frame is just the normal horizontal and vertical lines that you can use with a tabular. The width is fixed at 90% of the text width. As it is a tabular environment, each line of syntax must be ended with \\. Note that normal LaTeX processing occurs within the syntax environment, so you can effectively put what you like inside it. The center environment is defined in terms of a trivlist and \centering. I wanted to be able to control the space before and after the '\centering' so I defined the tightcenter environment which enabled me to do this.

\begin{lcode} LaTeX code \end{lcode}

I use the `lcode` environment for showing examples of LaTeX code. It is a special kind of `verbatim` environment where the font size is `\small` but the normal `\baselineskip` is used, and each line is indented.

At the bottom the environment is defined in terms of a `list`, although that is not obvious from the code; for details see the class code [Wil19a]. I wanted the environment to be a tight list and started off by defining two helper items.

```
% \@zeroseps sets list before/after skips to minimum values
\newcommand*{\@zeroseps}{\setlength{\topsep}{\z@}
                         \setlength{\partopsep}{\z@}
                         \setlength{\parskip}{\z@}}
% \gparindent is relative to the \parindent for the body text
\newlength{\gparindent} \setlength{\gparindent}{0.5\parindent}
```

The macro `\@zeroseps` sets the before, after and middle skips in a list to 0pt (`\z@` is shorthand for 0pt). The length `\gparindent` will be the line indentation in the environment.

```
% Now we can do the new lcode verbatim environment.
% This has no extra before/after spacing.
\newenvironment{lcode}{\@zeroseps
  \renewcommand{\verbatim@startline}%
    {\verbatim@line{\hskip\gparindent}}
  \small\setlength{\baselineskip}{\onelineskip}\verbatim}%
  {\endverbatim
   \vspace{-\baselineskip}\noindent}
```

The fragment `{\hskip\gparindent}` puts `\gparindent` space at the start of each line.

The fragment `\small\setlength{\baselineskip}{\onelineskip}` sets the font size to be `\small`, which has a smaller `\baselineskip` than the normal font, but this is corrected for by changing the local `\baselineskip` to the normal skip, `\onelineskip`. At the end of the environment there is a negative space of one line to compensate for a one line space that LaTeX inserts.

B

Showcases

The memoir memoir class has several features that involve a *style* and it provide several of these styles. This chapter is used to showcase these styles.

B.1 Chapter styles

For more about defining chapter styles, see section 6.5, page 84.

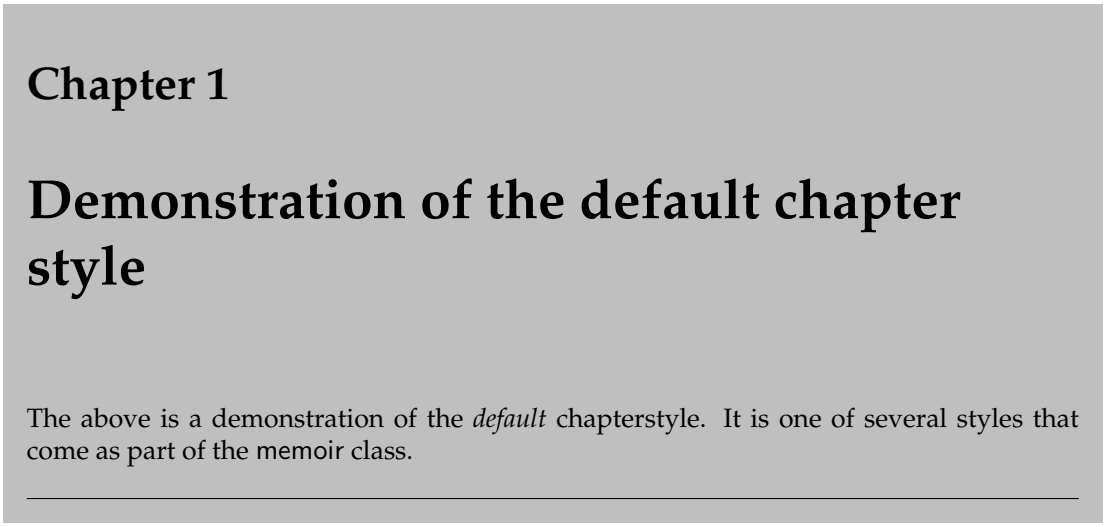


Figure B.1: The default chapterstyle

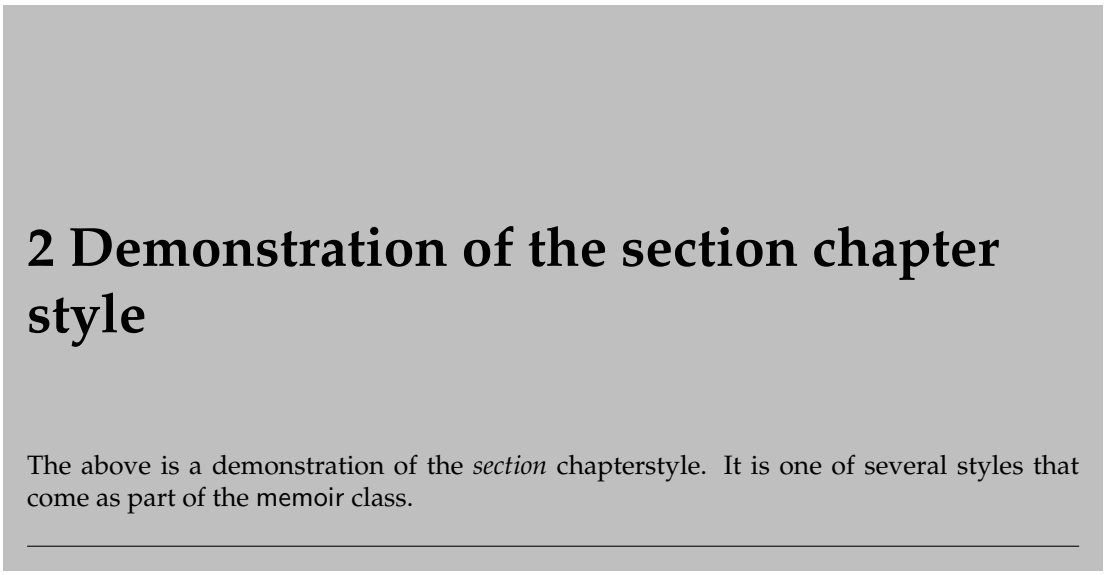


Figure B.2: The section chapterstyle

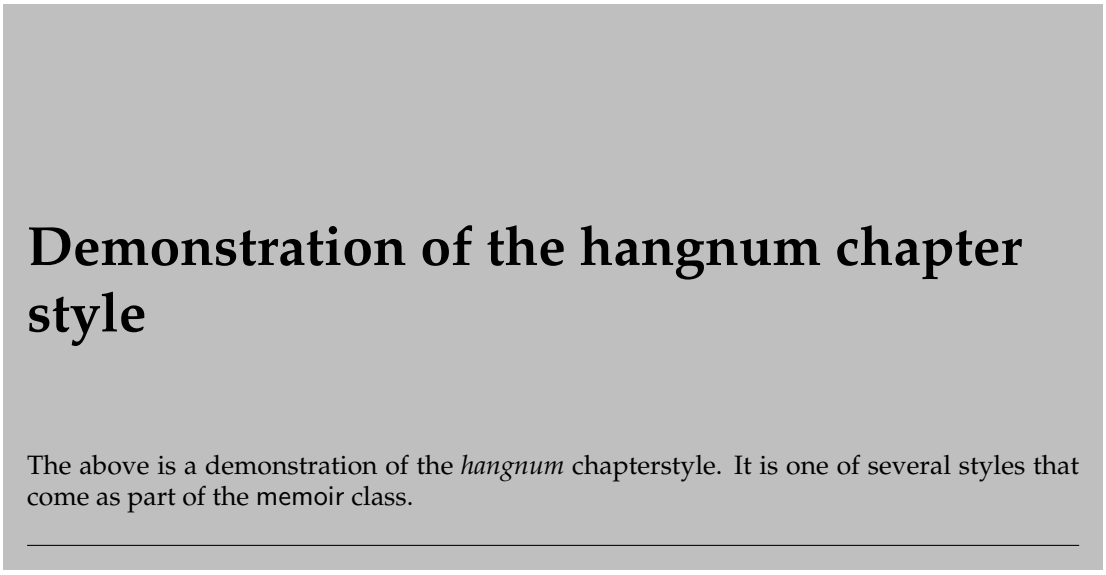


Figure B.3: The hangnum chapterstyle

CHAPTER 4

Demonstration of the companion chapter style

The above is a demonstration of the *companion* chapterstyle. It is one of several styles that come as part of the memoir class.

Figure B.4: The companion chapterstyle

5 Demonstration of the article chapter style

The above is a demonstration of the *article* chapterstyle. It is one of several styles that come as part of the memoir class.

Figure B.5: The article chapterstyle

Chapter 6

Demonstration of the bianchi chapter style

The above is a demonstration of the *bianchi* chapterstyle. It is one of several styles that come as part of the memoir class.

Figure B.6: The bianchi chapterstyle

DEMONSTRATION OF THE BRINGHURST CHAPTER STYLE

The above is a demonstration of the *bringhurst* chapterstyle. It is one of several styles that come as part of the memoir class.

Figure B.7: The bringhurst chapterstyle

Chapter 8

Demonstration of the brotherton chapter style

The above is a demonstration of the *brotherton* chapterstyle. It is one of several styles that come as part of the memoir class.

Figure B.8: The brotherton chapterstyle

Chapter 9

Demonstration of the chappell chapter style

The above is a demonstration of the *chappell* chapterstyle. It is one of several styles that come as part of the memoir class.

Figure B.9: The chappell chapterstyle

10 Demonstration of the crosshead chapter style

The above is a demonstration of the *crosshead* chapterstyle. It is one of several styles that come as part of the memoir class.

Figure B.10: The crosshead chapterstyle

I Demonstration of the culver chapter style

The above is a demonstration of the *culver* chapterstyle. It is one of several styles that come as part of the memoir class.

Figure B.11: The culver chapterstyle

— 2 —

Demonstration of the dash chapter style

The above is a demonstration of the *dash* chapterstyle. It is one of several styles that come as part of the memoir class.

Figure B.12: The dash chapterstyle

3

Demonstration of the demo2 chapter style

The above is a demonstration of the *demo2* chapterstyle. It is one of several styles that come as part of the memoir class.

Figure B.13: The demo2 chapterstyle

Chapter 4

Demonstration of the dowding chapter style

The above is a demonstration of the *dowding* chapterstyle. It is one of several styles that come as part of the memoir class.

Figure B.14: The dowding chapterstyle

5

Demonstration of the ell chapter style

The above is a demonstration of the *ell* chapterstyle. It is one of several styles that come as part of the memoir class.

Figure B.15: The ell chapterstyle

Chapter 6

Demonstration of the ger chapter style

The above is a demonstration of the *ger* chapterstyle. It is one of several styles that come as part of the memoir class.

Figure B.16: The ger chapterstyle

7 Demonstration of the komalike chapter style

The above is a demonstration of the *komalike* chapterstyle. It is one of several styles that come as part of the memoir class.

Figure B.17: The komalike chapterstyle

CHAPTER 8

Demonstration of the lyhne chapter style

The above is a demonstration of the *lyhne* chapterstyle. It is one of several styles that come as part of the memoir class.

Figure B.18: The lyhne chapterstyle. This style requires the graphicx package

CHAPTER 9

Demonstration of the madsen chapter style

The above is a demonstration of the *madsen* chapterstyle. It is one of several styles that come as part of the memoir class.

Figure B.19: The madsen chapterstyle. This style requires the graphicx package

Chapter 10

Demonstration of the ntglike chapter style

The above is a demonstration of the *ntglike* chapterstyle. It is one of several styles that come as part of the memoir class.

Figure B.20: The ntglike chapterstyle

1 Demonstration of the southall chapter style

The above is a demonstration of the *southall* chapterstyle. It is one of several styles that come as part of the memoir class.

Figure B.21: The southall chapterstyle

2 Demonstration of the tandh chapter style

The above is a demonstration of the *tandh* chapterstyle. It is one of several styles that come as part of the memoir class.

Figure B.22: The tandh chapterstyle

CHAPTER 3

DEMONSTRATION OF THE THATCHER CHAPTER STYLE

The above is a demonstration of the *thatcher* chapterstyle. It is one of several styles that come as part of the memoir class.

Figure B.23: The thatcher chapterstyle

CHAPTER 4

Demonstration of the veelo chapter style

The above is a demonstration of the *veelo* chapterstyle. It is one of several styles that come as part of the memoir class.

Figure B.24: The veelo chapterstyle. This style requires the graphicx package

5. Demonstration of the verville chapter style

The above is a demonstration of the *verville* chapterstyle. It is one of several styles that come as part of the memoir class.

Figure B.25: The verville chapterstyle

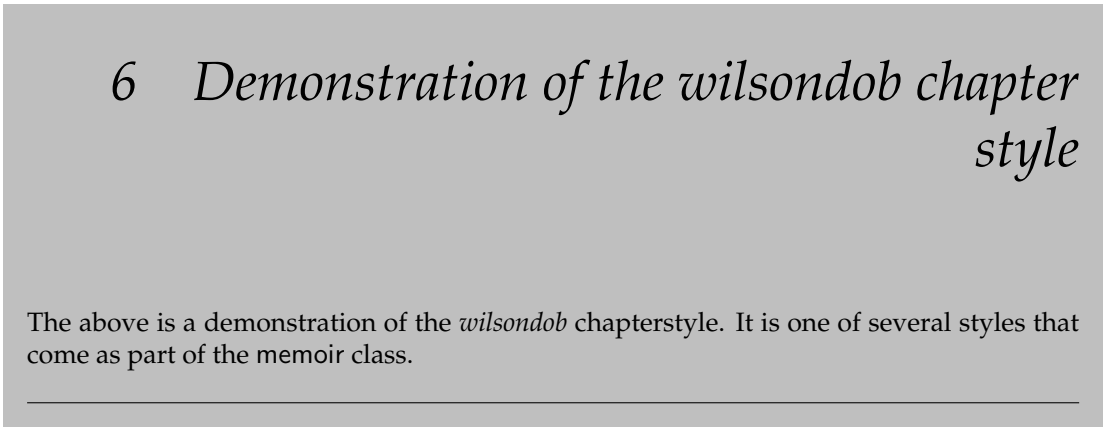


Figure B.26: The wilsondob chapterstyle

The code for some of these styles is given in below. For details of how the other chapter styles are defined, look at the documented class code. This should give you ideas if you want to define your own style.

Note that it is not necessary to define a new chapterstyle if you want to change the chapter headings — you can just change the individual macros without putting them into a style.

B.1.1 Chappell

A style that includes rules is one that I based on the chapter heads in [CB99] and which I have called *chappell* after the first author. The style, which is shown in Figure B.9, can easily form the basis for general heads in non-technical books.

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

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

B.1.2 Demo, Demo2 and demo3

I created a *demo* chapterstyle quite a time ago and used it on occasions in earlier editions of this Manual. Here is the original code.

```
\makechapterstyle{demo}{%
  \renewcommand*{\printchaptername}{\centering}
  \renewcommand*{\printchapternum}{\chapnumfont \numtoName{\c@chapter}}
  \renewcommand*{\chaptitlefont}{\normalfont\Huge\sffamily}
  \renewcommand*{\printchaptertitle}[1]{%
    \hrule\vskip\onelineskip \raggedleft \chaptitlefont ##1}
  \renewcommand*{\afterchaptertitle}%
                {\vskip\onelineskip \hrule\vskip \afterchapskip}
}% end demo
```

This has one serious failing and what I now believe is a poor design decision. The failing is that if you have any appendices that use the *demo* chapterstyle then they are numbered instead of being lettered. The poor design is that the position of the title with respect to the top of the page is not the same for numbered and unnumbered chapters. The *demo2* chapterstyle below fixes both of these at the expense of simplicity (at least for me).

```
\makechapterstyle{demo2}{%
  \renewcommand*{\printchaptername}{\centering}
  \renewcommand*{\printchapternum}{\chapnumfont
     \ifanappendix \thechapter \else \numtoName{\c@chapter}\fi}
  \renewcommand*{\chaptitlefont}{\normalfont\Huge\sffamily}
  \renewcommand*{\printchaptertitle}[1]{%
    \hrule\vskip\onelineskip \raggedleft \chaptitlefont ##1}
  \renewcommand*{\afterchaptertitle}{%
    \vskip\onelineskip \hrule\vskip \afterchapskip}
  \setlength{\beforechapskip}{3\baselineskip}
  \renewcommand*{\printchapternonum}{%
    \vphantom{\chapnumfont One}
    \afterchapternum%
    \vskip\topskip}
  \setlength{\beforechapskip}{2\onelineskip}
}% end{demo2}
```

You may find it instructive to compare the code for the *demo* and *demo2* chapterstyles.

Thec *demo* chapterstyle is still available in the class for backward compatibility reasons, but I strongly advise against anyone using it.

By chance I inadvertantly typest a chapterstyle that was a mixture of the *pedersen* and *demo2* styles. As a result there is now a *demo3* chapterstyle as well. The only difference between the two styles is in the definition of `\chapnumfont` which in *demo3* is:

```
\renewcommand*{\chapnumfont}{\normalfont\HUGE\itshape}
```

B.1.3 Pedersen

I have modified Troels Pedersen's original code to make it a little more efficient and flexible.

```
\newcommand*{\colorchapnum}{}
\newcommand*{\colorchaptitle}{}
\makechapterstyle{pedersen}{%
  \setlength{\beforechapskip}{-20pt}
  \setlength{\afterchapskip}{10pt}
  \renewcommand*{\chapnamefont}{\normalfont\LARGE\itshape}
  \renewcommand*{\chapnumfont}{\normalfont\HUGE\itshape\colorchapnum}
  \renewcommand*{\chaptitlefont}{\normalfont\huge\itshape\colorchaptitle}
  \renewcommand*{\afterchapternum}{}
  \renewcommand*{\printchaptername}{}
  \setlength{\midchapskip}{20mm}
  \renewcommand*{\chapternamenum}{}
  \renewcommand*{\printchapternum}{%
    \sidebar{\raisebox{0pt}[0pt][0pt]{\makebox[0pt][l]{%
      \resizebox{!}{\midchapskip}{\chapnumfont\thechapter}}}}}
  \renewcommand*{\printchaptertitle}[1]{\chaptitlefont ##1}
}
```

The chapter number is scaled up from its normal size and set in a sidebar.

`\colorchapnum` `\colorchaptitle`

The title is set with `colorchaptitle` and the number with `colorchapnum`, both of which default to doing nothing. Lars Madsen has suggested an attractive red color for these:

```
\usepackage{color}
\definecolor{ared}{rgb}{.647,.129,.149}
\renewcommand{\colorchapnum}{\color{ared}}
\renewcommand{\colorchaptitle}{\color{ared}}
\chapterstyle{pedersen}
```

The uncolored version is used for the chaptersyle for this chapter; because of setting the number in a sidebar it does not display well anywhere other than as a real chapter head.

B.1.4 Southall

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

```
\makechapterstyle{southall}{%
  \setlength{\afterchapskip}{5\baselineskip}
  \setlength{\beforechapskip}{36pt}
  \setlength{\midchapskip}{\textwidth}
  \addtolength{\midchapskip}{-\beforechapskip}
  \renewcommand*{\chapterheadstart}{\vspace*{2\baselineskip}}
```

1 Which I haven't seen

```
\renewcommand*{\chaptitlefont}{\huge\rmfamily\raggedright}
\renewcommand*{\chapnumfont}{\chaptitlefont}
\renewcommand*{\printchaptername}{}
\renewcommand*{\chapternamenum}{}
\renewcommand*{\afterchapternum}{}
\renewcommand*{\printchapternum}{%
  \begin{minipage}[t][\baselineskip][b]{\beforechapskip}
    {\vspace{0pt}\chapnumfont%%%\figureversion{lining}
                 \thechapter}
  \end{minipage}}
\renewcommand*{\printchaptertitle}[1]{%
  \hfill\begin{minipage}[t]{\midchapskip}
    {\vspace{0pt}\chaptitlefont ##1\par}\end{minipage}}
\renewcommand*{\afterchaptertitle}{%
  \par\vspace{\baselineskip}%
  \hrulefill \par\nobreak\noindent \vskip\afterchapskip}}
```

The resulting style is shown in Figure B.21.

B.1.5 Veelo

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

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

In the style as I modified it, `\beforechapskip` is used as the height of the number and `\midchapskip` is used as the length of the black bar.

```
\newlength{\numberheight}
\newlength{\barlength}
\makechapterstyle{veelo}{%
  \setlength{\afterchapskip}{40pt}
  \renewcommand*{\chapterheadstart}{\vspace*{40pt}}
  \renewcommand*{\afterchapternum}{\par\nobreak\vskip 25pt}
  \renewcommand*{\chapnamefont}{\normalfont\LARGE\flushright}
  \renewcommand*{\chapnumfont}{\normalfont\HUGE}
  \renewcommand*{\chaptitlefont}{\normalfont\HUGE\bfseries\flushright}
  \renewcommand*{\printchaptername}{%
    \chapnamefont\MakeUppercase{\@chapapp}}
  \renewcommand*{\chapternamenum}{}
  \setlength{\beforechapskip}{18mm}
  \setlength{\midchapskip}{\paperwidth}
  \addtolength{\midchapskip}{-\textwidth}
```

```
\addtolength{\midchapskip}{-\spinemargin}
\renewcommand*{\printchapternum}{%
  \makebox[0pt][l]{\hspace{.8em}%
    \resizebox{!}{\numberheight}{\chapnumfont \thechapter}%
    \hspace{.8em}%
    \rule{\midchapskip}{\beforechapskip}%
   }}%
 \makeoddfoot{plain}{}{}{\thepage}}
```

If you use this style you will also need to use the graphicx package [CR99] because of the `\resizebox` macro. The *veelo* style works best for chapters that start on recto pages.

C

Sniplets

This chapter is (over time) meant to hold various pieces of code for memoir that we have gathered over the years or others have contributed, and which we think might be useful for others. In some cases they will have been moved from the text to this place, in order to make the manual less cluttered.

If you have some memoir related code you would like to share, feel free to send it to `daleif@math.au.dk`.

SNIPLET OVERVIEW

Sniplet C.1 (Mirroring the output) 407
Sniplet C.2 (Remove pagenumber if only one page) 408
Sniplet C.3 (A kind of draft note) 408
Sniplet C.4 (Adding indentation to footnotes) 409
Sniplet C.5 (Background image and trimmarks) 409
Sniplet C.6 (Autoadjusted number widths in the ToC) 409
Sniplet C.7 (Using class tools to make a chapter ToC) 411
Sniplet C.8 (An appendix ToC) 413

SNIPLET C.1 (MIRRORING THE OUTPUT)

The memoir class is not quite compatible with the crop package. This is usually not a problem as we provide our own crop marks. But crop provide one feature that we do not: mirroring of the output. The following sniplet was posted on CTT by Heiko Oberdiek (2009/12/05, thread *Memoir and mirrored pdf output*)

```
\usepackage{atbegshi}
\usepackage{graphicx}
\AtBeginShipout{%
  \sbox\AtBeginShipoutBox{%
    \kern-1in\relax
    \reflectbox{%
      \rlap{\kern1in\copy\AtBeginShipoutBox}%
      \kern\stockwidth
    }%
  }%
}
```

SNIPLET C.2 (REMOVE PAGENUMBER IF ONLY ONE PAGE)

Memoir counts all the pages used. You can use this information in various ways. For example, say you are preparing a setup to write small assignments in, these may or may not be just one page. How do we remove the footer automatically if there is only one page?

Easy, place the following in the preamble (compile at least twice):

```
\AtEndDocument{\ifnum\value{lastsheet}=1\thispagestyle{empty}\fi}
```

SNIPLET C.3 (A KIND OF DRAFT NOTE)

Bastiaan Veelo has kindly provided example code for another form of a side note, as follows.

```
%% A new command that allows you to note down ideas or annotations in
%% the margin of the draft. If you are printing on a stock that is wider
%% than the final page width, we will go to some length to utilise the
%% paper that would otherwise be trimmed away, assuming you will not be
%% trimming the draft. These notes will not be printed when we are not
%% in draft mode.
\makeatletter
   \ifdraftdoc
     \newlength{\draftnotewidth}
     \newlength{\draftnotesignwidth}
     \newcommand{\draftnote}[1]{\@bsphack%
       {%% do not interfere with settings for other marginal notes
         \strictpagecheck%
         \checkoddpage%
         \setlength{\draftnotewidth}{\foremargin}%
         \addtolength{\draftnotewidth}{\trimedge}%
         \addtolength{\draftnotewidth}{-3\marginparsep}%
         \ifoddpage
           \setlength{\marginparwidth}{\draftnotewidth}%
           \marginpar{\flushleft\textbf{\textit{\HUGE !\ }}\small #1}%
         \else
           \settowidth{\draftnotesignwidth}{\textbf{\textit{\HUGE\ !}}}%
           \addtolength{\draftnotewidth}{-\draftnotesignwidth}%
           \marginpar{\raggedleft\makebox[0pt][r]{%% hack around
               \parbox[t]{\draftnotewidth}{%%%%%%%%% funny behaviour
                 \raggedleft\small\hspace{0pt}#1%
               }}\textbf{\textit{\HUGE\ !}}%
           }%
         \fi
       }\@esphack}
```

```
    \else
      \newcommand{\draftnote}[1]{\@bsphack\@esphack}
    \fi
\makeatother
```

Bastiaan also noted that it provided an example of using the `\foremargin` length. If you want to try it out, either put the code in your preamble, or put it into a package (i.e., `.sty` file) without the `\makeat...` commands.

SNIPLET C.4 (ADDING INDENTATION TO FOOTNOTES)

At times a document design calls for a footnote configuration equal to the default but everything indented more to the right. This can be achieved via

```
\newlength\myextrafootnoteindent
\setlength\myextrafootnoteindent{\parindent}
\renewcommand\makefootmarkhook{%
  \addtolength{\leftskip}{\myextrafootnoteindent}}
```

In this example we indent the footnotes to match the overall paragraph indentation. We need to save the current value of `\parindent` since it is reset in the footnotes.

SNIPLET C.5 (BACKGROUND IMAGE AND TRIMMARKS)

This sniplet comes from another problem described in CTT. If one use the eso-pic package to add a background image, this image ends up on top of the trim marks. To get it *under* the trim marks Rolf Niepraschk suggested the following trick

```
\RequirePackage{atbegshi}\AtBeginShipoutInit
\documentclass[...,showtrims]{memoir}
...
\usepackage{eso-pic}
...
```

SNIPLET C.6 (AUTOADJUSTED NUMBER WIDTHS IN THE TOC)

When the ToC is typeset the chapter, section etc. number is typeset within a box of a certain fixed with (one width for each sectional type). If this width is too small for the current document, the user have to manually adjust this width.

In this sniplet we present a method where we automatically record the widest.

It a later memoir version, we may add similar code to the core.

There are two ways to record the widest entries of the various types, either preprocess the entire ToC or measure it as a part of the ToC typesetting, store it in the `.aux` and reuse it on the next run. We will use the later approach. There is one caveat: The `.aux` file is read at `\begin{document}`, so we need to postpone our adjustments via `\AtBeginDocument`.

The following solution use some ToC related hooks within the class, plus the etoolbox and calc packages.

First we create some macros to store information within the .aux file, and retrieve it again.

```
\makeatletter
\newcommand\mem@auxrestore[2]{\csgdef{stored@value@#1}{#2}}
\newcommand\memstorevalue[2]{%
  \@bsphack%
  \immediate\write\@mainaux{\string\mem@auxrestore{#1}{#2}}%
  \@esphack}
\newcommand\RetrieveStoredLength[1]{%
  \ifcsdef{stored@value@#1}{\csuse{stored@value@#1}}{0pt}}%
\makeatletter
```

Here \RetrieveStoredLength can be used in most \setlength cases, at least when the calc package is loaded. The argument will be the name of the variable one asked to be stored. If no corresponding value has been found for a given name, 0 pt is returned.

Next we need to prepare the hooks. In this case we will show how to take care of \chapter, \section and \subsection. \chapter is relatively easy:[1]

```
\newlength\tmplen        % scratch length
\newlength\widestchapter % guess, they are zero by default
\renewcommand\chapternumberlinehook[1]{%
  \settowidth\tmplen{\hbox{\cftchapterfont#1}}%
  \ifdimgreater\tmplen\widestchapter{%
    \global\widestchapter=\tmplen}{}}
```

We use an alternative syntax to make the \widestchapter global.

Handling \section and \subsection is slightly more tricky, as they both use numberline. Instead we rely on the local value of the magic macro \cftwhatismyname.

```
\newlength\widestsection
\newlength\widestsubsection
\renewcommand\numberlinehook[1]{%
  % use a loop handler to loop over a list of possible
  % types. \forcsvlist comes from etoolbox
  \forcsvlist{\ToCHookListHandler{#1}}{section,subsection,subsubsection,%
    paragraph,subparagraph,figure,table}}
% the actual handler.
\newcommand\ToCHookListHandler[2]{%
  \edef\tmpstr{#2}%
  \ifdefstrequal{\cftwhatismyname}{\tmpstr}{%
    \settowidth\tmplen{\hbox{\csuse{cft\cftwhatismyname font}#1}}%
    \ifcslength{widest#2}{% is this length defined?
      \ifdimgreater\tmplen{\csuse{widest#2}}{%
        \global\csuse{widest#2}=\tmplen}{}}{}}{}}
```

1 In some cases you may want to use {\@chapapp@head\@cftbsnum #1\@cftasnum}

Even though the list mention more macros, we only use those we have added corresponding lengths for.

Next we need to store the values at the end of the document

```
\AtBeginDocument{\AtEndDocument{
    \memstorevalue{widestchapter}{\the\widestchapter}
    \memstorevalue{widestsection}{\the\widestsection}
    \memstorevalue{widestsubsection}{\the\widestsubsection}
  }}
```

Here is how to get the standard class setup for a three level TOC. We also add a little extra padding to the boxes. Remember that it may take a few compilations before the ToC settles down.

```
\newlength\cftnumpad          % padding
\setlength\cftnumpad{0.5em}
\AtBeginDocument{
  \cftsetindents{chapter}{0pt}{%
    \RetrieveStoredLength{widestchapter}+\cftnumpad}
  \cftsetindents{section}{%
    \cftchapterindent+\cftchapternumwidth}{%
    \RetrieveStoredLength{widestsection}+\cftnumpad}
  \cftsetindents{subsection}{%
    \cftsectionindent+\cftsectionnumwidth}{%
    \RetrieveStoredLength{widestsubsection}+\cftnumpad}
}
```

SNIPLET C.7 (USING CLASS TOOLS TO MAKE A CHAPTER TOC)

By using a few hooks, we will be able to create a simple chapter toc. First a few notes:

(a) In this class, the TOC data can be reused, thus we can load the TOC data as many times as we would like.

(b) Data in the TOC is stored as arguments the `\contentsline` macro, say (see also Figure 9.1 on page 147)

```
\contentsline{chapter}{\chapternumberline {1}Test}{3}
```

where the first argument determins which macro is used to process the data. Each of these macros look at the value of the `tocdepth` counter to know whether to typeset or not.

(c) Using some hooks we can insert local changes to `tocdepth` in order to only typeset the sections from the current chapter.

The idea is to be able to add hooks at key points in the ToC data, and then use these hooks to enable and disable typesetting.

We will need to add hooks just after a chapter line (like the one above), and we will need to be able to insert hooks just before items that mark the end of a chapter, that is the next `\chapter`, `\part`, `\book`, plus a macro like `\appendixpage` which also write to the ToC.

First we define hooks that add hooks into the TOC. We use a counter to make each start and end hook unique. We add *end markers* above the ToC entries for `\chapter`, `\part` and `\book`.

```
\newcounter{tocmarker}
\renewcommand\mempreaddchaptertotochook{\cftinserthook{toc}{end-\thetocmarker}}
\renewcommand\mempreaddparttotochook   {\cftinserthook{toc}{end-\thetocmarker}}
\renewcommand\mempreaddbooktotochook   {\cftinserthook{toc}{end-\thetocmarker}}
\renewcommand\mempreaddapppagetotochook{\cftinserthook{toc}{end-\thetocmarker}}
% start marker
\renewcommand\mempostaddchaptertotochook{%
  \stepcounter{tocmarker}\cftinserthook{toc}{start-\thetocmarker}}
\let\normalchangetocdepth\changetocdepth % for later
```

The hooks inserted into the TOC file, does nothing by default. You will notice that the line above will now look like:

```
\cftinsert {end-0}
\contentsline{chapter}{\chapternumberline {1}Test}{3}
\cftinsert {start-1}
...
\cftinsert {end-1}
\contentsline{chapter}{\chapternumberline {2}Test}{5}
```

Thus to get a chapter toc command we need to make sure that (1) all entries are disabled, (2) at `start-1` we reenable TOC entries, and (3) at `end-1` disable TOC entries again. Here is the rest of the code, explained via comments.

```
\makeatletter
\newcommand\chaptertoc{
  % make changes local, remember counters a global
  \begingroup
  % store current value, to be restored later
  \setcounter{@memmarkcntra}{\value{tocdepth}}
  % when ever \settocdepth is used, it adds the new value to the
  % ToC data. This cause problems when we want to disable all
  % entries. Luckily the data is added via a special macro, we we
  % redefine it, remember we stored the original value earlier.
  \let\changetocdepth\@gobble
  % disable all entries (using our copy from above)
  \normalchangetocdepth{-10}
  % enable toc data within our block, we go as far as subsubsection
```

```
  \cftinsertcode{start-\thetocmarker}{\normalchangetocdepth{3}}
  % when the block is done, disable the remaining
  \cftinsertcode{end-\thetocmarker}{\normalchangetocdepth{-10}}
  % remove the spacing above the toc title
  \let\tocheadstart\relax
  % remove the toc title itself
  \let\printtoctitle\@gobble
  % remove space below title
  \let\aftertoctitle\relax
  % reformat TOC entries:
  \setlength{\cftsectionindent}{0pt}
  \setlength{\cftsubsectionindent}{\cftsectionnumwidth}
  \setlength{\cftsubsubsectionindent}{\cftsubsectionindent}
  \addtolength{\cftsubsubsectionindent}{\cftsubsectionnumwidth}
  \renewcommand\cftsectionfont{\small}
  \renewcommand\cftsectionpagefont{\small}
  \renewcommand\cftsubsectionfont{\small}
  \renewcommand\cftsubsectionpagefont{\small}
  \renewcommand\cftsubsubsectionfont{\small}
  \renewcommand\cftsubsubsectionpagefont{\small}
  % include the actual ToC data
  \tableofcontents*
  \endgroup
  % restore tocdepth
  \setcounter{tocdepth}{\value{@memmarkcntra}}
  % to indent or not after the chapter toc
  \m@mindentafterchapter
  % space between chapter toc and text
  \par\bigskip
  % handles indentation after the macro
  \@afterheading}
\makeatother
```

Note that if the `\chapterprecistoc` or `\chapterprecis` has been used then that data is also added to the ToC data, and we will need to locally disable it in the chapter ToC. This can be done by adding

```
\let\precistoctext\@gobble
```

to the `\chaptertoc` definition above, just make sure it is added before calling before `\tableofcontents*`.

SNIPLET C.8 (AN APPENDIX TOC)

Here we assume a structure like

```
\tableofcontents*
\chapter
```

```
\chapter
\chapter
\appendix
\appendixpage
\appendixtableofcontents
\chapter
\chapter
\chapter
```

where the first ToC should just show until (and including) `\appendixpage`, and `\appendixtableofcontents` should only list the appendices.

We also assume that no `\settocdepth`'s have been issued after `\appendixpage`.

We only need a single hook after `\appendixpage`.

```
\renewcommand\mempostaddapppagetotochook{\cftinserthook{toc}{BREAK}}
\cftinsertcode{BREAK}{\changetocdepth{-10}}
\let\normalchangetocdepth\changetocdepth     % needed for later
```

Then the definition of the actual appendix ToC:

```
\makeatletter
\newcommand\appendixtableofcontents{
  \begingroup
  \let\changetocdepth\@gobble
  \normalchangetocdepth{-10}
  \cftinsertcode{BREAK}{\normalchangetocdepth{3}}
  \renewcommand\contentsname{Appendices overview}
  \tableofcontents*
  \endgroup
}
\makeatother
```

D

Pictures

There are many freely available LaTeX introductions on CTAN and other places. One thing that these apparently are not covering is the traditional picture environment. It can be very handy in many applications, though for more complex drawings the reader might be better of with TiKz/pgf or PSTricks. For the benefit of the general reader we here provide a lesson in the standard picture environment.

Writers comment: There are many extensions to the stock picture environment provided by the LaTeX kernel. We have chosen not to deal with them in this chapter but instead concentrate on what you get as is from the kernel. But there are a few handy packages that the reader might want to explore: picture (by Heiko Oberdiek) which extends the `\put` syntax to include arbitrary lengths, like 50mm; pict2e which is mentioned in [GM+07] but just recently was released; eepic. All packages are available from CTAN.

This chapter describes how to draw diagrams and pictures using LaTeX. Pictures are drawn in the `picture` environment. You can draw straight lines, arrows and circles; you can also put text into your pictures.

Most often pictures are drawn within a `figure` environment, but they may also be drawn as part of the normal text.

D.1 Basic principles

The positions of picture elements are specified in terms of a two-dimensional cartesian coordinate system. A *coordinate* is a number, such as `7`, `-21` or `1.78`. In the cartesian coordinate system, a pair of coordinates (i.e., a pair of numbers) specifies a position relative to the position designated as `(0,0)`. This special position is called the *origin*. The first of the coordinate pair gives the value of the horizontal distance from the origin to the position. A positive coordinate is an offset to the right and a negative number is an offset to the left. The first value of a coordinate pair is called the *x coordinate*. The second value of a coordinate pair is called the *y coordinate* and gives the vertical offset from the origin (positive upwards and negative downwards).

```
\unitlength
```

To draw a picture we also need to specify the units of measurement. By default, LaTeX takes the printer's point (there are 72.27 points to an inch) as the measurement of length. The value of the unit of length measurement within a `picture` environment is actually given by the value of the `\unitlength` length declaration. This can be changed to any length that you like via the `\setlength` command. For example,

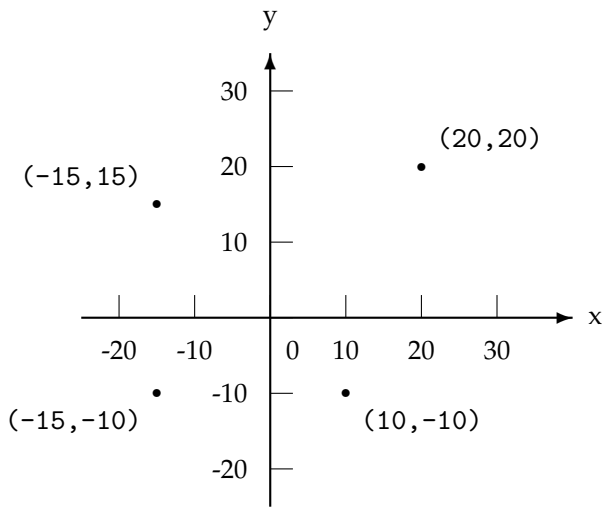


Figure D.1: Some points in the cartesian coordinate system

`\setlength{\unitlength}{2mm}`
will make the value of `\unitlength` to be two millimeters.

Figure D.1 shows the positions of some points and their coordinate values. Coordinate pairs are typed as a pair of numbers, separated by a comma, and enclosed in parentheses.

```
\thinlines
\thicklines
\linethickness{⟨len⟩}
```

In general, LaTeX can draw lines of only two thicknesses, thin and thick. The required thickness is specified via either a `\thicklines` or a `\thinlines` declaration, with the latter being the default.

There is another declaration, `\linethickness`, which can be used to change the thickness of horizontal and vertical lines only. It sets the thickness of these lines to ⟨*len*⟩, but has no effect on any sloping lines.

A `picture` environment has a required size pair argument that specifies the width and height of the picture, in terms of the value of `\unitlength`.

```
\begin{picture}(⟨width, height⟩) ⟨contents⟩ \end{picture}
\begin{picture}(⟨width, height⟩)(⟨llx, lly⟩) ⟨contents⟩ \end{picture}
```

The environment creates a box of size ⟨*width*⟩ by ⟨*height*⟩, which will not be split across pages. The default position of the origin in this environment is at the lower left hand corner of the box. For example,

```
\begin{picture}(80,160)
```

creates a picture box of width 80 and height 160 whose lower left hand corner is at `(0,0)`. There is also an optional coordinate pair argument (which comes after the required argu-

ment) that specifies the coordinates of the lower left hand corner of the box if you do not want the default origin.

```
\begin{picture}(80,160)(10,20)
```

specifies a picture box of width 80 and height 160, as before, but with the bottom left hand corner having coordinates of `(10,20)`. Thus, the top right hand corner will have coordinates `(90,180)`. Note that the optional argument is enclosed in parentheses not square brackets as is ordinarily the case. Typically, the optional argument is used when you want to shift everything in the picture. LaTeX uses the required argument to determine the space required for typesetting the result.

You are not limited to drawing within the box, but if you do draw outside the box the result might run into other text on the page, or even run off the page altogether. LaTeX only reserves the space you specify for the picture and does not know if anything protrudes. In particular

```
\begin{picture}(0,0)
```

creates a zero-sized picture which takes no space at all. This can be very useful if you need to position things on the page.

Within the `picture` environment, LaTeX is in a special *picture* mode (which is a restriced form of LR mode). The only commands that can appear in picture mode are `\put`, `\multiput` and `\qbezier` commands, and a few declarations such as the type style and the thickness declarations. By the way, you should only change the value of `\unitlength` outside picture mode otherwise LaTeX will get confused about its measurements.

D.2 Picture objects

In a picture everything is placed and drawn by the `\put` (or its `\multiput` variant) command.

`\put(`⟨*x*, *y*⟩`){`⟨*object*⟩`}`

`\put` places ⟨*object*⟩ in the picture with the object's *reference point* at position `(`⟨*x*, *y*⟩`)`.

The following sections describe the various picture objects.

D.2.1 Text

Text is the simplest kind of picture object. This is typeset in LR mode and the reference point is at the lower left hand corner of the text.

Source for example D.1

```
\setlength{\unitlength}{1mm}  % measurements in millimeters
\begin{picture}(30,10)        % define size of picture
\put(0,0){\framebox(30,10){}} % draw frame around picture
\put(10,5){Some text}         % place text
\thicklines
\put(10,5){\vector(1,1){0}}   % mark reference point
\end{picture}
\setlength{\unitlength}{1pt}  % reset measurements to default
```

Typeset example D.1: Picture: text

In the diagram, and those following, the reference point is indicated by an arrow. Also, a box is drawn round the diagram at the same size as the `picture` environment.

D.2.2 Boxes

A box picture object is made with one of the box commands. When used in picture mode, the box commands have a slightly different form than when in normal text. The first argument of a box command is a size pair that specifies the width and height of the box. The last argument is the text to be placed in the box. The reference point of a box is the lower left hand corner.

`\framebox(`⟨*width, height*⟩`)[`⟨*pos*⟩`]{`⟨*text*⟩`}`
`\makebox(`⟨*width, height*⟩`)[`⟨*pos*⟩`]{`⟨*text*⟩`}`

The `\framebox` command draws a framed box of the specified (⟨*width, height*⟩) dimensions around the text.

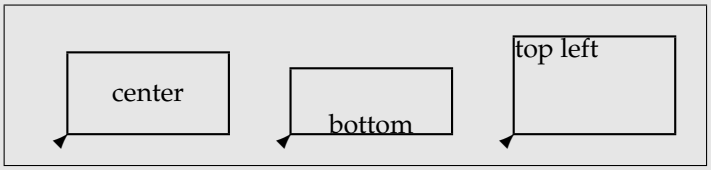
Source for example D.2

```
\setlength{\unitlength}{1pc}
\begin{picture}(22,5)
\put(0,0){\framebox(22,5){}}             % empty box
\thicklines
\put(2,1){\framebox(5,2.5){center}}      % centered text
\put(2,1){\vector(1,1){0}}               % ref point
\put(9,1){\framebox(5,2)[b]{bottom}}     % bottomed text
\put(9,1){\vector(1,1){0}}               % ref point
\put(16,1){\framebox(5,3)[tl]{top left}} % cornered text
\put(16,1){\vector(1,1){0}}              % ref point
\end{picture}
\setlength{\unitlength}{1pt}
```

The default position of the *text* is centered in the box. However, this can be changed via an optional argument (which is enclosed in square brackets), placed between the coordinate pair and the text argument. This argument consists of either one or two of the following letters.

l (left) Places the contents at the left of the box.

Typeset example D.2: Picture: text in boxes



`r` (right) Places the contents at the right of the box.

`t` (top) Places the contents at the top of the box.

`b` (bottom) Places the contents at the bottom of the box.

These place the text in the corresponding position in the box. In a two-letter argument the order of the letters is immaterial. For example, `[tr]` and `[rt]` will both result in the text being placed at the top right hand corner of the box. Unlike the normal `\framebox` command, a `\framebox` in a `picture` environment does not add any extra space around the text.

Corresponding to the `\framebox` there is a `\makebox` command which does not draw a frame around its contents. The `\makebox` command takes the same arguments as the `\framebox`. Particularly interesting is when you specify a zero sized `\makebox`. A `\makebox(0,0){text}` command will make the reference point the center of `text`. Similarly, the other positioning arguments which will adjust the reference point with respect to the box contents. This can be used for fine-tuning the position of text in a picture.

Source for example D.3

```
\setlength{\unitlength}{1pc}
\begin{picture}(16,2)
\put(0,0){\framebox(16,2){}}
\thicklines
\put(3.5,1){\makebox(0,0){center}}      % ref at text center
\put(3.5,1){\vector(0,-1){0}}
\put(7,1){\makebox(0,0)[b]{bottom}}     % ref at text bottom
\put(7,1){\vector(0,1){0}}
\put(11,1){\makebox(0,0)[tl]{top left}} % ref at text top left
\put(11,1){\vector(1,-1){0}}
\end{picture}
\setlength{\unitlength}{1pt}
```

You can draw a dashed box with the `\dashbox` command.

Typeset example D.3: Picture: positioning text

Typeset example D.4: Picture: dashed box

`\dashbox{`⟨*len*⟩`}(`⟨*width, height*⟩`)[`⟨*pos*⟩`]{`⟨*text*⟩`}`

The first argument of this command specifies the length of each dash. The following arguments are the same as for the other box commands.

Source for example D.4

```
\setlength{\unitlength}{4mm}
\begin{picture}(7,4)
\put(0,0){\framebox(7,4){}}
\thicklines
\put(1,1){\dashbox{0.5}(5,2)[tr]{top right}}
\put(1,1){\vector(1,1){0}}
\end{picture}
\setlength{\unitlength}{1pt}
```

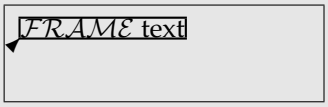
The appearance of the box is best when the width and height of the box are integer multiples of the dash length. In the example the dash length has been set to `0.5` with the width and height set as `(5,2)`; thus the width and height are respectively ten and four times the dash length.

The `\frame` command draws a frame around the contents of the box that exactly fits the contents.

`\frame{`⟨*contents*⟩`}`

It takes a single required argument which is the contents.

Typeset example D.5: Picture: framing



Source for example D.5

```
\setlength{\unitlength}{1pc}
\begin{picture}(10,3)
\put(0,0){\framebox(10,3){}}
\thicklines
\put(0.5,2){\frame{$\mathcal{FRAME}$ text}}
\put(0.5,2){\vector(1,1){0}}
\end{picture}
\setlength{\unitlength}{1pt}
```

The `\shortstack` command enables you to stack text vertically. It produces a box with a single column of text. As with the other boxes, the reference point is at the lower left hand corner, although no frame is drawn around the stack. The `\shortstack` command is an ordinary box making command, but it is not often used outside picture mode.

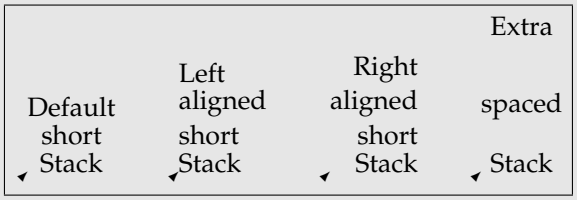
`\shortstack[`⟨*pos*⟩`]{`⟨*text*⟩`}`

Each line of ⟨*text*⟩, except for the last, is ended by a `\\` command. The default is to center each text line within the column. However, there is an optional positioning argument. A value of `l` for ⟨*pos*⟩ will left align the text and a value of `r` will right align the text.

Source for example D.6

```
\setlength{\unitlength}{1mm}
\begin{picture}(75,25)
\put(0,0){\framebox(75,25){}}
\put(3,3){\shortstack{Default \\ short \\ Stack}}
\put(3,3){\vector(1,1){0}}
\put(23,3){\shortstack[l]{Left\\aligned\\short\\Stack}}
\put(23,3){\vector(1,1){0}}
\put(43,3){\shortstack[r]{Right\\aligned\\short\\Stack}}
\put(43,3){\vector(1,1){0}}
```

Typeset example D.6: Picture: stacking




```
\put(63,3){\shortstack{Extra \\[4ex] spaced \\[2ex] Stack}}
\put(63,3){\vector(1,1){0}}
\end{picture}
\setlength{\unitlength}{1pt}
```

The rows in a stack are not evenly spaced. The spacing between two rows can be changed in one of two ways.

1. Add a strut to a row. A strut is a vertical rule with no width.
2. Use the optional argument to the \\ command. This optional argument is a length value.

   \\[⟨*len*⟩]

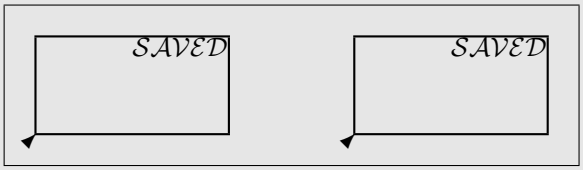
   It has the effect of adding additional ⟨*len*⟩ vertical space between the two lines that the \\ separates.

\newsavebox{⟨*box*⟩}
\savebox{⟨*box*⟩}(⟨*width, height*⟩)[⟨*pos*⟩]{⟨*text*⟩}
\sbox{⟨*box*⟩}{⟨*text*⟩}
\usebox{⟨*box*⟩}

Just as in normal text you can save and reuse boxes. The \savebox macro in picture mode is a variant of the normal text version, but the other three commands are the same in both picture and paragraph modes, and are described in Chapter 15. In picture mode you have to specify the size of the storage box when saving it, via the (⟨*width, height*⟩) argument to \savebox.

A \savebox command can be used within a picture to store a picture object. The first argument of \savebox is the name of the storage bin to be used. The following arguments are the same as the \makebox command. The result is stored, not drawn. When you have saved something it can be drawn in either the same or other pictures via the \usebox command. This command takes one argument, which is the name of the storage bin.

Typeset example D.7: Picture: saved boxes



Source for example D.7

```
\setlength{\unitlength}{1pc}
\begin{picture}(18,5)
\put(0,0){\framebox(18,5){}}
\newsavebox{\Mybox}
\savebox{\Mybox}(6,3)[tr]{$\mathcal{SAVED}$}
\thicklines
\put(1,1){\frame{\usebox{\Mybox}}}
\put(11,1){\frame{\usebox{\Mybox}}}
\put(1,1){\vector(1,1){0}}
\put(11,1){\vector(1,1){0}}
\end{picture}
\setlength{\unitlength}{1pt}
```

It can take LaTeX a long time to draw something. When a box is saved it actually contains the typeset contents, which then just get printed out when the box is used. It can save processing time if something which appears several times is saved and then used as and where required. On the other hand, a saved box can use up a significant amount of LaTeX's internal storage space. The `\sbox` command with an empty text argument can be used to delete the contents of a bin. For example,

```
\sbox{\Mybox}{}
```

will empty the `\Mybox` box. Note that this does not delete the storage box itself.

D.2.3 Lines

LaTeX can draw straight lines, but the range of slopes for lines is somewhat restricted. Further, very short lines cannot be drawn.

`\line(`⟨*i*, *j*⟩`){`⟨*distance*⟩`}`

The pair `(`⟨*i*, *j*⟩`)` specifies the *slope* of the line, and ⟨*distance*⟩ is a value that controls the length of the line. The line starts at its reference point (i.e., the place where it is `\put`). The

slope of the line is such that if a point on the line is slid along the line, then for every i units the point moves in the horizontal direction it will also have moved j units in the vertical direction. Negative values for i or j have the expected meaning. A move of -3 units in i means a move of 3 units to the left, and similarly a move of -4 units in j means a move of 4 units downwards. So, a line sloping up to the right will have positive values for i and j, while a line sloping up to the left will have a negative value for i and a positive value for j.

The ⟨*distance*⟩ argument specifies the length of the line in the x (horizontal) direction. One problem with this may have occured to you: what if the line is vertical (i.e., $i = 0$)? In this case only, ⟨*distance*⟩ specifies the vertical length of the line. The ⟨*distance*⟩ argument must be a non-negative value. For horizontal and vertical lines only, the actual length of the line is ⟨*distance*⟩. Figure D.2, which is produced from the code below, diagrams the line specification arguments.

```
\begin{figure}
\centering
\setlength{\unitlength}{1mm}
\begin{picture}(70,60)
\thicklines   % draw line and ref point
  \put(10,20){\line(2,1){40}}
  \put(10,20){\vector(1,-1){0}}
\thinlines    % draw axes
  \put(0,0){\vector(1,0){60}} \put(63,0){x}
  \put(0,0){\vector(0,1){50}} \put(0,53){y}
              % draw i and j vectors
  \put(20,25){\vector(1,0){20}}
  \put(30,22){\makebox(0,0)[t]{$i$}}
  \put(40,25){\vector(0,1){10}}
  \put(42,30){\makebox(0,0)[l]{$j$}}
              % draw distance vector
  \put(30,10){\vector(-1,0){20}}
  \put(30,10){\vector(1,0){20}}
  \put(30,8){\makebox(0,0)[t]{\textit{distance}}}
\end{picture}
\setlength{\unitlength}{1pt}
\caption{Specification of a line or arrow}
\label{flpic:spec}
\end{figure}
```

Only a fixed number of slopes are available. This is because LaTeX uses a special font for drawing lines — a line actually consists of little bits of angled rules joined together. Thus, there is only a limited number of values for i and j. They must both be integers and in the range $-6 \leq i, j \leq 6$. Also, they must have no common divisor other than 1. In other words, the ratio between i and j must be in its simplest form. You cannot, for example, have $(3, 6)$; instead it would have to be $(1, 2)$. The shortest line that LaTeX can draw is about ten points (1/7 inch approximately) in overall length. You can, though, draw lines that are too long to fit on the page.

Figure D.3 shows the lines and arrows slanting upwards and to the right that can be

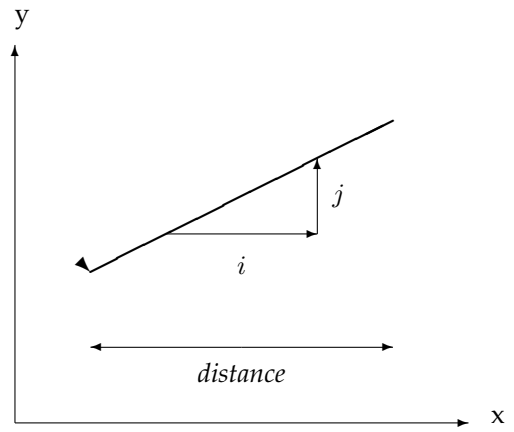


Figure D.2: Specification of a line or arrow

drawn in LaTeX. The slope (i, j) pair are shown to the right of the first set of lines and arrows, together with the j/i ratio which gives the slope of the line as a decimal number.

D.2.4 Arrows

As shown in Figure D.3 you can also draw a line with an arrowhead on it. These are specified by the `\vector` command.

`\vector(`⟨*i, j*⟩`){`⟨*distance*⟩`}`

This works exactly like the `\line` command and the arrowhead is put on the line at the end away from the reference point. That is, the arrow points away from the reference point. If the ⟨*distance*⟩ argument is too small (zero, for instance) the arrowhead only is drawn, with its point at the position where it is `\put`.

LaTeX is even more restrictive in the number of slopes that it can draw with arrows than it is with lines. The (i, j) slope specification pair must lie in the range $-4 \leq i, j \leq 4$. Also, as with the `\line` command, they must have no common divisor.

D.2.5 Circles

LaTeX can draw two kinds of circles. One is an open circle where only the perimeter is drawn, and the other is a solidly filled disk.

`\circle{`⟨*diameter*⟩`}`
`\circle*{`⟨*diameter*⟩`}`

The reference point for the open circle, drawn by the `\circle` command, and the disk, which is drawn by the `\circle*` command, is at the center of the circle. The argument to the commands is the ⟨*diameter*⟩ of the circle.

Source for example D.8

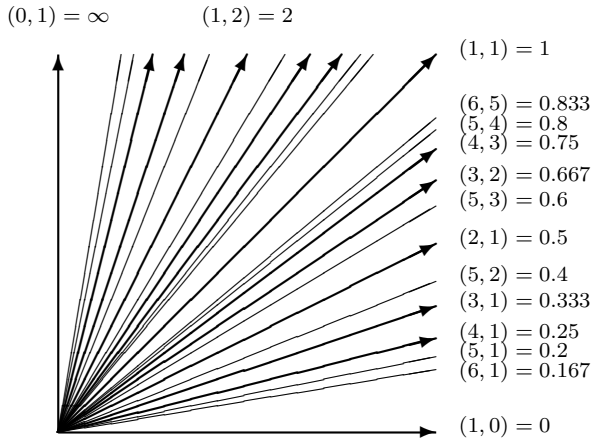
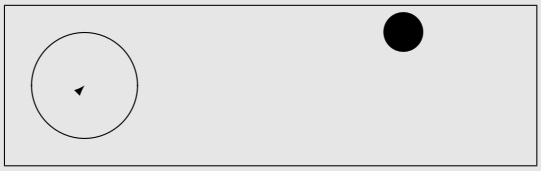


Figure D.3: Sloping lines and arrows

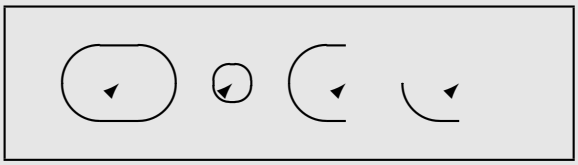
Typeset example D.8: Picture: circles



```
\setlength{\unitlength}{1pt}
\begin{picture}(200,60)
\put(0,0){\framebox(200,60){}}
\put(30,30){\circle{40}}
\put(30,30){\vector(1,1){0}}
\put(150,50){\circle*{20}}
\end{picture}
\setlength{\unitlength}{1pt}
```

Just as with the `\line` and `\vector` commands, there is only a limited range of circles that can be drawn. Typically, the maximum diamter of a `\circle` is about 40 points, while for a `\circle*` the maximum diameter is less, being about 15 points. LaTeX will choose

Typeset example D.9: Picture: ovals



the nearest sized circle to the one that you specify. Either consult your local guru to find what sized circles you can draw on your system, or try some experiments by drawing a range of circles to see what happens.

Quarter circles and boxes

In LaTeX an `\oval` is a rectangular box with rounded corners.

`\oval(`⟨*width, height*⟩`)[`⟨*portion*⟩`]`

The `\oval` command has one required argument which specifies the width and height of the box. The normally sharp corners of the box are replaced by quarter circles of the maximum possible radius (which LaTeX figures out for itself). Unlike the boxes discussed earlier, the reference point is at the 'center' of the oval.

Source for example D.9

```
\setlength{\unitlength}{1mm}
\begin{picture}(75,20)
\thicklines
\put(0,0){\framebox(75,20){}}
\put(15,10){\oval(15,10)}     % complete oval
\put(15,10){\vector(1,1){0}}
\put(30,10){\oval(5,5)}       % small oval
\put(30,10){\vector(1,1){0}}
\put(45,10){\oval(15,10)[l]}  % left half
\put(45,10){\vector(1,1){0}}
\put(60,10){\oval(15,10)[bl]} % bottom left quarter
\put(60,10){\vector(1,1){0}}
\end{picture}
\setlength{\unitlength}{1pt}
```

The `\oval` command also has one optional argument, ⟨*portion*⟩, which comes after the required argument. Use of the optional argument enables either half or a quarter of the

Typeset example D.10: Picture: text in oval

complete rounded box to be drawn. The argument is a one or two letter code drawn from the following.

- l (left) Draw the left of the oval.
- r (right) Draw the right of the oval.
- t (top) Draw the top of the oval.
- b (bottom) Draw the bottom of the oval.

These are similar to the optional positioning argument in the box commands. A one letter code will draw the designated half of the oval, while a two letter code results in the designated quarter of the oval being drawn. In all cases the reference point is at the center of the 'complete' oval.

Source for example D.10

```
\setlength{\unitlength}{1mm}
\begin{picture}(30,10)
\thicklines
\put(15,5){\oval(30,10)}
\put(15,5){\makebox(0,0){Text in oval}}
\end{picture}
\setlength{\unitlength}{1pt}
```

Unlike the boxes described in §D.2.2 there is no ⟨*text*⟩ argument for an `\oval`. If you want the rounded box to contain text, then you have to place the text inside the box yourself. The code in example D.10 shows one way of doing this; a zero-sized box is used to center the text at the center of the oval.

D.3 REPETITIONS

The `\multiput` command is a convenient way to place regularly spaced copies of an object in a picture.

`\multiput(`⟨*x*, *y*⟩`)(`⟨*dx*, *dy*⟩`){`⟨*num*⟩`}{`⟨*object*⟩`}`

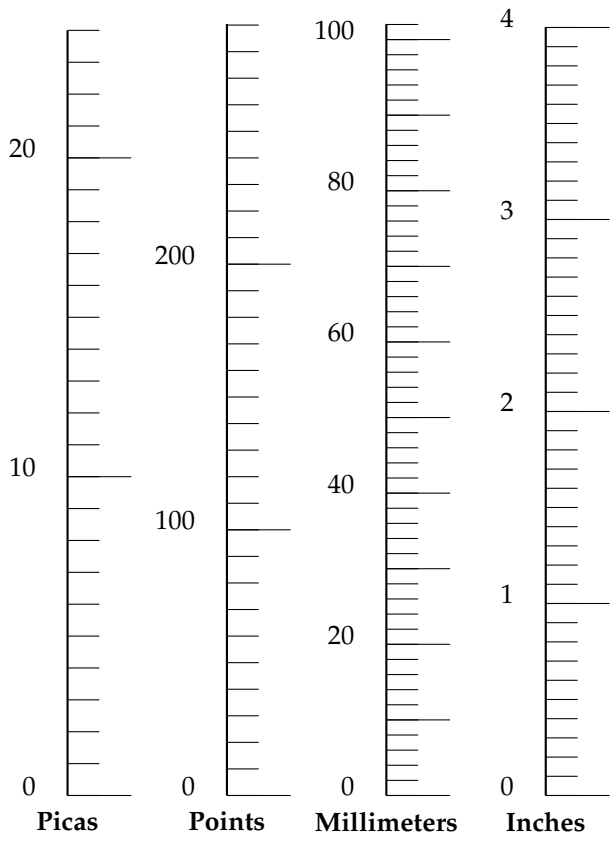


Figure D.4: Some measuring scales

As you can see, this is similar to the syntax for the `\put` command, except that there are two more required arguments, namely `(`⟨*dx*, *dy*⟩`)` and *num*.

The `(`⟨*dx*, *dy*⟩`)` argument is a pair of (decimal) numbers that specify the amount that the ⟨*object*⟩ shall be moved at each repetition. The first of this pair specifies the horizontal movement and the second the vertical movement. Positive values shift to the right or up, and negative numbers shift to the left or down. The ⟨*num*⟩ argument specifies how many times the ⟨*object*⟩ is to be drawn.

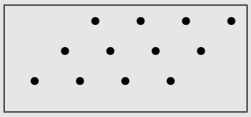
The code below produces Figure D.4. This example also shows that a `picture` can be placed inside another `picture`. Often it is useful to break a complex diagram up into pieces, with each piece being a separate `picture`. The pieces can then be individually positioned within the overall diagram.

```
\begin{figure}
\setlength{\unitlength}{1pc}
\centering
\begin{picture}(21,26)
%  Draw Pica scale
```

```
\put(2,2){\begin{picture}(5,24)
  \put(0,-0.5){\makebox(0,0)[t]{\textbf{Picas}}}
  \thicklines \put(0,0){\line(0,1){24.0}}
  \thinlines \multiput(0,0)(0,1){25}{\line(1,0){1}}
             \multiput(0,0)(0,10){3}{\line(1,0){2}}
  \put(-1,0){\makebox(0,0)[br]{0}}
  \put(-1,10){\makebox(0,0)[br]{10}}
  \put(-1,20){\makebox(0,0)[br]{20}}
  \end{picture}}
%  Draw Points scale
\put(7,2){\begin{picture}(5,24)
  \put(0,-0.5){\makebox(0,0)[t]{\textbf{Points}}}
  \thicklines \put(0,0){\line(0,1){24.2}}
  \thinlines \multiput(0,0)(0,0.8333){30}{\line(1,0){1}}
             \multiput(0,0)(0,8.333){3}{\line(1,0){2}}
  \put(-1,0){\makebox(0,0)[br]{0}}
  \put(-1,8.333){\makebox(0,0)[br]{100}}
  \put(-1,16.667){\makebox(0,0)[br]{200}}
  \end{picture}}
%  Draw Millimeter scale
\put(12,2){\begin{picture}(5,24)
  \put(0,-0.5){\makebox(0,0)[t]{\textbf{Millimeters}}}
  \thicklines \put(0,0){\line(0,1){24.2}}
  \thinlines \multiput(0,0)(0,0.4742){15}{\line(1,0){1}}
             \multiput(0,0)(0,2.3711){11}{\line(1,0){2}}
  \put(-1,0){\makebox(0,0)[br]{0}}
  \put(-1,4.742){\makebox(0,0)[br]{20}}
  \put(-1,9.484){\makebox(0,0)[br]{40}}
  \put(-1,14,226){\makebox(0,0)[br]{60}}
  \put(-1,18.968){\makebox(0,0)[br]{80}}
  \put(-1,23.71){\makebox(0,0)[br]{100}}
  \end{picture}}
%  Draw Inch scale
\put(17,2){\begin{picture}(5,24)
  \put(0,-0.5){\makebox(0,0)[t]{\textbf{Inches}}}
  \thicklines \put(0,0){\line(0,1){24.1}}
  \thinlines \multiput(0,0)(0,0.60225){41}{\line(1,0){1}}
             \multiput(0,0)(0,6.0225){5}{\line(1,0){2}}
  \put(-1,0){\makebox(0,0)[br]{0}}
  \put(-1,6.0225){\makebox(0,0)[br]{1}}
  \put(-1,12.045){\makebox(0,0)[br]{2}}
  \put(-1,18.0675){\makebox(0,0)[br]{3}}
  \put(-1,24.09){\makebox(0,0)[br]{4}}
  \end{picture}}

\end{picture}
\setlength{\unitlength}{1pt}
```

Typeset example D.11: Picture: repetitions



```
\caption{Some measuring scales} \label{flpic:scales}
\end{figure}
```

You can also make regular two-dimensional patterns by using a `\multiput` pattern inside another `\multiput`. As LaTeX will process each `\multiput` every time it is repeated it is often more convenient to store the results of the first `\multiput` in a bin and then use this as the argument to the second `\multiput`.

Source for example D.11

```
\setlength{\unitlength}{1mm}
\begin{picture}(32,14)
\put(0,0){\framebox(32,14){}}
\savebox{\Mybox}(8,8){\multiput(0,0)(4,4){3}{\circle*{1}}}
\multiput(4,4)(6,0){4}{\usebox{\Mybox}}
\sbox{\Mybox}{}
\end{picture}
\setlength{\unitlength}{1pt}
```

Remember that a storage bin must have been declared via a `\newsavebox` command before it can be used. I originally declared and used the `\Mybox` bin in §D.2.2. As the above example shows, you can change the contents of a storage bin by utilising it in another `\savebox`. Storage bins can use up a lot of LaTeX's memory. After you have finished with a storage bin empty it via the `\sbox` command with an empty last argument, as shown in the example.

D.4 Bezier curves

Standard LaTeX provides one further drawing command — the `\qbezier` command. This can be used for drawing fairly arbitrary curves.

`\qbezier[`⟨*num*⟩`](`⟨*Xs, Ys*⟩`)(`⟨*Xm, Ym*⟩`)(`⟨*Xe, Ye*⟩`)`

The command will draw what geometers call a *quadratic Bezier curve* from the point (⟨*Xs, Ys*⟩) to the point (⟨*Xe, Ye*⟩). The curve will pass somewhere near to the point (⟨*Xm, Ym*⟩).

Bezier curves are named after Pierre Bezier who first used them in 1962. They are widely used in Computer Aided Design (CAD) programs and other graphics and font

design systems. Descriptions, with varying degrees of mathematical complexity, can be found in many places: when I was a practicing geometer these included [FP80], [Mor85] and [Far90]; no doubt there are more recent sources available and there is a brief review in [Wil04a].

Figure D.5 shows two of these curves. The figure was produced by the code below.

```
\begin{figure}
\setlength{\unitlength}{1mm}
\centering
\begin{picture}(100,100)

\thicklines % first curve
\qbezier(10,50)(50,90)(50,50)
\thinlines % draw lines joining control points
\put(10,50){\line(1,1){40}}
\put(50,90){\line(0,-1){40}}
% label control points
\put(10,45){\makebox(0,0)[t]{\texttt{(10,50)}}}
\put(50,95){\makebox(0,0)[b]{\texttt{(50,90)}}}
\put(55,50){\makebox(0,0)[l]{\texttt{(50,50)}}}

\thicklines % second curve
\qbezier[25](50,50)(50,10)(90,50)
\thinlines % draw lines joining control points
% \put(50,50){\line(0,-1){40}}
% \put(50,10){\line(1,1){40}}
% label control points
\put(50,5){\makebox(0,0)[t]{\texttt{(50,10)}}}
\put(90,55){\makebox(0,0)[b]{\texttt{(90,50)}}}

\end{picture}
\setlength{\unitlength}{1pt}
\caption{Two Bezier curves}
\label{lpicf:bez}
\end{figure}
```

The three points used to specify the position and shape of the Bezier curve are called *control points*. The curve starts at the first control point and is tangent to the line joining the first and second control points. The curve stops at the last control point and is tangent to the line joining the last two control points.

In Figure D.5 the lines joining the control points for the first curve have been drawn in. The locations of all the control points for the two curves are labeled.

The second Bezier curve is the same shape as the first one, but rotated 180 degrees. The first control point of this curve is the same as the last control point of the first curve. This means that the two curves are joind at this point. The line, although it is not drawn, connecting the first two control points of the second curve is in the same direction as the line joining the last two control points of the first curve. This means that the two curves are also tangent at the point where they join. By stringing together several Bezier curves

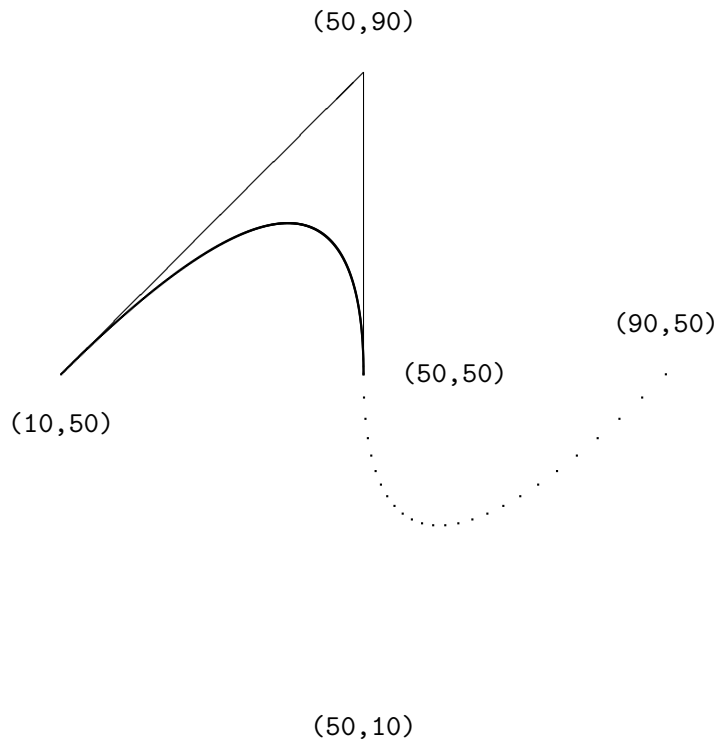


Figure D.5: Two Bezier curves

you can draw quite complex curved shapes.

`\qbeziermax`

The Bezier curves are actually drawn as a linearized form using a series of rectangular blobs of ink. Left to itself, LaTeX will attempt to pick the number of blobs to give the smoothest looking curve, up to a maximum number. (Each blob takes up space in LaTeX's internal memory, and it may run out of space if too many are used in one picture.) The maximum number of blobs per Bezier curve is set by the `\qbeziermax` command. This can be adjusted with the `\renewcommand` command. For example:

```
\renewcommand{\qbeziermax}{250}
```

will set the maximum number of blobs to be 250.

Another method of controlling the number of blobs is by the optional ⟨*num*⟩ argument to the `\qbezier` command. If used, it must be a positive integer number which tells LaTeX exactly how many blobs to use for the curve.

E

LaTeX and TeX

Strictly speaking, LaTeX is a set of macros built on top of the TeX program originally developed by Donald Knuth [Knu86, Knu84] in the early 1980's. TeX is undoubtedly one of the most robust computer programs to date.

Leslie Lamport says that most TeX commands can be used with LaTeX and lists those that cannot be used [Lam94, Appendix E]. Apart from this he says nothing about any TeX commands. I have used some TeX macros in the code examples and so I need to talk a little bit about these.

I like to think of the commands and macros as falling into one of several groups.

- TeX primitives. These are the basic constructs of the TeX language.
- TeX commands or macros. These are part of the plain TeX system and are constructed from the TeX primitives.
- LaTeX kernel commands or macros. These are defined in the LaTeX kernel and are based on plain TeX primitives or commands. In turn, some higher level kernel macros are constructed from more basic aspects of the kernel. The kernel does redefine some of the plain TeX commands.
- Class command. These are mainly built up on the kernel commands but may use some basic TeX.
- Package commands. These are similar to the class commands but are less likely to directly use TeX macros.
- User commands. Typically these are limited to the commands provided by the class and any packages that might be called for, but more experienced users will employ some kernel commands, like `\newcommand`, to make their authoring more efficient.

Although TeX is designed as a language for typesetting it is also a 'Turing complete' language which means that it can perform any function that can be programmed in any familiar programming language. For example, an interpreter for the BASIC language has been written in TeX, but writing this kind of program using TeX is something that only an expert[1] might consider.

Nevertheless, you may have to, or wish to, write a typesetting function of your own. This chapter presents a few of the programming aspects that you may need for this, such

1 Probably also a masochist with plenty of time.

as performing some simple arithmetic or comparing two lengths. For anything else you will have to read one or more of the TeX books or tutorials.

In England witnesses at a trial have to swear to 'Tell the truth, the whole truth, and nothing but the truth'. I will try and tell the truth about TeX but, to misquote Hamlet

> There are more things in heaven and TeX, Horatio,
> Than are dreamt of in your philosophy.

E.1 THE TEX PROCESS

As we are delving deeper than normal and because at the bottom it is the TeX language that does all the work, it is useful to have an idea of how TeX processes a source file to produce a `dvi` file. It is all explained in detail by Knuth [Knu84] and also perhaps more accessibly by Eijkhout [Eij92]; the following is a simplified description. Basically there are four processes involved and the output from one process is the input to the following one.

Input The input process, which Knuth terms the 'eyes', reads the source file and converts what it sees into *tokens*. There are essentially two kinds of token. A token is either a single character such as a letter or a digit or a punctuation mark, or a token is a control sequence. A *control sequence* consists of a backslash and either all the alphabetic characters immediately following it, or the single non-alphabetic following it. Control sequence is the general term for what I have been calling a macro or a command.

Expansion The expansion processor is what Knuth calls 'TeX's mouth'. In this process some of the tokens from the input processor are expanded. Expansion replaces a token by other tokens or sometimes by no token. The expandible tokens include macros, conditionals, and a number of TeX primitives.

Execution The execution process is TeX's 'stomach'. This handles all the tokens output by the expansion processor. Control sequences that are not expandible are termed *executable*, and the execution processor executes the executable tokens. Character tokens are neither expandible nor executable. It handles any macro defintions and also builds horizontal, vertical and mathematical lists.

Layout The layout processor (TeX's 'bowels') breaks horizontal lists into paragraphs, mathematical lists into formulae, and vertical lists into pages. The final output is the `dvi` file.

In spite of the sequential nature implied by this description the overall process includes some feedback from a later process to an earlier one which may affect what that does.

It is probably the expansion processor that needs to be best understood. Its input is a sequence of tokens from the input processor and its output is a sequence of different tokens.

In outline, the expansion processor takes each input token in turn and sees if it is expandible; if it is not it simply passes it on to the output. If the token is expandible then it is replaced by its expansion. The most common expandible tokens are control sequences that have been defined as macros. If the macro takes no arguments then the macro's name is replaced by its definition. If the macro takes arguments, sufficient tokens are collected

to get the values of the arguments, and then the macro name is replaced by the definition. The expansion processor then looks at the first token in the replacement, and if that is expandible it expands that, and so on.

Nominally, the eventual output from the expansion processor is a stream of non-expandible tokens. There are ways, however of controlling whether or not the expansion processor will actually expand an expandible token, and to control the order in which things get expanded, but that is where things get rapidly complicated.

The layout processor works something like this. Ignoring maths, TeX stores what you type in two kinds of lists, vertical and horizontal. As it reads your words it puts them one after another in a horizontal list. At the end of a paragraph it stops the horizontal list and adds it to the vertical list. At the beginning of the next paragraph it starts a new horizontal list and adds the paragraph's words to it. And so on. This results in a vertical list of horizontal lists of words, where each horizontal list contains the words of a paragraph.

It then goes through each horizontal list in turn, breaking it up into shorter horizontal lists, one for each line in the paragraph. These are put into another vertical list, so conceptually there is a vertical list of paragraphs, and each paragraph is a vertical list of lines, and each line is a horizontal list of words, or alternatively one vertical list of lines. Lastly it chops up the vertical list of lines into page sized chunks and outputs them a page at a time.

TeX is designed to handle arbitrary sized inserts, like those for maths, tables, sectional divisions and so forth, in an elegant manner. It does this by allowing vertical spaces on a page to stretch and shrink a little so that the actual height of the typeblock is constant. If a page consists only of text with no widow or orphan then the vertical spacing is regular, otherwise it is likely to vary to some extent. Generally speaking, TeX is not designed to typeset on a fixed grid, but against this other systems are not designed to produce high quality typeset mathematics. Attempts have been made to tweak LaTeX to typeset on a fixed grid but as far as I know nobody has been completely successful.

TeX works somewhat more efficiently than I have described. Instead of reading the whole document before breaking paragraphs into lines, it does the line breaking at the end of each paragraph. After each paragraph it checks to see if it has enough material for a page, and outputs a page whenever it is full. However, TeX is also a bit lazy. Once it has broken a paragraph into lines it never looks at the paragraph again, except perhaps to split it at a page break. If you want to change, say, the width of the typeblock on a particular page, any paragraph that spills over from a previous page will not be reset to match the new measure. This asynchronous page breaking also has an unfortunate effect if you are trying to put a note in say, the outside margin, as the outside is unknown until after the paragraph has been set, and so the note may end up in the wrong margin.

E.2 LaTeX files

The `aux` file is the way LaTeX transfers information from one run to the next and the process works roughly like this.

- The `aux` file is read at the start of the `document` environment. If `\nofiles` has not been specified a new empty `aux` file is then created which has the side effect of destroying the original `aux` file.

- Within the `document` environment there may be macros that write information to the `aux` file, such as the sectioning or captioning commands. However, these macros will not write their information if `\nofiles` has been specified.
- At the end of the `document` environment the contents of the `aux` file are read.

Under normal circumstances new output files are produced each time LaTeX is run, but when `\nofiles` is specified only the `dvi` and `log` files will be new — any other files are unchanged.

In the case of the sectioning commands these write macros into the `aux` file that in turn write information into a `toc` file, and the `\tableofcontents` command reads the `toc` file which contains the information for the Table of Contents. To make this a bit more concrete, as LaTeX processes a new document through the first two runs, the following events occur.

1. Initially there is neither an `aux` nor a `toc` file. At the start of the `document` environment a new empty `aux` file is created.
2. During the first run the `\tableofcontents` typesets the Contents heading and creates a new empty `toc` file.

   During the run sectional commands write information into the new `aux` file. At the end of the `document` environment the `aux` file is read. Contents information in the `aux` file is written to the `toc` file. Lastly all the output files are closed.
3. For the second run the `aux` file from the previous run is read at the start of the `document` environment; no information can be written to a `toc` file because the `toc` file is only made available by the `\tableofcontents` command. The `aux` file from the previous run is closed and the new one for this run is created.

   This time the `\tableofcontents` reads `toc` file that was created during the previous run which contains the typesetting instructions for the contents, and then starts a new `toc` file.

   And so the process repeats itself.

The `aux` file mechanism means that, except for the simplest of documents, LaTeX has to be run at least twice in order to have all the information to hand for typesetting. If sections are added or deleted, two runs are necessary afterwards to ensure that everything is up to date. Sometimes three, or even more, runs are necessary to guarantee that things are settled.

E.3 SYNTAX

The LaTeX syntax that you normally see is pretty regular. Mandatory arguments are enclosed in curly braces and optional arguments are enclosed in square brackets. One exception to this rule is in the `picture` environment where coordinate and direction pairs are enclosed in parentheses.

The TeX syntax is not regular in the above sense. For example, if in LaTeX you said

```
\newcommand*{\cmd}[2]{#1 is no. #2 of}
\cmd{M}{13} the alphabet. % prints: M is no. 13 of the alphabet
```

Then in TeX you would say

```
\def\cmd#1#2{#1 is no. #2 of}
```

and you could then use either of the following calls:

```
\cmd M{13} the alphabet.  % prints: M is no. 13 of the alphabet
\cmd{M}{13} the alphabet. % prints: M is no. 13 of the alphabet
```

A simplistic explanation of the first TeX call of `\cmd` is as follows. A control sequence starts with a backslash, followed by either a single character, or one or more of what TeX thinks of as letters (normally the 52 lower- and upper-case alphabetic characters); a space or any non-letter, therefore, ends a multiletter control sequence. TeX and LaTeX discard any spaces after a macro name. If the macro takes any arguments, and `\cmd` takes two, TeX will then start looking for the first argument. An argument is either something enclosed in braces or a single token. In the example the first token is the character 'M', so that is the value of the first argument. TeX then looks for the second argument, which is the '13' enclosed in the braces. In the second example, both arguments are enclosed in braces.

Here are some TeX variations.

```
\cmd B{2} the alphabet. % prints: B is no. 2 of the alphabet.
\cmd B2 the alphabet.   % prints: B is no. 2 of the alphabet.
\cmd N14 the alphabet.  % prints: N is no. 1 of4 the alphabet.
```

The result of `\cmd B{2}` is as expected. The results of `\cmd B2` and `\cmd N14` should also be expected, and if not take a moment to ponder why. The 'B' and 'N' are the first arguments to `\cmd` in the two cases because a single character is a token. Having found the first argument TeX looks for the second one, which again will be a token as there are no braces. It will find '2' and '1' as the second arguments and will then expand the `\cmd` macro. In the case of `\cmd B2` this gives a reasonable result. In the case of `\cmd N14`, TeX expands `\cmd N1` to produce 'N is in position 1 of', then continues printing the rest of the text, which is '4 the alphabet', hence the odd looking result.

E.4 (La)TeX commands

I have used some TeX commands in the example code and it is now time to describe these. Only enough explanation is given to cover my use of them. Full explanations would require a doubling in the size of the book and a concomitant increase in the price, so for full details consult the *TeXbook* which is the definitive source, or one of the TeX manuals listed in the Bibliography. I find *TeX by Topic* particularly helpful.

I have also used LaTeX commands that are not mentioned by Lamport. LaTeX uses a convention for command names; any name that includes the `@` character is an 'internal' command and may be subject to change, or even deletion. Normal commands are meant to be stable — the code implementing them may change but their effect will remain unaltered. In the LaTeX kernel, and in class and package files the character `@` is automatically treated as a letter so it may be used as part of a command name. Anywhere else you have to use `\makeatletter` to make `@` be treated as a letter and `\makeatother` to make `@` revert to its other meaning. So, if you are defining or modifying or directly using any command that includes an `@` sign then this must be done in either a `.sty` file or if in the document itself it must be surrounded by `\makeatletter` and `\makeatother`.

The implication is 'don't use internal commands as they may be dangerous'. Climbing rocks is also dangerous but there are rock climbers; the live ones though don't try climbing Half Dome in Yosemite or the North Face of the Eiger without having first gained experience on friendlier rocks.

The LaTeX kernel is full of internal commands and a few are mentioned in Lamport. There is no place where you can go to get explanations of all the LaTeX commands, but if you run LaTeX on the `source2e.tex` file which is in the standard LaTeX distribution you will get the commented kernel code. The index of the commands runs to about 40 double column pages. Each class and package introduce new commands over and above those in the kernel.

LaTeX includes `\newcommand`, `\providecommand` and `\renewcommand` as means of (re-)defining a command, but TeX provides only one method.

`\def`⟨*cmd*⟩⟨*arg-spec*⟩`{`⟨*text*⟩`}`

`\def` specifies that within the local group the command `\cmd` is defined as ⟨*text*⟩, and any previous definitions of ⟨*cmd*⟩ within the group are overwritten. Neither the ⟨*text*⟩ nor any arguments can include an end-of-paragraph. The LaTeX equivalent to `\def` is the pair of commands `\providecommand*` followed by `\renewcommand*`.

The ⟨*arg-spec*⟩ is a list of the argument numbers (e.g., `#1#2`) in sequential order, the list ending at the '{' starting the ⟨*text*⟩. Any spaces or other characters in the argument list are significant. These must appear in the actual argument list when the macro is used.

`\long \global`
`\gdef`⟨*cmd*⟩⟨*arg-spec*⟩`{`⟨*text*⟩`}`
`\edef`⟨*cmd*⟩⟨*arg-spec*⟩`{`⟨*text*⟩`}`
`\xdef`⟨*cmd*⟩⟨*arg-spec*⟩`{`⟨*text*⟩`}`

If you use the `\long` qualifier before `\def` (as `\long\def...`) then the ⟨*text*⟩ and arguments may include paragraphs. The LaTeX version of this is the unstarred `\providecommand` followed by `\renewcommand`.

To make a command global instead of local to the current group, the `\global` qualifier can be used with `\def` (as `\global\def...`) when defining it; `\gdef` is provided as a shorthand for this common case.

Normally any macros within the replacement ⟨*text*⟩ of a command defined by `\def` are expanded when the command is called. The macro `\edef` also defines a command but in this case any macros in the replacement ⟨*text*⟩ are expanded when the command is defined. Both `\long` and `\global` may be used to qualify `\edef`, and like `\gdef` being shorthand for `\global\def`, `\xdef` is short for `\global\edef`.

There is much more to the `\def` family of commands than I have given; consult elsewhere for all the gory details.

`\let`⟨*cmda*⟩=⟨*cmdb*⟩

The `\let` macro gives ⟨*cmda*⟩ the same definition as ⟨*cmdb*⟩ *at the time the* `\let` *is called*. The = sign is optional. `\let` is often used when you want to save the definition of a command.

Here is a short example of how some of `\def` and `\let` work.

```
\def\name{Alf}
\let\fred = \name
  \name, \fred.          % prints Alf, Alf.
```

```
\def\name{Fred}
  \name, \fred.          % prints Fred, Alf.
\def\name{\fred red}
  \name, \fred.          % prints Alfred, Alf.
```

\csname ⟨*string*⟩\endcsname

If you have ever tried to define commands like `\cmd1`, `\cmd2` you will have found that it does not work. TeX command names consists of either a single character or a name composed solely of what TeX thinks of as alphabetic characters. However, the `\csname` `\endcsname` pair turn the enclosed ⟨*string*⟩ into the control sequence `\string`, which means that you can create `\cmd1` by

```
\csname cmd1\endcsname
```

Note that the resulting `\cmd1` is not defined (as a macro).

\@namedef{⟨*string*⟩} \@nameuse{⟨*string*⟩}

The kernel `\@namedef` macro expands to `\def\<string>`, where ⟨*string*⟩ can contain any characters. You can use this to define commands that include non-alphabetic characters. There is the matching `\@nameuse` macro which expands to `\<string>` which then lets you use command names that include non-alphabetic characters. For example:

```
\@namedef{fred2}{Frederick~II}
...
\makeatletter\@nameuse{fred2}\makeatother reigned from ...
```

At any point in its processing TeX is in one of six *modes* which can be categorized into three groups:

1. horizontal and restricted horizontal;
2. vertical and internal vertical;
3. math and display math.

More simply, TeX is in either horizontal, or vertical, or math mode. In horizontal mode TeX is typically building lines of text while in vertical mode it is typically stacking things on top of each other, like the lines making up a paragraph. Math gets complicated, and who can do with more complications at this stage of the game?

\hbox to ⟨*dimen*⟩{⟨*text*⟩} \hb@xt@⟨*dimen*⟩{⟨*text*⟩} \vbox to ⟨*dimen*⟩{⟨*text*⟩}

With `\hbox`, ⟨*text*⟩ is put into a horizontal box, and similarly `\vbox` puts ⟨*text*⟩ into a vertical box. The sizes of the boxes depend on the size of the ⟨*text*⟩. The optional `to` ⟨*dimen*⟩ phrase sets the size of the box to the fixed ⟨*dimen*⟩ value. If the ⟨*text*⟩ does not fit neatly inside a fixed size box then TeX will report `overfull` or `underfull` warnings. LaTeX supplies the `\hb@xt@` command as a shorthand for `\hbox to`.

Inside a horizontal box TeX is in restricted horizontal mode which means that everything in the box is aligned horizontally. Inside a vertical box TeX is in internal vertical mode and the contents are stacked up and aligned vertically.

\dp⟨*box*⟩ \ht⟨*box*⟩ \wd⟨*box*⟩

The depth, height and width of a box are returned by the macros \dp, \ht and \wd respectively.

\leavevmode

TeX may be in either vertical or horizontal mode and there are things that can be done in one mode while TeX reports an eror if they are attempted in the other mode. When typesetting a paragraph TeX is in horizontal mode. If TeX is in vertical mode, \leavevmode makes it switch to horizontal mode, but does nothing if TeX is already in horizontal mode. It is often used to make sure that TeX is in horizontal mode when it is unclear what state it might be in.

E.5 CALCULATION

LaTeX provides some methods for manipulating numbers and these, of course, are composed from TeX's more basic methods. Sometimes it is useful to know what TeX itself provides. We have met most, if not all, of LaTeX's macros earlier but I'll collect them all here for ease of reference.

E.5.1 Numbers

In LaTeX a counter is used for storing an integer number.

\newcounter{⟨*counter*⟩} \setcounter{⟨*counter*⟩}{⟨*number*⟩} \stepcounter{⟨*counter*⟩} \refstepcounter{⟨*counter*⟩}

A new counter called ⟨*counter*⟩, without a backslash, is created using \newcounter. Its value can be set to a ⟨*number*⟩ by the \setcounter command and \stepcounter increases its value by one. If the counter is to be used as the basis for a \label, its value must be set using \refstepcounter, neither \stepcounter nor \setcounter will work as expected in this case.

Internally, a LaTeX *counter* is represented by a TeX *count* — the \newcounter macro creates a TeX count named \c@⟨*counter*⟩, and the other \...counter macros similarly operate on the \c@⟨*counter*⟩ count.

\newcount⟨*count*⟩

The TeX \newcount command creates a new count, ⟨*count*⟩, which *does* include an initial backslash. For example

```
\newcount\mycount
```

TeX's method of assigning a number to a count uses nothing like \setcounter.

⟨*count*⟩ [=] ⟨*number*⟩

The [and] enclosing the = sign are there only to indicate that the = sign is optional. For example:

```
\mycount = -24\relax  % \mycount has the value -24
\mycount 36\relax     % now \mycount has the value 36
```

Table E.1: Some internal macros for numbers

\m@ne	-1	\z@	0	\@ne	1
\tw@	2	\thr@@	3	\sixt@@n	16
\@xxxii	32	\@cclv	255	\@cclvi	256
\@m	1000	\@Mi	10001	\@Mii	10002
\@Miii	10003	\@Miv	10004	\@MM	20000

I have added \relax after the digits forming the number for safety and efficiency. When TeX is reading a number it keeps on looking until it comes across something that is not part of a number. There are things that TeX will treat as part of a number which you might not think of, but \relax is definitely not part of a number. See, for example, [Eij92, chapter 7] for all the intricate details if you need them.

There are some numbers that are used many times in the LaTeX kernel and class codes. To save having to use \relax after such numbers, and for other reasons of efficiency, there are commands that can be used instead of typing the digits. These are listed in Table E.1. The command \z@ can be used both for the number zero and for a length of 0pt. Do not use the commands to print a number.

TeX has a limited vocabulary for arithmetic. It can add to a count, and can multiply and divide a count, but only by integers. The result is always an integer. This may be disconcerting after a division where any remainder is discarded. The syntax for these operations is:

\advance⟨*count*⟩ [by] ⟨*number*⟩
\multiply⟨*count*⟩ [by] ⟨*number*⟩
\divide⟨*count*⟩ [by] ⟨*number*⟩

The by is a TeX keyword and the brackets are just there to indicate that it can be missed out. Some examples:

```
\advance\mycount by -\mycount  % \mycount is now 0
\mycount = 15\relax            % \mycount is now 15
\divide\mycount by 4\relax     % \mycount is now 3
\multiply\mycount 4\relax      % \mycount is now 12
\advance\mycount by \yourcount % \mycount is now \yourcount + 12
```

The value of a count can be typeset by prepending the count by the \the command, e.g., \the\mycount.

E.5.2 Lengths

Every length has an associated unit. For convenience I'll use '*dimension*' as shorthand for a number and a length unit.

dimension: ⟨*number*⟩⟨*length-unit*⟩

For example, a *dimension* may be 10pt, or 23mm, or 1.3pc.

Unlike LaTeX, TeX distinguishes two kinds of lengths. A TeX \dimen is a length that is fixed; in LaTeX's terms it is a *rigid* length. On the other hand a TeX \skip is a length that may stretch or shrink a little; it is what LaTeX calls a *rubber* length.

\newdimen⟨*dimen*⟩ \newskip⟨*skip*⟩

The TeX macros `\newdimen` and `\newskip` are used for creating a new ⟨*dimen*⟩ or a new ⟨*skip*⟩. For instance:

```
\newdimen\mydimen
\newskip\myskip
```

The value of a `\dimen` is a *dimension* and the value of a `\skip` is what TeX calls *glue*. It so happens that LaTeX's `\newlength` always creates a new skip — all LaTeX lengths are created as rubber lengths. Glue has at least one and possibly as many as three parts.

glue: *dimension* [`plus` *dimension*] [`minus` *dimension*]

The optional `plus` part is the amount that the glue can stretch from its normal size and the optional `minus` part is the amount the glue can shrink below its normal size. Both `plus` and `minus` are TeX keywords. Glue can never shrink more than the `minus` *dimension* and it normally does not stretch more than the `plus` *dimension*.

`\@plus` `\@minus`

LaTeX supplies `\@plus` and `\@minus` which expand to `plus` and `minus` respectively. Writing `\@plus` instead of `plus` uses one instead of four tokens, saving three tokens, and `\@minus` in place of `minus` saves four tokens — remember that a TeX token is either a control sequence (e.g. `\@minus`) or a single character (e.g., `m`). TeX's memory is not infinite — it can only hold so many tokens — and it makes sense for kernel and class or package writers to use fewer rather than more to leave sufficient space for any that authors might want to create.

In TeX, assigning a value to a length (`\dimen` or `\skip`) is rather different from the way it would be done in LaTeX.

⟨*dimen*⟩ [=] ⟨*dimension*⟩
⟨*skip*⟩ [=] ⟨*glue*⟩

The [and] enclosing the = sign are there only to indicate that the = sign is optional. For example:

```
\newdimen\mydimen
\mydimen = 3pt    % \mydimen has the value 3pt
\mydimen   -13pt  % now \mydimen has the value -13pt
\myskip = 10pt plus 3pt minus 2pt % \myskip can vary between
                                  % 8pt and 13pt (or more)
\myskip = 10pt plus 3pt           % \myskip can vary between
                                  % 10pt and 13pt (or more)
\myskip = 10pt minus 2pt          % \myskip can vary between
                                  % 8pt and 10pt
\myskip = 10pt                    % \myskip is fixed at 10pt
```

Like counts, the value of a length can be typeset by prepending the length by the `\the` command, e.g., `\the\myskip`.

TeX's lengths can be manipulated in the same way as a count, using the `\advance`, `\multiply` and `\divide` macros. Ignoring some details, lengths can be added together but may only be multiplied or divided by an integer number.

```
▷ \Wdimen = 10pt ⇒
                                  Wdimen = 10.0pt
▷ \Wskip = 15pt plus 5pt minus 3pt ⇒
            Wskip = 15.0pt plus 5.0pt minus 3.0pt
▷ \advance\Wskip by \Wskip ⇒
           Wskip = 30.0pt plus 10.0pt minus 6.0pt
▷ \multiply\Wskip by 3 ⇒
          Wskip = 90.0pt plus 30.0pt minus 18.0pt
▷ \divide\Wskip by 17 ⇒
 Wskip = 5.29411pt plus 1.7647pt minus 1.05882pt
▷ \advance\Wskip by \Wdimen ⇒
Wskip = 15.29411pt plus 1.7647pt minus 1.05882pt
▷ \advance\Wdimen by \Wskip ⇒
                               Wdimen = 25.29411pt
```

A length can be multiplied by a fractional number by prepending the length with the number. For example:

```
▷ \Wdimen = 0.5\Wdimen ⇒
           Wdimen = 12.64705pt
▷ \Wskip = 0.5\Wskip ⇒
             Wskip = 7.64705pt
```

When `\multiply` or `\divide` is applied to a `\skip` all its parts are modified, both the fixed part and any elastic components. However, if a `\skip` is multiplied by a fractional number then it loses any elasticity it might have had. In the same vein, if a `\skip` is added to a `\dimen` any elasticity is lost. A `\skip` can be coerced into behaving like a `\dimen` but a `\dimen` is always rigid. For example, typing
'`\Wdimen = 10pt plus 2pt minus 1pt`' results in: 'plus 2pt minus 1pt'.

`\newlength{`⟨*len*⟩`}`

LaTeX's `\newlength` macro creates a new rubber length (internally it uses `\newskip`); there is no LaTeX specific macro to create a rigid length (i.e., a `\dimen`).

LaTeX has a variety of macros for setting or changing its length values.

`\setlength{`⟨*len*⟩`}{`⟨*glue*⟩`}`

The LaTeX `\setlength` macro assigns the value ⟨*glue*⟩ to the rubber length ⟨*len*⟩. Some examples of this are:

```
▷ \setlength{\Wlen}{10pt} ⇒
                                        Wlen = 10.0pt
▷ \setlength{\Wlen}{10pt plus 2pt} ⇒
                                  Wlen = 10.0pt plus 2.0pt
▷ \setlength{\Wlen}{10pt minus 1pt} ⇒
                                 Wlen = 10.0pt minus 1.0pt
▷ \setlength{\Wlen}{10mm plus 2pt minus 1pt} ⇒
                     Wlen = 28.45274pt plus 2.0pt minus 1.0pt
```

As shown in the last example above where both mm and pt are used as a length unit, the `\the` applied to a length always prints the value in pt units.

`\settowidth{`⟨*len*⟩`}{`⟨*text*⟩`}`
`\settoheight{`⟨*len*⟩`}{`⟨*text*⟩`}`
`\settodepth{`⟨*len*⟩`}{`⟨*text*⟩`}`

These put the ⟨*text*⟩ into a box and then set the ⟨*len*⟩ to the width, height and depth respectively of the box.

`\addtolength{`⟨*len*⟩`}{`⟨*glue*⟩`}`

LaTeX's `\addtolength` macro is the equivalent of TeX's `\advance` command. There are no equivalents to TeX's `\multiply` or `\divide` but in any case a length can still be multiplied by prepending it with a fractional number.

`\z@`
`fil fill filll`

`\z@` is a very useful LaTeX command when specifying lengths. Depending on the context it either stands for the number 0 (zero) or 0pt (zero length). TeX has three kinds of infinitely stretchy length units that can be used in the `plus` or `minus` parts of a skip. `fil` is infinitely more flexible than any fixed amount, but `fill` is infitely more flexible than `fil` and `filll` is infinitely more flexible than anything else at all. These infinite glues can be used to push things around.

`\hskip`⟨*skip*⟩
`\vskip`⟨*skip*⟩

The TeX command `\hskip` inserts ⟨*skip*⟩ horizontal space and likewise `\vskip` inserts ⟨*skip*⟩ vertical space.

`\hfil \hfill \hfilneg \hss`

These commands are all TeX primitives and are equivalent to horizontal skips with some kind of infinite glue, as indicated below (note the use of `fil` as a length unit, it being preceeded by a number):

```
\hfil     -> \hskip 0pt plus 1fil
\hfill    -> \hskip 0pt plus 1fill
\hfilneg  -> \hskip 0pt           minus 1fil
\hss      -> \hskip 0pt plus 1fil minus 1fil
```

`\vfil \vfill \vfilneg \vss`

These commands are all TeX primitives and are equivalent to vertical skips with some kind of infinite glue, as indicated below:

```
\vfil     -> \vskip 0pt plus 1fil
\vfill    -> \vskip 0pt plus 1fill
\vfilneg  -> \vskip 0pt           minus 1fil
\vss      -> \vskip 0pt plus 1fil minus 1fil
```

E.6 PROGRAMMING

One of the commonest programming operations is to possibly do one thing if something is true and to possibly do another thing if it is not true. Generally speaking, this is called an 'if-then-else' or *conditional* statement.

`\if...` ⟨*test*⟩ ⟨*true-text*⟩ [`\else` ⟨*false-text*⟩] `\fi`

TeX has several kinds of 'if-then-else' statements which have the general form shown above. The statement starts with an `\if...` and is finished by a matching `\fi`. As usual, the brackets enclose optional elements, so there need be no `\else` portion. The ⟨*true-text*⟩, it it exists, is processed if the ⟨*test*⟩ is `true` otherwise the ⟨*false-text*⟩, if both the `\else` clause and ⟨*false-text*⟩ are present, is processed.

The simplest kind of `\if...` is defined by the `\newif` macro.

`\newif\if`⟨*name*⟩

`\newif` creates three new commands, the `\ifname` and the two declarations, `\nametrue` and `\namefalse`, for setting the value of `\ifname` to `true` or `false` respectively. In this case the ⟨*test*⟩ is embedded in the `\if...`. For example:

```
\newif\ifpeter
...
\ifpeter
  My name is Peter.
\else
  Call me Ishmael.
\fi
```

or a more likely scenario is

```
\newif\ifmine
  \minetrue % or \minefalse
\newcommand{\whose}{%
  \ifmine It's mine. \else I don't know whose it is. \fi}
```

Here are some of the other more commonly used kinds of ifs.

`\ifdim` ⟨*dimen1*⟩ ⟨*rel*⟩ ⟨*dimen2*⟩
`\ifnum` ⟨*number1*⟩ ⟨*rel*⟩ ⟨*number2*⟩
`\ifodd` ⟨*number*⟩

The ⟨*rel*⟩ in `\ifnum` and `\ifdim` is one of the three characters: `<` (less than), `=` (equals), or `>` (greater than). `\ifdim` results in `true` if the two lengths are in the stated relationship otherwise it results in `false`. Similarly `\ifnum` is for the comparison of two integers. The `\ifodd` test is `true` if the integer ⟨*number*⟩ is an odd number, otherwise it results in `false`.

Among other things, the LaTeX class code that organizes the page layout checks if the length values are sensible. The following code is a snippet from the layout algorithm. It checks that the sum of the margins and the width of the typeblock is the same as the width of the page after trimming. `\@tempdima` and `\@tempdimb` are two 'scratch' lengths used in many calculations.

```
\@tempdimb= -1pt              % allow a difference of 1pt
\@tempdima=\paperwidth              % paperwidth
```

```
\advance\@tempdima by -\foremargin  % minus the foremargin
\advance\@tempdima -\textwidth      % minus the textwidth
\advance\@tempdima -\spinemargin    % minus the spinemargin
\ifdim\@tempdima < \@tempdimb       % should be close to zero
  %% error                         % otherwise a problem
\fi
```

Changing the subject, on the offchance that you might want to see how the Fibonacci sequence progresses, the first thirty numbers in the sequence are: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181, 6765, 10946, 17711, 28657, 46368, 75025, 121393, 196418, 317811, 514229 and 832040. I got LaTeX to calculate those numbers for me, and it could have calculated many more. They were produced by just saying `\fibseries{30}`. The French mathematician Édouard Lucas (1842–1891) studied sequences like this and was the one to give it the name Fibonacci. Lucas also invented the game called the Tower of Hanoi with Henri de Parville (1838–1909), supplying the accompanying fable [dP84, RBC74]:

> In the great temple at Benares beneath the dome that marks the center of the world, rests a brass plate in which are fixed three diamond needles, each a cubit high and as thick as the body of a bee. On one of these needles, at the creation, God placed sixty-four discs of pure gold, the largest disc resting on the brass plate, and the others getting smaller and smaller up to the top one. This is the tower of Bramah. Day and night unceasingly the priests transfer the discs from one diamond needle to another according to the fixed and immutable laws of Bramah, which require that the priest on duty must not move more than one disc at a time and that he must place this disc on a needle so that there is no smaller disc below. When the sixty-four discs shall have been thus transferred from the needle which at creation God placed them, to one of the other needles, tower, temple, and Brahmins alike will crumble into dust and with a thunderclap the world will vanish.

The number of separate transfers of single discs is $2^{64} - 1$ or just under eighteen and a half million million moves, give or take a few, to move the pile. At the rate of one disc per second, with no mistakes, it would take more than 58 million million years before we would have to start being concerned.

In his turn, Lucas has a number sequence named after him. There are many relationships between the Fibonacci numbers F_n and the Lucas numbers L_n, the simplest, perhaps, being

$$L_n = F_{n-1} + F_{n+1} \tag{E.1}$$
$$5F_n = L_{n-1} + L_{n+1} \tag{E.2}$$

The first 15 numbers in the Lucas sequence are: 2, 1, 3, 4, 7, 11, 18, 29, 47, 76, 123, 199, 322, 521 and 843. These were produced by saying `\gfibseries{2}{1}{15}`. The Lucas numbers are produced in the same manner as the Fibonacci numbers, it's just the starting pairs that differ.

However, it is the definition of the `\fibseries` and `\gfibseries` macros that might be more interesting in this context.

First, create four new counts. `\fibtogo` is the number of terms to be calculated, `\fib` is the current term, and `\fibprev` and `\fibprevprev` are the two prior terms.

```
\newcount\fib
\newcount\fibprev
\newcount\fibprevprev
\newcount\fibtogo
```

The argument to `\fibseries` is the number of terms. The counts `\fibprevprev` and `\fibprev` are set to the starting pair in the sequence. Provided the number of terms requested is one or more the macro `\@fibseries` is called to do the work.

```
\newcommand*{\fibseries}[1]{%
  \fibprevprev=1\relax
  \fibprev=1\relax
  \ifnum #1>0\relax
    \@fibseries{#1}%
  \fi}
```

The macro `\@fibseries` calculates and prints the terms.

```
\newcommand*{\@fibseries}[1]{%
  \fibtogo=#1\relax
```

It's simple if no more than two terms have been asked for — just print them out.

```
  \ifnum \fibtogo=\@ne
    \the\fibprevprev
  \else
    \ifnum \fibtogo=\tw@
      \the\fibprevprev{} and \the\fibprev
    \else
```

Three or more terms have to be calculated. We reduce the number to be calculated by 2, and print the first two terms.

```
      \advance\fibtogo by -\tw@
      \the\fibprevprev, \the\fibprev
```

We now have to calculate the rest of the terms, where each term is the sum of the two previous terms. The macro `\@fibnext` calculates the next term, prints it out and reduces the number of terms left to be calculated (`\fibtogo`) by one. If there are terms left to be done then the process is repeated until they have all been printed.

```
      \loop
        \@fibnext
      \ifnum \fibtogo>\z@
      \repeat
    \fi
  \fi}
```

The `\@fibnext` macro calculates a term in the series, uses `\printfibterm` to print it, and decrements the `\fibtogo` count.

```
\newcommand*{\@fibnext}{%
```

```
\fib=\fibprev
\advance\fib by \fibprevprev
\fibprevprev=\fibprev
\fibprev=\fib
\printfibterm
\advance\fibtogo \m@ne}
```

The last of the macros, `\printfibterm`, typesets a term in the sequence. If the term is the last one print an 'and' otherwise print a ',', then a space and the term.

```
\newcommand*{\printfibterm}{%
  \ifnum \fibtogo=\@ne \space and \else , \fi
  \the\fib}
```

You have met all of the macros used in this code except for TeX's `\loop` construct. I find the syntax for this a little unusual.

`\loop` ⟨*text1*⟩ `\if...` ⟨*text2*⟩ `\repeat`

The construct starts with `\loop` and is ended by `\repeat`; the `\if...` is any conditional test, but without the closing `\fi`. TeX processes ⟨*text1*⟩, then if the `\if...` is `true` it processes ⟨*text2*⟩ and repeats the sequence again starting with ⟨*text1*⟩. On the other hand, as soon as the result of the `\if...` is `false` the loop stops (i.e., TeX jumps over ⟨*text2*⟩ and goes on to do whatever is after the `\repeat`).

The `\gfibseries` macro that I used for the Lucas numbers is a generalisation of `\fibseries`, where the first two arguments are the starting pair for the sequence and the third argument is the number of terms; so `\gfibseries{1}{1}{...}` is equivalent to `\fibseries{...}`.

```
\newcommand*{\gfibseries}[3]{%
  \fibprevprev=#1\relax
  \fibprev=#2\relax
  \ifnum #3>0\relax
    \@fibseries{#3}%
  \fi}
```

The calculation of the terms in the Fibonacci and in the generalised sequences is the same so `\@fibseries` can be used again.

I used the TeX `\loop` construct in the `\@fibseries` macro but LaTeX has a similar construct.

`\@whilenum` ⟨*ifnum test*⟩ `\do {`⟨*body*⟩`}`
`\@whiledim` ⟨*ifdim test*⟩ `\do {`⟨*body*⟩`}`

As long as the appropriate ⟨*test*⟩ is `true` the ⟨*body*⟩ is processed.

In `\@fibseries` I used `\ifnum`s to check for 3 possible values. There is another `\if...` form that can be used for this type of work.

`\ifcase` ⟨*number*⟩ ⟨*text for 0*⟩ `\or` ⟨*text for 1*⟩ `\or` ⟨*text for 2*⟩
...
`\or` ⟨*text for N*⟩ [`\else` ⟨*text for anything else*⟩] `\fi`

If the ⟨*number*⟩ is 0 then ⟨*text for 0*⟩ is processed, but if ⟨*number*⟩ is 1 then ⟨*text for 1*⟩ is processed, but if ⟨*number*⟩ is ... Each ⟨*text for ...*⟩ is separated by an `\or`. If ⟨*number*⟩ is

anything other than the specified cases (i.e., less than zero or greater than N) then if the `\else` is present ⟨*text for anything else*⟩ is processed.

Here's another version of the `\@fibseries` macro using `\ifcase` and `\@whilenum`.

```
\renewcommand*{\@fibseries}[1]{%
  \fibtogo=#1\relax
  \ifcase \fibtogo % ignore 0
  \or  % \fibtogo=1
    \the\fibprevprev
  \or  % \fibtogo=2
    \the\fibprevprev{} and \the\fibprev
  \else % fibtogo > 2
    \advance\fibtogo by -\tw@
    \the\fibprevprev, \the\fibprev
    \@whilenum \fibtogo > \z@ \do {% must kill space after the {
      \@fibnext}%
  \fi}
```

TeX has more programing constructs than I have shown here and these will be explained in any good TeX book. LaTeX also has more than I have shown but in this case the best place to look for further information is in the LaTeX kernel code, for example in `ltcntrl.dtx`.

F

The terrors of errors

No matter how conscientious you are a mistake or two will occasionally creep into your document source. The good news is that whatever happens TeX will not destroy your files — it may produce some odd looking output, or even no output at all, but your work is safe. The bad news is that you have to correct any errors that TeX finds. To assist you in this TeX stops whenever it comes across what it thinks is an error and tells you about it. If you're not sure what to do it will also provide some possibly helpful advice.

TeX underlies LaTeX which underlies classes and packages. You may get messages than originate from TeX, or from LaTeX, or from the class and any packages you may be using. I'll describe the TeX, LaTeX, and class messages below.

In general, you will see a message on your terminal and LaTeX will stop and wait for you to respond. It prints a question mark and is expecting you to type one of the following:

- ⟨*return*⟩ (or ⟨*enter*⟩ or what is the equivalent on your keyboard): LaTeX will continue processing the document.
- `H` (help): the help message is output and LaTeX waits for you to respond again.
- `S` (scroll): Continue processing, outputting any further error messages, but not stopping.
- `Q` (quiet): Continue processing without stopping and with no further messages.
- `R` (run): Like the `Q` option but not even stopping if your document requires some user input.
- `I` (insert): To insert some material for TeX to read but no changes are made to the source file.
- `E` (edit): This may return you to an editor so you can change the file. What actually happens is system dependent.
- `X` (exit): Stop this LaTeX run.

On the system I am used to the case of the characters does not matter. I must admit that the only ones I have used are ⟨*return*⟩, `q`, `h` and `x`, in approximately that order of frequency.

All messages are output to the `log` file so you can study them later if you need to.

F.1 TEX MESSAGES

The following is an alphabetical list of some of TeX's messages, abbreviated in some cases, together with their corresponding remarks. As an example of how these appear on your terminal, if you had a line in your source that read:
`resulting in $x^3^4$.`
then TeX would output this:

```
! Double superscript
l.102 resulting in $x^3^
                       4^$.
?
```

If you typed `h` in response to this you would then see:

```
I treat `x^1^2' essentially like `x^1{}^2'.
```

TeX's messages start with `!` followed by the particular message text. The second line starts `l.` and a number, which is the number of the line in your file where the error is. This is followed by the text of the line itself up to the point where the error was detected, and the next line in the report shows the rest of the erroneous line. The last line of the report is a `?` and TeX awaits your response.

In the listing I have used `this font for the error message` and *this font for the comment message*.

`! A box was supposed to be here.`
I was expecting to see `\hbox` *or* `\vbox` *or* `\copy` *or* `\box` *or something like that. So you might find something missing in your output. But keep trying; you can fix this later.*

`! Argument of ... has an extra }.`
I've run across a '}' that doesn't seem to match anything. For example, `` `\def\a#1{...}' `` *and* `` `\a}' `` *would produce this error. If you simply proceed now, the* `\par` *that I've just inserted will cause me to report a runaway argument that might be the root of the problem. But if your '}' was spurious, just type '2' and it will go away.*

In LaTeX terms, the example can be translated into
`` `\newcommand{\a}[1]{...}' `` and `` `\a}' ``.

If you can't find the extra } it might be that you have used a fragile command in a moving argument. Footnotes or math in division titles or captions are a fruitful source for this kind of error. You shouldn't be putting footnotes into titles that will get listed in the ToC. For maths, put `\protect` before each fragile command.

`! Arithmetic overflow.`
I can't carry out that multiplication or division, since the result is out of range.

The maximum number that TeX can deal with is 2,147,483,647 and it balks at dividing by zero.

`! Dimension too large.`
I can't work with sizes bigger than about 19 feet. Continue and I'll use the largest value I can.

`! Display math should end with $$.`
The '$' that I just saw supposedly matches a previous '$$'. So I shall assume that you typed '$$' both times.

Although $$ is one of TeX's methods for starting and ending display math, do *not* use it in LaTeX.

`! Double subscript.`
I treat `'x_1_2'` *essentially like* `'x_1{}_2'`.

This would produce x_{1_2}. If you were after say, x_{2_3} instead, type `x_{2_{3}}`.

`! Double superscript.`
I treat `'x^1^2'` *essentially like* `'x^1{}^2'`.

This would produce x^{1^2}. If you were after say, x^{2^3} instead, type `x^{2^{3}}`.

`! (\end occurred inside a group at level ...).`

This is message is output at the end of a run. It means that you have not ended all the groups that you started; a group can be started by a simple open brace ({), but there are other starting mechanisms as well, such as `\begin{...}`. If the problem is a missing `\end{...}`, LaTeX is kind enough to tell you what the mismatch is.

`! (\end occurred when ...  was incomplete).`

`! Extra \fi.` or `Extra \else.` or `Extra \or.`
I'm ignoring this; it doesn't match any `\if`.

`! Extra \endcsname.`
I'm ignoring this, since I wasn't doing a `\csname`.

`! Extra \right.`
I'm ignoring a `\right` *that had no matching* `\left`.

`! Extra }, or forgotten \endgroup, $, or \right.`
I've deleted a group closing symbol because it seems to be spurious, as in '$x}$'. But perhas the } is legitimate and you forgot something else, as in `'\hbox{$x}'`. *In such cases the way to recover is to insert both the forgotten and the deleted material, e.g., by typing 'I$}'.*

The braces or math mode delimeters didn't match. You might have forgotten a `{`, `\[`, `\(` or `$`.

`! Extra ...`
Things are pretty mixed up, but I think the worst is over.

`! Extra alignment tab has been changed to \cr.`
You have given more `\span` *or & marks than there were in the preamble to the* `\halign` *or* `\valign` *now in progress. So I'll asume that you meant to type* `\cr` *instead.*

Internally, LaTeX uses `\halign` for its `array` and `tabular` environments. The message means that you have too many column entries in a row (i.e., too many `&` before the end of the row). Perhaps you have forgotten to put `\\` at the end of the preceding row.

`! File ended while scanning ....` or `Forbidden control sequence found while scanning ....`
I suspect you have forgotten a '}', causing me to read past where you wanted me to stop. I'll try to recover; but if the error is serious you'd better type 'E' or 'X' now and fix your file.

`! Font ...not loadable: Metric (TFM) file not found.`

`! Font ...not loadable: Bad metric (TFM) file.`
I wasn't able to read the size data for this font, so I will ignore the font specification. [Wizards can fix TFM files using TFtoPL/PLtoTF.] You might try inserting a different font spec; e.g., type `'I\font<same font id>=<substitute font name>'.`

LaTeX can't find a font you have asked for.

`! Huge page cannot be shipped out.`
The page just created is more than 18 feet tall or more than 18 feet wide, so I suspect something went wrong.

`! I can't find file '...', please type another.`

TeX couldn't find the file you asked it to read. You can also get this message with LaTeX if you have missed the braces around the argument to `\input`.

`! I can't go on meeting you like this.`
One of your faux pas seems to have wounded me deeply... in fact, I'm barely conscious. Plase fix it and try again.

`! I can't write on file '...', please type another.`

TeX couldn't write on a file, you might have mispelled the name or not have permission to use it.

`! Illegal parameter number in definition of ....`
You meant to type ## instead of #, right? Or maybe a } was forgotten somewhere earlier, and things are all screwed up? I'm going to assume that you meant ##.

This is probably due to a command defining command like `\newcommand` or `\renewcommand` or `\providecommand`, or an environment defining command like `\newenvironment` or `\renewenvironment`, where a `#` has been used incorrectly. Apart from the command `\#`, a `#` can only be used to indicate an argument parameter, like `#3` which denotes the third argument. You cannot use an argument parameter, like the `#3` in the last argument of either the `\newenvironment` or the `\renewenvironment` commands.

You get the same error if you try to include any of the above defining commands inside another one.

`! Illegal unit of measure (replaced by filll).`
I dddon't go any higher than filll.

You have tried to use a `filll` with more than 3 'l's.

`! Illegal unit of measure (mu inserted).`
The unit of measurement in math glue must be mu. To recover gracefully from this error it's best to delete the erroneous units; e.g., type '2' to delete two letters. (See Chapter 27 of The TeXbook.)

TeX was in math mode and expecting a length, which must be in `mu` units.

`! Illegal unit of measure (pt inserted).`
Dimensions can be in units of em, ex, in, pt, pc, cm, mm, dd, cc, bp, or sp; but yours is a new one! I'll assume you meant to say pt, for printers' points. To recover gracefully from this error it's best to delete the erroneous units; e.g., type '2' to delete two letters. (See Chapter 27 of The TeXbook.)

TeX was expecting a length but it found just a number without a known length unit. For example you wrote `2ib` instead of `2in`.

`! Improper \hyphenation will be flushed.`
Hyphenation exceptions must contain only letters and hyphens. But continue; I'll forgive and forget.

`! Incomplete ...all text was ignored after line ....`
A forbidden control sequence occurred in skipped text. This kind of error happens when you say '`\if`...' and forget the matching '`\fi`'. I've inserted a '`\fi`'; this might work.

`! Infinite glue shrinkage found in a paragraph.`
The paragraph just ended includes some glue that has infinite shrinkability, e.g., '`\hskip 0pt minus 1fil`'. Such glue doesn't belong there—it allows a paragraph of any length to fit on one line. But it's safe to proceed, since the offensive shrinkability has been made finite.

`! Limit controls must follow a math operator.`
I'm ignoring this misplaced `\limits` *or* `\nolimits` *command.*

`! Misplaced &.` or `Misplaced \cr.` or `Misplaced \span.`
I can't figure out why you would want to use a tab mark or `\cr` *or* `\span` *here. If you just want an ampersand the remedy is simple: Just type* `I\&` *now. But if some right brace up above has ended a previous alignment prematurely, you're probably due for more error messages, and you might try typing 'S' now just to see what is salvageable.*

In LaTeX the most likely of these messages is the `Misplaced &`. You can only use a naked `&` in environments like `array` and `tabular` as column separators. Anywhere else you have to use `\&`.

`! Misplaced \noalign.`
I expect to see `\noalign` *only after the* `\cr` *of an alignment. Proceed, and I'll ignore this case.*

`! Misplaced \omit.`
I expect to see `\omit` *only after the tab marks or the* `\cr` *of an alignment. Proceed, and I'll ignore this case.*

`! Missing \cr inserted.`
I'm guessing that you meant to end an alignment here.

You might have missed a `\\` at the end of the last row of a `tabular` or `array`.

`! Missing = inserted for ....`
I was expecting to see '<', '=', or '>'. Didn't.

`! Missing # inserted in alignment preamble.`
There should be exactly one # between &'s, when an `\halign` *or* `\valign` *is being set up. In this case you had none, so I've put one in; maybe that will work.*

If you get this in LaTeX then there are problems with the argument to an `array` or `tabular`.

`! Missing $ inserted.` or `Missing \endgroup inserted.` or `Missing \right inserted.` or `Missing } inserted.`
I've inserted something that you may have forgotten. (See the <inserted text> above.) With luck, this will get me unwedged, But if you really didn't forget anything, try typing '2' now; then my insertion and my current dilemma will both disappear.

This is a general response to the above messages. There is also a more specific response for each of the messages, as listed below.

`! Missing $ inserted.`
I've inserted a begin-math/end-math symbol since I think you left one out. Proceed with fingers crossed.

Certain commands can only be executed in math mode and there are others that cannot be used in math mode. TeX has come across a command that cannot be used in the current mode, so it switches into, or out of, math mode on the assumption that that was what you had forgotten to do.

`! Missing \endcsname inserted.`
The control sequence marked <to be read again> should not appear between `\csname` *and* `\endcsname`.

`! Missing { inserted.`
A left brace was mandatory here, so I've put one in. You might want to delete and/or insert some corrections so that I will find a matching right brace soon. If you're confused by all this, try typing 'I}' now.

`! Missing { inserted.`
Where was the left brace? You said something like `\def\a}'`, *which I'm going to interpret as* `\def\a{}'`.

In LaTeX terms, the example wrongdoing would be `\newcommand{\a}}`

`! Missing { inserted.`
I've put in what seems becessary to fix the current column of the current alignment. Try to go on, since this might almost work.

It seems that a `{` might have been missing in a `tabular` or `array` entry.

`! Missing control sequence inserted.`
Please don't say `'\def cs{...}'`, *say* `'\def\cs{...}'`. *I've inserted an inaccessible control sequence so that your definition will be completed without mixing me up too badly. You can recover graciously from this error, if you're careful; see exercise 27.2 in The TeXbook.*

`! Missing delimeter(. inserted).`
I was expecting to see something like `'('` *or* `'\{'` *or* `'\}'` *here. If you typed, e.g.,* `'{'` *instead of* `'\{'` *you should probably delete the* `'{'` *by typing '1' now, so that braces don't get unbalanced. Otherwise just proceed. Acceptable delimeters are characters whose* `\delcode` *is nonnegative, or you can use '*`\delimeter` *<delimeter code>'.*

`! Missing number, treated as zero.`
A number should have been here; I inserted '0'. (If you can't figure out why I needed to see a number, look up 'weird error' in the index to The TeXbook.)

In LaTeX this is often caused by a command expecting a number or a length argument but not finding it. You might have forgotten the argument or an opening square bracket in the text might have been taken as the start of an optional argument. For example, the `\\` (newline) command takes an optional length argument, so the following will produce this error:

```
... next line\\
[Horatio:] ...
```

`! Not a letter.`
Letters in `\hyphenation` *words must have* `\lccode`*>0.*

One or more characters in the argument to the `\hyphenation` command should not be there.

`! Number too big.`
I can only go up to 2147483647 = '17777777777 = "7FFFFFFF, so I'm using that number instead of yours.

These all represent the same value, firstly in decimal, secondly in octal, and lastly in hexadecimal notations.

`! Output loop-- ...consecutive dead cycles.`
I've concluded that your `\output` *is awry; it never does a* `\shipout`*, so I'm shipping* `\box255` *out myself. Next time increase* `\maxdeadcycles` *if you want me to be more patient!*

TeX appears to be spinning its wheels, doing nothing.

`! Overfull \hbox (...pt too wide).`

This is a warning that TeX couldn't cram some text into the alloted horizontal space.

`! Overfull \vbox (...pt too high).`

This is a warning that TeX couldn't find a good place for a pagebreak, so it has put too much onto the current page.

`! Paragraph ended before ...was complete.`

I suspect you've forgotten a '}', causing me to apply this control sequence to too much text. How can we recover? My plan is to forget the whole thing and hope for the best.

Either a blank line or a `\par` command appeared in the argument to a macro that cannot handle paragraphs (e.g., a macro that was defined using `\newcommand*`).

`! Please type a command or say '\end'.`

This is the message that causes me the most trouble. My computer always ignores whatever I say to it and even typing `\end` has no effect. What I usually do, after having tried a few variations like `\end{document}`, is to kill the program by whatever means the operating system provides. Some other possible responses include:

- Type `\stop`
- Type `\csname @@end\endcsname` (LaTeX stores TeX's version of `\end` as `\@@end`)
- Type some macro that you think is unknown, perhaps `\qwertyuiod`, then respond to the error message: `Undefined control sequence.`
- Sometimes nothing works except killing the program. If you are are sure you know how to kill a program, try the following highly contrived code:

```
\documentclass{article}
  \newif\ifland
  \newif\ifprint
  \newcommand{\Xor}[2]{\ifx #1 #2}
\begin{document}
%  \Xor{\ifland}{\ifprint}% try uncommenting this
  \iffalse
\end{document}
```

`! Runaway argument.` or `Runaway definition.` or `Runaway preamble.` or `Runaway text.`

`! Sorry, but I'm not programmed to handle this case.`

I'll just pretend that you didn't ask for it. If you're in the wrong mode, you might be able to return to the right one by typing 'I}' or 'I$' or 'I`\par`'.

`! TeX capacity exceeded, sorry [...].`

If you absolutely need more capacity, you can ask a wizard to enlarge me.

This is dealt with in more detail below.

`! Text line contains an invalid character.`

A funny symbol that I can't read has just been input. Continue, and I'll forget that it ever happened.

The input file contains a nonprinting (control) character; only printing characters should be in the file. Some programs, like word processors, insert invisible characters into their output file. If you have used one of these to prepare your input file, make sure you save it as a plain text file (also known as an ASCII file).

`! That makes 100 errors; please try again.`

`! This can't happen (...).`
I'm broken. Please show this to someone who can fix can fix

This is the message you should never see!

`! Too many }'s.`
You've closed more groups than you opened. Such booboos are generally harmless, so keep going.

There are more closing braces (}) than there are opening braces ({).

`! Unbalanced output routine.`
Your sneaky output routine has fewer real {'s than }'s. I can't handle that very well; good luck.

A package or class has done nasty things to one of LaTeX's most delicate parts — the output routine.

`! Unbalanced write command.`
On this page there's a `\write` *with fewer real {'s than }'s. I can't handle that very well; good luck.*

`! Undefined control sequence.`
The control sequence at the end of the top line of your error message was never `\def`*'ed. If you have misspelled it (e.g., '*`\hobx`*'), type 'I' and the correct spelling (e.g., 'I*`\hbox`*'). Otherwise just continue, and I'll forget whatever was undefined.*

TeX has come across a macro name that it does not know about. Perhaps you misspelled it, or it is defined in a package you did not include. Another possibility is that you used a macro name that included the `@` character without enclosing it between `\makeatletter` and `\makeother` (see §E.4). In this case TeX would think that the name was just the portion up to the `@`.

`! Underfull \hbox (badness ...).`

This is a warning. There might be some extra horizontal space. It could be caused by trying to use two `\newline` or `\\` commands in succession with nothing intervening, or by using a `\linebreak` command or typesetting with the `\sloppy` declaration.

`! Underfull \vbox (badness ...).`

This is a warning that TeX couldn't find a good place for a pagebreak, so it produced a page with too much whitespace on it.

`! Use of ...doesn't match its definition.`
If you say, e.g., `'\def\a1{...}'`*, then you must always put '1' after '*`\a`*', since the control sequence names are made up of letters only. The macro here has not been followed by the required stuff, so I'm ignoring it.*

`! You can't use '...'  in '...'.`

This often manifests itself in the form
`You can't use '\spacefactor' in vertical mode`
the cause is usually trying to use a macro with `@` in its name, typically in the preamble (see §E.4). The solution is to enclose the macro within `\makeatletter` and `\makeatother`.

Another version is
`You can't use 'macro parameter character #' in ... mode.`
In this case you have used a naked `#` in ordinary text; it can only be used in the definition of a macro. In ordinary text you have to use `\#`.

F.1.1 TeX capacity exceeded

TeX has run out of computer space before it finished processing your document. The most likely cause is an error in the input file rather than there really not being enough space — I have processed documents consisting of more than 1400 pages without any capacity problems.

You can very easily make TeX run out of space. Try inputting this:

```
\documentclass{article}
\newcommand*{\fred}{Fred}          % should print 'Fred'
% try to make it print 'Frederick' instead
\renewcommand{\fred}{\fred erick}
\begin{document}
  His name is \fred.
\end{document}
```

and TeX will tell you that it has run out of stack space:

```
! TeX capacity exceeded, sorry [input stack size=15000].
\fred ->\fred
              erick
l.5 His name is \fred
                    .
No pages of output.
Transcript written on errors.log.
```

The offending code above tries to define `\fred` in terms of itself, and TeX just keeps chasing round and round trying to pin down `\fred` until it is exhausted.

At the end of the `log` file for a run, TeX prints the memory space it has used. For example:

```
Here is how much of TeX's memory you used:
 2432 strings out of 60985
 29447 string characters out of 4940048
 106416 words of memory out of 8000001
 5453 multiletter control sequences out of 10000+65535
 8933 words of font info for 31 fonts out of 1000000 for 1000
 276 hyphenation exceptions out of 1000
 26i,11n,21p,210b,380s stack positions out of
            15000i,4000n,6000p,200000b,40000s
```

The error message says what kind of space it exhausted (input stack size in the example above). The most common are:

`buffer size` Can be caused by too long a section or caption title appearing in the ToC, LoF, etc. Use the optional argument to produce a shorter entry.

`exception dictionary` There are too many words listed in `\hyphenation` commands. Remove any that are not actually used and if that doesn't work, remove the less common ones and insert `\-` in the words in the text.

`hash size` The document defines too many command names and/or uses too many cross-referencing `\labels`.

`input stack size` Typically caused by a self-referencing macro definition.

`main memory size` There are three main things that cause TeX to run out of main memory:

- Defining a lot of very long complicated macros.
- Having too many `\index` or `\glossary` commands on a page.
- Creating such a complicated page that TeX cannot hold all it needs to process it.

The solution to the first two problems is to simplify and eliminate. The third is more problematic.

Large `tabulars`, `arrays` and `pictures` (the `\qbezier` command is a memory hog) can gobble up memory. A queue of floats also demands memory space. Try putting a `\clearpage` just before the place where the error occurs and if it still runs out of room then there may be an error in your file, otherwise you did exceed the capacity.

If you have a long paragraph or a long `verbatim` environment try breaking it up, as TeX keeps these in memory until it is ready to typeset them. If you have a queue of floats make sure that you have done your best to help LaTeX find a way to output them (see §10.4) and try adding `\clearpage` at appropriate places to flush the queue.

`pool size` Typically caused by having too many characters in command names and label names.

It can also be caused by omitting the right brace that ends the argument of a counter command (`\setcounter` or `\addtocounter`) or of a `\newenvironment` or `\newtheorem` command.

`save stack size` This happens if commands or environments are nested too deeply. For instance a `picture` that contains a `picture` that includes a `\multiput` that includes a `picture` that includes a ...

F.2 LATEX ERRORS

LaTeX errors introduce themselves differently from those that TeX finds. For example, if you ever happended to use the `\caption` command outside a float, like:

```
\caption{Naked}
```

you would get the message:

```
! LaTeX Error: \caption outside float.

See the LaTeX manual or LaTeX Companion for explanation.
Type H <return> for immediate help.
 ...

l.624 \caption
              {Naked}
?
```

If you then typed `H` in response you would get the following helpful message:

```
You're in trouble here. Try typing <return> to proceed.
If that doesn't work, type X <return> to quit.
?
```

The majority of LaTeX's help messages follow this formula, so I have not noted them in the alphabetical listing below.

`\< in mid line`

A `\<` appears in the middle of a line in a `tabbing` environment; it should only come at the start of a line.

`...  allowed only in math mode`

You have tried to use a math command in a non-math mode.

`Bad \line or \vector argument`

A `\line` or `\vector` has a negative length argument or the slope is not within the allowed range.

`Bad math environment delimeter`

If in math mode there is a start math mode command like `\(` or `\[` or if in LR or paragraph mode there is an end math mode command like `\)` or `\]`. The basic problem is unmatched math mode delimeters or unbalanced braces.

`\begin{...} ended by \end{...}`

The name of the `\begin` argument is not the same as the name of the `\end` argument. This could be caused by a typo or a missing `\end`.

`Can only be used in the preamble`

Some commands can only be used in the preamble, such as `\usepackage`, but there was one of these after the `\begin{document}`.

`\caption outside float`

You have used the `\caption` command outside a float, such as a `figure` or `table` environment.

`Command \... already defined or name \end... illegal`

This is normally because you have used one of the `\new...` commands to define a command or environment or counter name that has already been used; remember also that defining an environment `foo` automatically defines the macro `\foo`. Either choose a new name or use the appropriate `\renew...`; also, see §19.1. In the unlikely event that you have tried to define something beginning with `\end...`, choose another name.

`Command ... invalid in math mode`

You have used a non-math command in math mode.

`Command ... not provided in base LaTeX2e`

You have tried to use a symbol that is not part of basic LaTeX. Try loading the latexsym or amsfonts package which might define the symbol.

`Counter too large`

You are using a non-numeric counter representation, such as letters or footnote symbols, and the counter has exceeded the allowed number (for example there are only 26 alphabetic characters).

`Environment ... undefined`

LaTeX does not know the name of the argument of a `\begin`. You have probably misspelled it.

`File not found. Type X to quit or <RETURN> to proceed or enter new name (Default extension: ...)`

LaTeX cannot find the file you requested. The extension `tex` results from a problematic `\input` or `\include`; the extension `sty` from a `\usepackage` and an extension `cls` from a `\documentclass`.

`Float(s) lost`

Usually caused by having too many `\marginpar`s on a page.

`Illegal character in array argument`

There is an illegal character in the argument of an `array` or `tabular` environment, or in the second argument of a `\multicolumn` command.

`\include cannot be nested`

A file that is `\included` cannot `\include` any other files.

`\LoadClass in package file`

This is an error in a package file you are using (you can only use `\LoadClass` in a class file). Complain to the author.

`Lonely \item -- perhaps a missing list environment`

An `\item` command appears to be outside any list environment.

`Missing \begin{document}`

If you haven't forgotten `\begin{document}` then there is something wrong in the preamble as LaTeX is trying to typeset something before the document starts. This is often caused by missing the backslash from a command, misplaced braces round an argument, a stray character, or suchlike.

`Missing @-exp in array argument`

The `@` character is not followed by an @-expression in the argument of an `array` or `tabular` environment, or in the second argument of a `\multicolumn` command.

`Missing p-arg in array argument`

There is a `p` not followed by braces in the argument of an `array` or `tabular` environment, or in the second argument of a `\multicolumn` command.

`No counter ... defined`

The argument to a `\setcounter` or `\addtocounter` command, or in the optional argument to `\newcounter` or `\newtheorem` is not the name of a counter. Perhaps you misspelled the name. However, if the error occured while an `aux` file was being read then you might well have used a `\newcounter` in an `\included` file.

`No room for a new ...`

TeX is limited in the numbers of different things it can handle. You might not recognize the thing that the message mentions as some of them are hidden in LaTeX. The LaTeX `counter` uses a TeX `\count` for example, and a length is a TeX `\skip`. Most things are limited to a maximum of 256 but there can be no more than 16 files open for reading and 16 for writing.

`No \title given`

You did not put a `\title` command before using `\maketitle`.

`Not in outer par mode`

There is a float (e.g., a `figure` or a `\marginpar`) in math mode or in a parbox (e.g., in another float).

`Option clash for ...`

The same package was used twice but with different options. It is possible for one package to use another package which might be the cause if you can't see anything obvious.

`Page height already too large`

You are trying to use `\enlargethispage` when the page is already too large.

`\pushtabs and \poptabs don't match`

There are unmatched `\pushtabs` and `\poptabs` in a `tabbing` environment.

`\RequirePackage or \LoadClass in Options Section`

This is a problem in a class or package file. Complain to the author.

`Something's wrong -- perhaps a missing \item`

This can be caused by not starting a list environment, such as `itemize` with a `\item` command, or by omitting the argument to the `thebibliography` environment. There are many other non-obvious causes, such as calling some macro that ends up using `\addvspace` or `\addpenalty` when not in `vmode`.

`Suggested extra height (...)  dangerously large`

LaTeX is concerned that you a trying to increase the page size too much with the `\enlargthispage` command.

`Tab overflow`

In the `tabbing` environment a `\=` has exceeded LaTeX's maximum number of tab stops.

`The file needs format ...  but this is ...`

The document uses a document class or package that is not compatible with the version of LaTeX you are using. If you are using only standard files then there is a problem with your LaTeX installation.

`There's no line to end here`

A `\newline` or `\\` appears in vertical mode, for example between paragraphs. Or perhaps you have tried to put `\\` immediately after an `\item` to start the text on a new line. If this is the case, then try this:

```
\item \mbox{} \\
...
```

`This may be a LaTeX bug`

This is a message you don't want to see as it is produced by the output routine — perhaps the most obscure part of LaTeX. It is probably due to an earlier error. If it is the first error, though, and you can't see anything wrong, ask for somebody's help.

`Too deeply nested`

There are too many list environments nested within each other. At least four levels are usually available but some list environments are not obvious (for example the `quotation` environment is actually a list).

`Too many columns in eqnarray environment`

An `eqnarray` environment has three `&` column separators with no `\\` between.

`Too many unprocessed floats`

There may be too many `\marginpar`s to fit on a page, but it's more likely that LaTeX hasn't been able to find locations for printing all the figures or tables. If one float cannot be placed, all later ones are saved until LaTeX runs out of storage space. See §10.4 for details on how LaTeX decides to place a float.

`Two \documentclass commands`

Your document has two `\documentclass` commands; only one is permitted.

`Two \LoadClass commands`

This is an error in the class file. Complain to the author.

`Undefined tab position`

A `\>`, `\+`, `\-`, or `\<` tabbing command is trying to move to a tab position that has not been defined by a `\=` command.

`Unknown option ... for class/package ...`

You have asked for an option that the class or package does not know about. Perhaps you have mispelled something, or omitted a comma.

`\usepackage before \documentclass`

In general, the `\usepackage` command can only be used in the preamble.

`\verb ended by end of line`

The argument of a `\verb` command runs past the end of the line. Perhaps you forgot to put in the correct ending character.

`\verb illegal in command argument`

A `\verb` cannot be part of the argument to another command.

F.3 LATEX WARNINGS

Most warnings are given at the point in the document where a potential problem is discovered, while others are output after the document has been processed.

For example, the following code

```
... \ref{joe}... \cite{FRED96} ...
```

may produce warnings like

```
Latex Warning: Reference ‘joe’ on page 12 undefined
               on input line 881.
Latex Warning: Citation ‘FRED96’ on page 12 undefined
               at lines 890--897.
```

during the document processing, and then at the end there will also be the warning:

```
LaTeX Warning: There were undefined references.
```

Some warning messages pinpoint where a problem might lie, as in the citation warning above, while others make no attempt to do so. In the alphabetical listing that follows I have not included such information, even if it is supplied.

`Citation ... on page ... undefined`

The key in a `\cite` command was not defined by any `\bibitem`.

`Citation ... undefined`

The key in a `\cite` command was not defined by any `\bibitem`.

`Command ... invalid in math mode`

The command is not permitted in math mode but was used there anyway. Remember that font size commands and `\boldmath` or `\unboldmath` cannot be used in math mode.

`Float too large for page by ...`

A float (table or figure) is too tall to fit properly on a page by the given amount. It is put on a page by itself.

`Font shape ... in size ... not available size ... substituted`

You asked for a font size that was not available. The message also says what font is being used instead.

`Font shape ... undefined using ... instead`

You asked for a font shape that was not available. The message also says what font is being used instead.

`h float specifier changed to ht` or `!h float specifier changed to !ht`

A float has an optional `h` or `!h` argument but as it wouldn't fit on the curent page it has been moved to the top pf the next page.

`Label ... multiply defined`

Two `\label` or `\bibitem` commands have the same argument (at least during the previous LaTeX run).

`Label(s) may have changed. Rerun to get cross-references right`

This is only output at the end of the run.

One of the numbers printed by `\cite`, `\ref` or `\pageref` commands might be incorrect because the correct values have changed since the preceding LaTeX run.

`Marginpar on page ... moved`

A `\marginpar` was moved down the page to avoid overwriting an earlier one. The result will not be aligned with the `\marginpar` call.

`No \author given`

There is no `\author` command before calling `\maketitle`.

`No positions in optional float specifier. Default added (so using 'tbp')`

You have used an empty optional argument to a float, for example:
`\begin{figure}[]`
so it has used
`\begin{figure}[tbp]`
instead.

`Optional argument of \twocolumn too tall on page ...`

The contents of the optional argument to `\twocolumn` was too long to fit on the page.

`\oval, \circle, or \line size unavailable`

You have asked for too large (or too small) an oval or circle, or too short a line, in a `picture`.

`Reference ... on page ... undefined`

The argument of a `\ref` or `\pageref` has not been defined on the preceding run by a `\label` command.

`Size substitutions with differences up to ... have occured. Please check the transcript file carefully and redo the format generation if necessary!`

This is only output at the end of the run.

Some fonts have had to be used as substitutes for requested ones and they are a different size.

`Some shapes were not available, defaults substituted`

This is only output at the end of the run.

At least one font had to be substituted.

`Text page ... contains only floats`

The page should have included some textual material but there was no room for it.

`There were multiply defined labels`

This is only output at the end of the run.

Two or more `\label` or `\cite` commands had the same argument.

`There were undefined references`

This is only output at the end of the run.

There was at least one `\ref` or `\pageref` or `\cite` whose argument had not been defined on the preceding run by a `\label` or `\biblabel` command.

`Unused global option(s) [...]`

The listed options were not known to the document class or any packages you used.

`You have requested release ... of LaTeX but only release ... is available`

You are using a class or package that requires a later release of LaTeX than the one you are using. You should get the latest release.

`You have requested version ... of class/package ... but only version ... is available`

You (or the class or one of the packages you are using) needs a later release of a class or package than the one you are using. You should get the latest release.

F.4 CLASS ERRORS

The class errors introduce themselves differently from those that LaTeX finds. Instead of starting with
`! LaTeX Error:`
the class errors start with
`! Class memoir Error:`
After that, it is indistinguishable from a LaTeX error. For example, if you ever happened to input the next line as line 954 in your document you would get the error message that follows
`\sidecapmargin{either}`

```
! Class memoir Error: Unrecognized argument for \sidecapmargin.

See the memoir class documentation for explanation.
Type H <return> for immediate help.
 ...

l.954 \sidecapmargin{either}
?
```

If you then typed `H` (or `h`) in response you would get the following helpful message:

```
Try typing <return> to proceed.
If that doesn't work, type X <return> to quit.
?
```

The majority of the help messages follow this formula, so I have not noted them in the alphabetical listing below.

`... is negative`

The value is negative. It should be at least zero.

`... is not a counter`

An argument that should be the name of a counter is not.

`... is zero or negative`

The value must be greater than zero.

`>{...} at wrong position: token ignored`

A `>{...}` in the argument to an `array` or `tabular` is incorrectly placed and is being ignored.

`<{...} at wrong position: changed to !{...}`

A `<{...}` in the argument to an `array` or `tabular` is incorrectly placed. It has been changed to `!{...}` instead.

`A pattern has not been specified`

You are trying to use the `patverse` or `patverse*` environment without having first defined a pattern.

`Argument to \setsidecappos is not t or c or b`

The argument will be assumed to be `c`.

`Argument to \overridesidecapmargin neither left nor right`

The argument to `\overridesidecapmargin` must be either `left` or `right`. The attempted override will be ignored.

`Cannot change a macro that has delimited arguments`

You are using `patchcmd` on a macro that has delimted arguments.

`Empty preamble: ‘l’ used`

The argument to an `array` or `tabular` is empty. The specification `{l}` is being used instead.

`Font command ... is not supported`

You have tried to use a deprecated font command. Either replace it with the current font command or declaration or use the oldfontcommands class option.

`\footskip is too large for \lowermargin by ...`

The `\footskip` is too large for the `\lowermargin`. Either increase the `\lowermargin` or decrease the `\footskip`.

`\headheight and/or \headsep are too large for \uppermargin by ...`

The sum of the `\headheight` and the `\headsep` is larger than the `\uppermargin`. Either increase the `\uppermargin` or reduce the others.

`Illegal pream-token (...): ‘c’ used`

An illegal character is used in the argument to an `array` or `tabular`. The ‘c’ specifier is being used instead (which centers the column).

`Index ... outside limits for array ...`

Trying to access an index for the array data structure that is not between the low and high indices.

`Limits for array ... are in reverse order`

The low index is not less than the high index in `\newarray`.

`Missing arg: token ignored`

The argument to a column specifier for a `array` or `tabular` is missing.

`No array called ...`

You have tried to access an unknown array data structure.

`Not defined: ...`

You are using `\patchcmd` on a macro that is not defined.

`Not redefinable: ...`

You are using `\patchcmd` on a macro that it is unable to modify.

`Only one column-spec. allowed`

There can only be one column specifier in a `\multicolumn`.

`Optional argument is not one of: classic, fixed, lines, or nearest. I will assume the default.`

You have provided an unknown name for the optional argument to `\checkthelayout`. The default `classic` will be used instead.

`\paperheight and/or \trimtop are too large for \stockheight by ...`

The sum of the `\paperheight` and the `\trimtop` is larger than the `\stockheight`. Either increase the `\stockheight` or reduce the others.

`\paperwidth and/or \trimedge are too large for \stockwidth by ...`

The sum of the `\paperwidth` and the `\trimedge` is larger than the `\stockwidth`. Either increase the `\stockwidth` or reduce the others.

`\spinemargin and/or \textwidth and/or \foremargin are too large for \paperwidth by ...`

The sum of the `\spinemargin` and the `\textwidth` and the `\foremargin` is larger than the `\paperwidth`. Either increase the `\paperwidth` or reduce the others.

`The combination of argument values is ambiguous. The lengths will be set to zero`

The combination of values in the arguments to one of the commands for page layout does not make sense.

`The 'extrafontsizes' option is required to use the '...pt' option`

If you want to use a '...pt' class option greater than 25pt you also have to use the extrafontsizes option. The class will use the 17pt option.

`Unknown document division name (...)`

You have used an unknown division name in the argument to `\settocdepth` or `\setsecnumdepth` and friends. If you haven't mistyped it you will have to use `\setcounter` instead.

`Unknown mark setting type '...' for ...mark`

In `\createmark` or `\createplainmark` the mark setting type should have been `left` or `both` or `right`. The class will use `both`.

`Unknown numbering type ... for ...mark`

In `\createmark` the class expected either `shownumber` or `nonumber` for displaying the number. It will use `shownumber`.

`Unrecognized argument for \sidecapmargin`

The argument to `\sidecaption` should be `left` or `right` or `inner` or `outer`.

`\uppermargin and/or \textheight and/or \lowermargin are too large for \paperheight by ...`

The sum of the `\uppermargin` and the `\textheight` and the `\lowermargin` is larger than the `\paperheight`. Either increase the `\paperheight` or reduce the others.

`You have used the '*pt' option but file ... can't be found`

You have used the *pt option but the corresponding `clo` file can't be found. Check your definitions of `\anyptfilebase` and `\anyptsize`. The `mem10.clo` file will be used instead.

`XeTeX is required to process this document`

The document needs to be processed via XeTeX. Try using `xelatex` instead of `(pdf)latex`, or try removing any XeTeX packages from the document.

F.5 CLASS WARNINGS

These are introduced by `Class memoir Warning:`

For example `\addtodef{alf}{\joe}{fred}` will produce a message along the lines of:

```
Class memoir Warning: 'alf' is not a macro on input line 91.
```

while `\addtodef{\joe}{alf}{fred}` might produce:

```
Class memoir Warning: '\joe' is not a macro on input line 97.
```

The following is an alphabeticised list of the class warnings.

`... at index ... in pattern ... is not a digit`

The character at the given position in the verse pattern is not a digit.

`... is not a macro`

Using `\addtodef` or `\addtoiargdef` you have tried to extend the definition of an unknown macro.

`... is not an input stream`

You are trying to access a non-existent input stream.

`... is not an output stream`

You are trying to access a non-existent output stream.

`Bad \sidebarmargin argument`

The argument to `\sidebarmargin` is not recognized. The class will use `right`.

`Characters dropped after \end{...}`

At the end of a `verbatim` environment there should be no characters after the `\end{...}` on the same line.

`Column ... is already defined`

The column type has been defined by a previous `\newcolumntype`.

`Counter ... already defined`

For information only, the counter in `\providecounter` is already defined.

`Do not use \footnote in \maketitle. Use \thanks instead`

You cannot use `\footnote` in any of the `\maketitle` elements (i.e., `\title` or `\author` or `\date`) but you can use `\thanks`.

`Empty 'thebibliography' environment`

There are no `\bibitems` in the `thebibliography` environment.

`Environment ... already defined`

For information only, the environment in `\provideenvironment` is already defined.

`Index ... for pattern ... is out of bounds`

The index for the verse pattern is either too low or too high.

`Input stream ... is already defined`

You are trying to use `\newinputstream` to create an already existing input stream.

`Input stream ... is not open`

You are trying to access or close an input stream that is closed.

`Input stream ... is open`

You are trying to open an input stream that is already open.

`Length ... already defined`

For information only, the length in `\providelength` is already defined.

`Marginpar on page ... moved by ...`

A marginal note has been lowered by the given amount to avoid overwriting a previous note; the moved note will not be aligned with its `\marginpar`. (This is a more informative message than the normal LaTeX one.)

`No more to read from stream ...`

There is nothing left in the stream to be read.

`Optional argument of \twocolumn too tall on page ...`

The contents of the optional argument to `\twocolumn` was too long to fit on the page.

`Output stream ... is already defined`

You are trying to use `\newoutputstream` to create an already existing output stream.

`Output stream ... is not open`

You are trying to access or close an output stream that is closed.

`Output stream ... is open`

You are trying to open an output stream that already open.

`Redefining primitive column ...`

The argument to `\newcolumntype` is one of the basic column types.

`Stream ... is not open`

You are trying to access a stream, either input or output, that is closed.

`The ... font command is deprecated. Use ... or ... instead`

You are using a deprecated font command. Consider using one of the alternatives.

`The counter will not be printed. The label is: ...`

The optional ⟨*style*⟩ argument to the `enumerate` environment does not include one of the special characters.

`Undefined index file ...`

You are trying to add an index entry to an unknown `idx` file.

`Unknown toclevel for ...`

The division name you have used for `\settocdepth` is not recognized.

`\verb may be unreliable inside tabularx`

A `\verb` in a `tabularx` may work, but may not.

`X columns too narrow (table too wide)`

The width of the X columns in a `tabularx` had to be made too narrow.

G

Comments

G.1 ALGORITHMS

Over time we may use this section to explain, or list some of the algorithms for some of the macros in the class. The information may be useful to some.

G.1.1 Autoadjusting `\marginparwidth`

This algorithm is used within `\fixthelayout` unless the user have used `\setmarginnotes`.

```
if twocolumn then:
  marginparwidth = min{inner margin,outer margin}
else
  if twoside then:
    if marginpar always left or always right then:
      marginparwidth = min{inner margin,outer margin}
    else if marginpar in outer margin then:
      marginparwidth = outer margin
    else if marginpar in inner margin then:
      marginparmargin = inner margin
    end if
  else
    if marginpar in left margin then:
      marginparwidth = inner margin
    else
      marginparwidth = outer margin
    end if
  end if
end if
marginparwidth = marginparwidth - 2marginparsep
if marginparwidth < 1pt then:
  marginparwidth = 1pt
end if
```

Notes

CHAPTER 3 TEXT AND FONTS

['work in progress'] (page 35) XeTeX enables you to use Opentype fonts with LaTeX, and supports both left-to-right and right-to-left typesetting. It has become very popular with those involved in linguistics and non-Latin scripts.

[using the appropriate package] (page 35) I have found Christopher League's *TeX support for the FontSite 500 CD*, obtainable from http:contrapunctus.net/fs500tex, extremely useful in providing packages for a wide range of PostScript fonts for me to use. You do have to buy a CD containing the sources of the fonts from FontSite (http://www.fontsite.com); it cost me a total of $37.12, including taxes and shipping, in 2002 for 512 PostScript and TrueType professional quality fonts that are legal and very reasonably priced.

Many of the fonts fall into the Decorative/Display category but the book fonts include:

Blackletter Alte Schwabacher, Engravers Old English, Fette Fraktur, Fette Gotisch, and Olde English.

Uncial/Mediaeval American UncialXXAmerican Uncial, Linden, and Rosslaire.

Geralde/Venetian Bergamo (also known as Bembo), Caslon, Garamond, Goudy Old Style, Jenson Recut (also known as Centaur), URW Palladio (also known as Palatino), Savoy (also known as Sabon), Schnittger, University Old Style, and Vendome.

Transitional URW Antiqua, Baskerville, Century Old Style, ATF Clearface, English Serif, Jessica (also known as Joanna), Lanston Bell, New Baskerville, and Nicholas Cochin.

Modern/Didone Basel (also known as Basilia), Bodoni, Modern, and Walbaum.

Free Form Barbedour, Bernhard Modern, Della Robbia, Engravers Litho, Flanders, and Lydian.

Sans Serif There are over 20 in this category but some of the ones I am most familiar with are: Chantilly (also known as Gill Sans), Franklin Gothic, Function (also known as Futura), Lanston Koch, News Gothic, Opus (also known as Optima), Struktor (also known as Syntax), and Unitus (also known as Univers).

Slab Serif Cheltenham, Clarendon, Egyptian, Glytus (also known as Glypha), URW Latino (also known as Melior), Litho Antique, Serific (also known as Serifa).

Script There are some sixteen Script fonts.

Decorative There are over fifty Decorative fonts.

Symbol There are a dozen miscellaneous symbol fonts which include, among others, arrows, borders, fleurons and various icons.

CHAPTER 12 PAGE NOTES

[Put no mark . . . finally printed] (page 252) This manual uses both footnotes and endnotes. For identifying the endnotes I have used the 'words' method for identifying the parent location of an endnote, so as not to start confusing the reader with two sets of note marks in the body of the text. Typically either footnotes or endnotes are used, not both, so the question of distinguishing them does not normally arise.

Command summary

`*pt` Class option for an author-defined size for the body font. 3
`\:` A medium space (4/18 em). 342
`\@docsep{`⟨*num*⟩`}` 146
Distance, as ⟨*num*⟩ math units, bewteen dots in the dotted lines in the ToC etc.
`\@dottedtocline{`⟨*level*⟩`}{`⟨*indent*⟩`}{`⟨*numwidth*⟩`}` 148
For a ToC, (LoF, LoT) entry at ⟨*level*⟩ specifies the ⟨*indent*⟩ and ⟨*numwidth*⟩ and draws a dotted line between the title and page number.
`\@fnsymbol{`⟨*num*⟩`}` 243
Converts ⟨*num*⟩ to the footnote marker representation.
`\@makefnmark` 240
Typesets the footnote marker where `\footnote` is called.
`\@pnumwidth{`⟨*length*⟩`}` Space for a page number in the ToC etc. 146
`\@thefnmark` Value of the footnote marker. 240
`\@tocrmarg{`⟨*length*⟩`}` Right hand margin for titles in the ToC etc. 146
`\\[`⟨*length*⟩`]` Ends a verse line, and adds ⟨*length*⟩ vertical space. 266
`\\*[`⟨*length*⟩`]` 266
Ends a line while preventing a pagebreak, and adds ⟨*length*⟩ vertical space.
`\\>[`⟨*length*⟩`]` Shorthand for `\verselinebreak`. 267
`10pt` Class option for a 10pt body font. 2
`11pt` Class option for a 11pt body font. 2
`12pt` Class option for a 12pt body font. 2
`14pt` Class option for a 14pt body font. 3
`17pt` Class option for a 17pt body font. 3
`20pt` Class option for a 20pt body font. 3
`25pt` Class option for a 25pt body font. 3
`30pt` Class option for a 30pt body font. 3
`36pt` Class option for a 36pt body font. 3
`48pt` Class option for a 48pt body font. 3
`60pt` Class option for a 60pt body font. 3
`9pt` Class option for a 9pt body font. 2

`a3paper` Class option for A3 stock paper size. 1
`a4paper` Class option for A4 stock paper size. 1
`a5paper` Class option for A5 stock paper size. 1
`a6paper` Class option for A6 stock paper size. 1
`\abnormalparskip{`⟨*length*⟩`}` 49
Sets the inter-paragraph spacing to ⟨*length*⟩.
`\abovecaptionskip` Vertical space above a caption. 207
`\aboverulesep` Space above a tabular rule. 222

\abslabeldelim{⟨text⟩} 72
⟨text⟩ is typeset immediately after \abstractname in a run-in heading.
\absleftindent Indentation of the left of the abstract text. 72
\absnamepos 73
Position of a non run-in title for an abstract (flushleft, center, or flushright).
\absparindent Paragraph indent in the abstract environment. 72
\absparsep Paragraph separation in the abstract environment. 72
\absrightindent Indentation of the right of the abstract text. 72
\abstitleskip Space around the title of an abstract. 73
\begin{abstract} Environment for typesetting an abstract. 71
\abstractcol 72
Declares that an abstract in a twocolumn document should be typeset like an unnumbered chapter.
\abstractintoc 72
Specifies that the abstact's title is to be added to the ToC.
\abstractname An abstract's title. 72
\abstractnamefont 72
Font for typesetting and abstract's title (\abstractname).
\abstractnum 72
Specifies that an abstract is to be typeset like a numbered chapter.
\abstractrunin 72
Specifies that the title of an abstract should be set as a run-in heading.
\abstracttextfont Font for typesetting the body text of an abstract. 72
\addappheadtotoc Adds ToC entry with the title \appendixtocname. 78
\addcontentsline{⟨file⟩}{⟨kind⟩}{⟨text⟩} 146
Writes heading/caption data to the ⟨file⟩ in the form of the \contentsline macro.
\added{⟨change-id⟩} 326
Change mark for someting added; ⟨change-id⟩ is put in the margin.
\addlinespace{⟨width⟩} 223
Puts extra space, default \defaultaddspace, between a pair of tabular rows.
\addtocontents{⟨file⟩}{⟨text⟩} Inserts ⟨text⟩ into ⟨file⟩ (ToC, etc). 148
\addtodef{⟨macro⟩}{⟨prepend⟩}{⟨append⟩} 332
Inserts ⟨prepend⟩ at the start of the current definition of ⟨macro⟩ and ⟨append⟩ at the end, treating the result as if it had been defined by \renewcommand.
\addtodef*{⟨macro⟩}{⟨prepend⟩}{⟨append⟩} 333
Inserts ⟨prepend⟩ at the start of the current definition of ⟨macro⟩ and ⟨append⟩ at the end, treating the result as if it had been defined by \renewcommand*.
\addtoiargdef{⟨macro⟩}{⟨prepend⟩}{⟨append⟩} 332
Inserts ⟨prepend⟩ at the start of the current definition of ⟨macro⟩ (which takes a single argument) and ⟨append⟩ at the end, treating the result as if it had been defined by \renewcommand.
\addtoiargdef*{⟨macro⟩}{⟨prepend⟩}{⟨append⟩} 333
Inserts ⟨prepend⟩ at the start of the current definition of ⟨macro⟩ (which takes a single argument) and ⟨append⟩ at the end, treating the result as if it had been defined by \renewcommand*.
\addtonotes{⟨text⟩} Inserts ⟨text⟩ into the endnotes ent file. 255
\addtopsmarks{⟨pagestyle⟩}{⟨prepend⟩}{⟨append⟩} 116
Inserts ⟨prepend⟩ and ⟨append⟩ before and after the current definition of \makepsmarks for ⟨pagestyle⟩.
\addtostream{⟨stream⟩}{⟨text⟩} 293
Adds ⟨text⟩ to the file associated with the output ⟨stream⟩.
\begin{adjustwidth}{⟨left⟩}{⟨right⟩} 135
Temporarily adds the lengths {⟨left⟩} and {⟨right⟩} to the left and right margins.

\begin{adjustwidth*}{⟨left⟩}{⟨right⟩} 135
A sophisticated form of adjustwidth. Temporarily adds the lengths {⟨left⟩} and {⟨right⟩} to the spine and outer margins on odd (recto) pages, and on even (verso) pages adds them to the outer and spine margins, respectively.
\afterbookskip Spacing below a \book title. 81
\afterchapskip Space after chapter title. 86
\afterchapternum 86
Called after printing a chapter number; by default inserts \midchapskip space.
\afterchaptertitle 86
Called after printing a chapter title; by default inserts \afterchapskip space.
\afterepigraphskip Vertical space after an epigraph. 259
\afterbookskip Spacing below a \part title. 81
\afterPoemTitle Called after printing the title of a \PoemTitle. 271
\afterPoemTitlenum 271
Called after printing the number of a \PoemTitle.
\afterPoemTitleskip Vertical space after a poem title 272
\afterXtitle 152
Generic macro called after typesetting the title of the 'X List of'.
\aliaspagestyle{⟨alias⟩}original 112
Defines the ⟨alias⟩ pagestyle to be the same as the ⟨original⟩ pagestyle.
\alsoname Wording for a *see also* index entry. 314
\begin{altverse} Alternate lines in the stanza are indented. 269
\amname 330
Abbreviation for ante meridiem used in \printtime* (default am)
\and Use within the argument to \author to separate author's names. 67
\andnext Use within the argument to \author to insert a newline.. 67
\anyptfilebase 4
First part of the name of the clo file for the *pt class option (default mem).
\anyptsize 4
Second part (the pointsize) of the name the clo file for the *pt class option (default 10).
\begin{appendices} 78
An environment form of \appendix; chapters are restored to their condition (including numbering) after the environment ends.
\appendix 78
Sets chapter numbering to alphabetic and sets the chapter name to \appendixname.
\appendixname 78
Name given to chapters in appendices (default Appendix).
\appendixpage 78
Creates an unnumbered anonymous part-like page with the title \appendixpagename and adds it to the ToC.
\appendixpage* Like \appendixpage but makes no ToC entry. 78
\appendixpagename 78
Title used for an \appendixpage (default Appendices.
\appendixrefname Name for an appendix used by \Aref. 302
\appendixtocname 78
Title of ToC entry added by \addappheadtotoc (default Appendices.
\Aref{⟨labstr⟩} 302
Prints a named (\appendixrefname) reference to a \labeled appendix.
\begin{array}[⟨pos⟩]{⟨preamble⟩} 211
Environment for setting math elements in an array form.
\arraybackslash Use instead of \\ in a tabular column. 228

\arraycolsep Half the space between columns in an array. 230
\arrayrulewidth 230
Width of lines (e.g., \hline, \vline, etc.) in an array or tabular.
\arraystretch 230
Multiplication factor for the normal row spacing in tabulars and arrays.
\arraytostring{⟨arrayname⟩}{⟨result⟩} 340
Defines the macro ⟨result⟩ to be the sequence of characters in the array ⟨arrayname⟩.
article Class option for simulating the article class. 5
article 88
Chapter style similar to a section head in the article class but with different sized fonts.
\AtBeginClass{⟨pack⟩}{⟨code⟩} 347
Inserts ⟨code⟩ just before the ⟨class⟩ class is used.
\AtBeginFile{⟨filename⟩}{⟨code⟩} 346
Inserts ⟨code⟩ just before the ⟨filename⟩ is input (or included, etc.).
\AtBeginPackage{⟨pack⟩}{⟨code⟩} 347
Inserts ⟨code⟩ just before the ⟨pack⟩ package is used.
\atcentercr Breaks a line in a (unusual) paragraph. 131
\AtEndClass{⟨class⟩}{⟨code⟩} 347
Inserts ⟨code⟩ just after the ⟨class⟩ class is used.
\AtEndFile{⟨filename⟩}{⟨code⟩} 346
Inserts ⟨code⟩ just after the ⟨filename⟩ is input (or included, etc.).
\AtEndPackage{⟨pack⟩}{⟨code⟩} 347
Inserts ⟨code⟩ just after the ⟨pack⟩ package is used.
\atendtheglossaryhook 322
Vacuous macro called as the first thing by \end{theglossary}.
\author{⟨text⟩} 67
Used by \maketitle to typeset ⟨text⟩ as the document author.
\autocols[⟨width⟩]{⟨pos⟩}{⟨num⟩}{⟨style⟩}{⟨entries⟩} 234
Lists the ⟨entries⟩ down ⟨num⟩ columns in a tabular fashion.
\autorows[⟨width⟩]{⟨pos⟩}{⟨num⟩}{⟨style⟩}{⟨entries⟩} 233
Lists the ⟨entries⟩ in rows across ⟨num⟩ columns in a tabular fashion.

b3paper Class option for B3 stock paper size. 1
b4paper Class option for B4 stock paper size. 1
b5paper Class option for B5 stock paper size. 1
b6paper Class option for B6 stock paper size. 1
\backmatter 75
Prohibits sectional numbering and floats, etc., will be numbered continuously.
\baselineskip (length) 23
The normal vertical distance between the baselines of adjacent lines of text (depends on the size of the font).
\beforebookskip Spacing above a \book title. 81
\beforechapskip Space above chapter name and number. 86
\beforeepigraphskip Vertical space before an epigraph. 259
\beforepartskip Spacing above a \part title. 81
\beforePoemTitleskip Vertical space before a poem title. 272
\begintheglossaryhook 322
Vacuous macro called as the last thing by \begin{theglossary}.
\belowcaptionskip Vertical space below a caption. 207
\belowrulesep Space below a tabular rule. 222
\bfseries Declaration for using a bold font. 42

bianchi A two line, centered chapterstyle with rules above and below. 90
`\bibintoc` Title of `thebibliography` will be added to the ToC. 305
`\bibitem[`⟨*label*⟩`]{`⟨*labstr*⟩`}` 307
Introduces an entry in the bibliography. The ⟨*labstr*⟩ argument corresponds to a `\cite`'s ⟨*labstr*⟩ argument. The optional ⟨*label*⟩ overides the default numerical printed entry label.
`\bibitem` 305
Starts a new bibliographic entry in a `thebibliography` listing.
`\bibitemsep` Vertical space between entries in a bibliography. 307
`\bibliography{`⟨*bibfile-list*⟩`}` 307
Print the bibliography having used BibTeX to extract entries from the ⟨*bibfile-list*⟩ of comma-separated names of `bib` files.
`\bibliographystyle{`⟨*style*⟩`}` 307
Typeset the bibliographic entries according to ⟨*style*⟩.
`\biblistextra` Called immediately after the `bibitemlist` is set up. 306
`\bibmark` Can be used in pagestyles for page headers in a bibliography. 306
`\bibname` The title for a bibliography 305
`\bibsection` Initialises the bibliography and typesets the title. 306
`\bicaption[`⟨*label*⟩`]{`⟨*short1*⟩`}{`⟨*long1*⟩`}{`⟨*NAME*⟩`}{`⟨*long2*⟩`}` 197
A bilingual caption in a float but only the first added to the 'List of…'.
`\bicontcaption{`⟨*long1*⟩`}{`⟨*NAME*⟩`}{`⟨*long2*⟩`}` 198
A continued bilingual caption.
`\bionenumcaption[`⟨*label*⟩`]{`⟨*short1*⟩`}{`⟨*long1*⟩`}{`⟨*NAME*⟩`}{`⟨*short2*⟩`}{`⟨*long2*⟩`}` 196
A bilingual caption with both captions numbered in the float but only the first in the 'List of…'.
`\bitwonumcaption[`⟨*label*⟩`]{`⟨*short1*⟩`}{`⟨*long1*⟩`}{`⟨*NAME*⟩`}{`⟨*short2*⟩`}{`⟨*long2*⟩`}` 196
A bilingual caption with both captions numbered in the float and in the 'List of…'.
`\begin{blockdescription}` 137
A list of descriptions of `\item`s formatted as indented block paragraphs.
`\blockdescriptionlabel{`⟨*label*⟩`}` 137
Specifies the format of the ⟨*label*⟩ of an `\item` in a `blockdescription` environment.
`\book{`⟨*title*⟩`}` 77
Typesets a numbered book ⟨*title*⟩ on a page by itself, adding ⟨*title*⟩ to the ToC.
`\bookblankpage` 82
Follow a book title page with a blank page (the default).
`\booknamefont` Font used by `\printbookname` for the book name. 81
`\booknamenum` Called between printing a book name and number. 81
`\booknumberline{`⟨*num*⟩`}` Typeset book number in ToC 156
`\booknumberlinebox{`⟨*length*⟩`}{`⟨*code*⟩`}` 156
The box command used to typeset the book number within the ToC, note that it will automatically align to the left
`\booknumberlinehook{`⟨*num*⟩`}` 156
The first thing to be called within `\booknumberline`, does nothing by default.
`\booknumfont` Font used by `\printbooknum` for the book number. 81
`\bookpageend` Code to finish off typesetting a book title page. 82
`\bookpagemark{`⟨*title*⟩`}` 82
For setting any marks with the title from a `\book` for a running header.
`\bookrefname` Name for a book used by `\Bref`. 302
`\booktitlefont` Font used by `\printbooktitle` for the title. 82
`\bottomrule[`⟨*width*⟩`]` 222
Draws a rule across a tabular, default width `\heavyrulewidth`.
`\bottomsectionpenalty` 77
Penalty on a `\raggedbottomsection` short page.

\bottomsectionskip 77
Amount of stretch on a \raggedbottomsection short page.
\begin{boxedverbatim} 287
May put a box around the verbatim material; lines may be numbered and page breaks are allowed.
\begin{boxedverbatim*} 287
May put a box around the verbatim* material; lines may be numbered and page breaks are allowed.
\boxedverbatiminput{⟨file⟩} 294
Acts like boxedverbatim except the contents is read from the ⟨file⟩ file.
\boxedverbatiminput*{⟨file⟩} 294
Acts like boxedverbatim* except the contents is read from the ⟨file⟩ file.
\break TeX macro to break a line. 131
\Bref{⟨labstr⟩} 302
Prints a named (\bookrefname) reference to a \labeled book.
bringhurst 90
A raggedright, unnumbered, small caps chapterstyle with a textwidth rule below.
broadsheetpaper Class option for broadsheet stock paper size. 2
brotherton 90
A chapterstyle like the default but with the number spelled out.
\bs prints \. 349
\bvbox 288
Rectangular boxes will be drawn for boxedverbatim environments.
\bvboxsep 288
Separation between text and framing in boxedverbatim environments.
\bvendofpage{⟨text⟩} 288
Use ⟨text⟩ as the boxedverbatim page break marker at the bottom of a page.
\bvendrulehook 288
Called at the end of a boxedverbatim environment, and before a pagebreak.
\bvleftsidehook 288
Called before each line in a boxedverbatim environment.
\bvnumbersinside 289
Line numbers typeset inside a boxedverbatim box.
\bvnumbersoutside 289
Line numbers typeset outside a boxedverbatim box.
\bvperpagefalse Do not mark page breaks in a boxedverbatim. 288
\bvperpagetrue 288
Visibly break a boxedverbatim at a page break using \bvtopofpage and \bvendofpage.
\bvrightsidehook 288
Called after each line in a boxedverbatim environment.
\bvsides 288
Draw vertical rules on each side of boxedverbatim environments.
\bvtopandtail 288
Draw horizontal rules before and after boxedverbatim environments.
\bvtopmidhook 288
Called after \bvtoprulehook at the start of a boxedverbatim environment.
\bvtopofpage{⟨text⟩} 288
Use ⟨text⟩ as the boxedverbatim page break marker at the top of a page.
\bvtoprulehook 288
Called at the start of a boxedverbatim environment and after a pagebreak.

`\calccentering{`⟨*length*⟩`}` 137
Sets the ⟨*length*⟩ command to the value to add/subtract from margins to center text on the physical page.
`\cancelthanksrule` 70
Specifies that the `\footnoterule` is to be used from now on.
`\caption[`⟨*short*⟩`]{`⟨*long*⟩`}` 206
Typeset a caption with title ⟨*long*⟩, and add it, or ⟨*short*⟩ instead if given, to a 'List of…'.
`\captiondelim{`⟨*delim*⟩`}` 188
Specifies ⟨*delim*⟩ to be the delimeter between the number and title in a caption.
`\captionnamefont{`⟨*fontspec*⟩`}` 188
Set the font for the first part (name and number) of a caption, upto and including the delimeter.
`\captionstyle[`⟨*short*⟩`]{`⟨*style*⟩`}` 188
Set the paragraph style for the caption. The optional ⟨*short*⟩ is the style for captions shorter than a full line.
`\captiontitlefinal{`⟨*text*⟩`}` 189
⟨*text*⟩ will be put immediately at the end of a caption title, but will not appear in a 'List of…'.
`\captiontitlefont{`⟨*fontspec*⟩`}` Set the font for the caption title text. 188
`\captionwidth{`⟨*length*⟩`}` Set the caption width to ⟨*length*⟩. 189
`\cardinal{`⟨*number*⟩`}` Typesets `{`⟨*number*⟩`}` as a cardinal number. 336
`\begin{center}` 129
Text to be set raggedleft and raggedright, with each line centered.
`\centerfloat` 129
Within a float, centers it with respect to the typeblock; the float may extend into both margins.
`\centering` 129
Declaration for text to be set raggedleft and raggedright, with each line centered.
`\cftaddnumtitleline{`⟨*ext*⟩`}{`⟨*kind*⟩`}{`⟨*num*⟩`}{`⟨*title*⟩`}{`⟨*page*⟩`}` 160
Writes a `\contentsline` to the 'List of…' ⟨*ext*⟩ file for a ⟨*kind*⟩ entry with number ⟨*number*⟩ and ⟨*title*⟩ and ⟨*page*⟩ number.
`\cftaddtitleline{`⟨*ext*⟩`}{`⟨*kind*⟩`}{`⟨*title*⟩`}{`⟨*page*⟩`}` 160
Writes a `\contentsline` to the 'List of…' ⟨*ext*⟩ file for a ⟨*kind*⟩ entry with ⟨*title*⟩ and ⟨*page*⟩ number.
`\cftbeforeKskip` 154
Generic vertical space before a 'K' entry in a 'List of…'.
`\cftbookbreak` Starts a `\book` entry in the ToC. 154
`\cftchapterbreak` Starts a `\chapter` entry in the ToC. 154
`\cftdot` The 'dot' in the dotted leaders in 'List of…'. 153
`\cftdotsep` 153
The separation between dots in a dotted leader in a 'List of…'.
`\cftinsertcode{`⟨*name*⟩`}{`⟨*code*⟩`}` 161
Defines Toc (LoF, LoT) ⟨*name*⟩ insertion to be ⟨*code*⟩.
`\cftinserthook{`⟨*file*⟩`}{`⟨*name*⟩`}` 161
Inserts code ⟨*name*⟩ into ⟨*file*⟩ (e.g., `toc`, `lof`, etc.
`\cftKafterpnum` 157
Called after typesetting the page number of a 'K' entry in a 'List of…'.
`\cftKaftersnum` 155
Called immediately after typesetting the number of a 'K' entry in a 'List of…'.
`\cftKaftersnumb` 155
Called immediately after typesetting the number box of a 'K' entry in a 'List of…'.
`\cftKdotsep` 156
Separation between dots in a leader between the title and page number of a 'K' entry in a 'List of…'.

\cftKfont 155
Controls the appearance of the number and title of a 'K' entry in a 'List of…'.
\cftKformatpnum{⟨pnum⟩} 157
Typesets the page number ⟨pnum⟩ of a 'K' entry in a 'List of…'.
\cftKformatpnumhook{⟨num⟩} 157
When formatting the page number in the ToC (via \cftKformatpnum) this hook is given the page value. Does nothing by default
\cftKindent 154
Generic indent of an 'K' entry from the left margin in a 'List of…'.
\cftKleader 156
Leader between the title and page number of a 'K' entry in a 'List of…'.
\cftKname 155
Called as the first element of the 'K' entry line in a 'List of…'.
\cftKnumwidth 155
Generic space for the number of a 'K' entry in a 'List of…'.
\cftKpagefont Font for the page number of a 'K' entry in a 'List of…'. 157
\cftKpresnum 155
Called immediately before typesetting the number of a 'K' entry in a 'List of…'.
\cftlocalchange{⟨ext⟩}{⟨pnumwidth⟩}{⟨tocrmarg⟩} 160
Writes commands to the ⟨ext⟩ 'List of…' file resetting \@pnumwidth and \@tocrmarg to the specified values.
\cftnodots 153
A separation between dots in a dotted leader in a 'List of…' that is too large for any dots to occur.
\cftpagenumbersoff{⟨kind⟩} 159
Eliminates page numbers for the ⟨kind⟩ entries in a 'List of…'.
\cftpagenumberson{⟨kind⟩} 159
Reverses the effect of \cftpagenumbersoff.
\cftparfillskip Fills the last line in a paragraph in a 'List of…'. 159
\cftparskip The \parskip to be used in a 'List of…'. 153
\cftpartbreak Starts a \part entry in the ToC. 154
\cftsetindents{⟨kind⟩}{⟨indent⟩}{⟨numwidth⟩} 157
Set the ⟨kind⟩ entry *indent* to ⟨indent⟩ and its *numwidth* to ⟨numwidth⟩.
\cftwhatismyname 156
Since \numberline is shared by \section and siblings, \cftwhatismyname can be used to tell which section type is calling \numberlinehook
\changecaptionwidth 189
Captions will be set within the width specified by \captionwidth.
\changed{⟨change-id⟩} 326
Change mark for someting changed; ⟨change-id⟩ is put in the margin.
\changeglossactual[⟨file⟩]{⟨char⟩} 321
Specifies ⟨char⟩ as the actual character for glossary ⟨file⟩.
\changeglossnum[⟨file⟩]{⟨thecounter⟩} 321
Specifies ⟨thecounter⟩ as the ⟨num⟩ for glossary ⟨file⟩.
\changeglossnumformat[⟨file⟩]{⟨format⟩} 321
Specifies ⟨format⟩ as the format for ⟨num⟩ for glossary ⟨file⟩.
\changeglossref[⟨file⟩]{⟨thecounter⟩} 321
Specifies ⟨thecounter⟩ as the ⟨ref⟩ for glossary ⟨file⟩.
\changemarks Print change marks. 325
\chapindent 90
A length you can use in a defining chapter style (or anything else).

\chapnamefont Font used by \printchaptername. 86
\chapnumfont Font used by \printchapternum. 86
chappell 90
A centered chapterstyle with a rule between the number line (in a roman font) and the title line in italics.
\chapter[⟨*toc-title*⟩][⟨*head-title*⟩]{⟨*title*⟩} 77
Starts a new page and puts the chapter number and ⟨*title*⟩ at the top of the page, adding ⟨*title*⟩ to the ToC and possibly the running headers. If given ⟨*toc-title*⟩ is used instead of ⟨*title*⟩ for the ToC and running header. If given ⟨*head-title*⟩ is used for a running header. It restarts numbering of any subsidiary elements such as \section of floats.
\chapter*[⟨*head-title*⟩]{⟨*title*⟩} 77
Starts a new page and puts an unnumbered chapter ⟨*title*⟩ at the top of the page. If given ⟨*head-title*⟩ is used for a running header.
\chapterheadstart 86
Called at start of printing a chapter header; by default inserts \beforechapskip space.
\chapternamenum 86
Inserted between printing the chapter name and number. It is usually a space.
\chapternumberline{⟨*num*⟩} Typeset chapter number in ToC 156
\chapternumberlinebox{⟨*length*⟩}{⟨*code*⟩} 156
The box command used to typeset the chapter number within the ToC, note that it will automatically align to the left
\chapternumberlinehook{⟨*num*⟩} 156
The first thing to be called within \chapternumberline, does nothing by default.
\chapterprecis{⟨*text*⟩} Prints ⟨*text*⟩ and also adds it to the ToC. 93
\chapterprecishere{⟨*text*⟩} Typesets ⟨*text*⟩ for a chapter precis. 93
\chapterprecistoc{⟨*text*⟩} Adds ⟨*text*⟩ for a chapter precis to the ToC. 93
\chapterrefname Name for a chapter used by \Cref. 302
\chapterstyle{⟨*style*⟩} Sets chapter heading layout to ⟨*style*⟩. 87
\chaptitlefont Font used by \printchaptertitle. 86
\checkandfixthelayout[⟨*algorithm*⟩] 23
Command to check and implement the page layout specifications, adjusting the \textheight using ⟨*algorithm*⟩ (classic, fixed, lines, or nearest, the default being classic) for the calculation.
\checkarrayindex{⟨*arrayname*⟩}{⟨*index*⟩} 340
Checks if ⟨*index*⟩ is a valid index value for the array ⟨*arrayname*⟩. Sets \ifbounderror true if there is a problemn otherwise false.
\checkifinteger{⟨*num*⟩} 341
If ⟨*num*⟩ is an integer and not less than zero, sets \ifinteger true, otherwise false.
\checkoddpage 334
Sets \ifoddpage true if called on an odd-numbered page, otherwise false.
\checkthelayout[⟨*algorithm*⟩] 23
Command to check the page layout specifications, adjusting the \textheight using ⟨*algorithm*⟩ (classic, fixed, lines, or nearest, the default being classic) for the calculation.
\circle{⟨*diam*⟩} Picture object of a open circle diameter ⟨*diam*⟩. 425
\circle*{⟨*diam*⟩} 425
Picture object of a black closed circle diameter ⟨*diam*⟩.
\cite[⟨*detail*⟩]{⟨*labstr-list*⟩} 305
Citation in the text to bibliographic items specified in the ⟨*labstr-list*⟩ of comma-separated bibliographic identifiers; optional information, e.g., page number, is supplied via ⟨*detail*⟩.
\citeindexfile File name for the citation index. 318
\clearforchapter Selects a chapter's opening page. 84

\cleartoevenpage[⟨text⟩] 261
Clears the current page and moves to the next verso page; the optional ⟨text⟩ is put on the skipped page (if there is one).
\cleartooddpage[⟨text⟩] 335
Flushes any pending floats to then start typesetting on the next odd page. The optional ⟨text⟩ is put on the skipped page (if there is one).
\cleartorecto Simpler form of \cleartooddpage. 335
\cleartoverso Simpler form of \cleartoevenpage. 335
\closeinputstream{⟨stream⟩} 294
Detaches and closes the file associated with the input ⟨stream⟩.
\closeoutputstream{⟨stream⟩} 293
Detaches and closes the file associated with the output ⟨stream⟩.
\cmd{⟨cmd⟩} 349
where ⟨cmd⟩ is a macro name like \cmd, prints and indexes \cmd.
\cmdprint{⟨cmd⟩} where ⟨cmd⟩ is a macro name like \cmd, prints \cmd. 349
\cmidrule[⟨width⟩](⟨trim⟩){⟨m–n⟩} 223
Draws a rule, default thickness \cmidrulewidth, across tabular columns ⟨m⟩ to ⟨n⟩; the ends may be ⟨trim⟩ed by \cmidrulekern.
\cmidrulekern Trim amount for \cmidrule. 223
\cmidrulewidth Default width for a \cmidrule tabular rule. 223
\colorchapnum Color for the number in the *pedersen* chapterstyle. 404
\colorchaptitle Color for the title in the *pedersen* chapterstyle. 404
\columnsep (length) Width of the gutter between columns. 22
\columnseprule (length) Thickness of the rule in the gutter. 22
\begin{comment} Skip over the environment. 284
\commentsoff{⟨name⟩} 284
Process contents of the ⟨name⟩ comment environment.
\commentson{⟨name⟩} 284
Skip contents of the ⟨name⟩ comment environment.
companion Chapterstyle like those in the *LaTeX companion* series. 87
\contcaption{⟨text⟩} 191
A continued caption, replacing the original title with ⟨text⟩.
\contentsline{⟨kind⟩}{⟨text⟩}{⟨page⟩} 146
An entry in a "List of..." file for a ⟨kind⟩ entry, with ⟨text⟩ being the title which was on ⟨page⟩.
\contentsname The title for the Table of Contents. 151
\continuousmarks 68
Specifies that the thanks/footnote marker is not zeroed after titling.
\continuousnotenums 253
Declaration to make the numbering of endnotes continuous throughout the document.
\contsubbottom[⟨list-entry⟩][⟨subtitle⟩]{⟨text⟩} 200
A continued \subbottom.
\contsubcaption[⟨list-entry⟩]{⟨subtitle⟩} A continued \subcaption. 200
\contsubtop[⟨list-entry⟩][⟨subtitle⟩]{⟨text⟩} A continued \subtop. 200
\copypagestyle{⟨copy⟩}original 112
Creates a new pagestyle called ⟨copy⟩ using the ⟨original⟩ pagestyle specification.
\counterwithin{⟨ctr⟩}{⟨within⟩} 331
Makes the counter ctr (created via \newcounter) act as though it had been initially defined to be reset by counter within. It also redefines \thectr to include \thewithin.
\counterwithin*{⟨ctr⟩}{⟨within⟩} 331
Makes the counter ctr (created via \newcounter) act as though it had been initially defined to be reset by counter within, leaving the original definition of \thectr.

\counterwithout{⟨ctr⟩}{⟨within⟩} 331
Makes the counter ctr (created via \newcounter{⟨ctr⟩}[⟨within⟩]) act as though it had been initially defined via \newcounter{⟨ctr⟩}. It also redefines \thectr to typeset as arabic numerals.
\counterwithout*{⟨ctr⟩}{⟨within⟩} 331
Makes the counter ctr (created via \newcounter{⟨ctr⟩}[⟨within⟩]) act as though it had been initially defined via \newcounter{⟨ctr⟩}, leaving the original definition of \thectr.
\cplabel 334
Prefix for labels used by \checkoddpage odd/even page checking.
\createmark{⟨sec⟩}{⟨marks⟩}{⟨show⟩}{⟨prefix⟩}{⟨postfix⟩} 114
Defines the \secmark macro where ⟨show⟩ (shownumber or nonumber) controls whether the division number will be displayed within \mainmatter, ⟨marks⟩ is left, both or right, and ⟨prefix⟩ and ⟨postfix⟩ are affixed before and after the ⟨sec⟩ (division) number.
\createplainmark{⟨type⟩}{⟨marks⟩}{⟨text⟩} 114
Defines the \typemark macro using ⟨text⟩ as the mark, where ⟨marks⟩ is left, both or right.
\Cref{⟨labstr⟩} 302
Prints a named (\chapterrefname) reference to a \labeled chapter.
crosshead A centered chapterstyle in a bold font. 91
crownvopaper Class option for crown octavo stock paper size. 2
\cs{⟨name⟩} prints \name. 349
\begin{ctabular}[⟨pos⟩]{⟨format⟩} 232
Like tabular except that it will continue over a page break.
culver One line, centered, bold chapterstyle using Roman numerals. 91
\currenttitle Gives the title of the last sectional division command. 304

\dashbox{⟨len⟩}(⟨width,height⟩)[⟨pos⟩]{⟨text⟩} 420
A picture object framing ⟨text⟩ (at position ⟨pos⟩) in a dashed box ⟨width⟩ by ⟨height⟩. The dashes and spaces are each ⟨len⟩ long.
dash 91
Two line, centered, regular font, chapterstyle. The number has a dash on either side.
\date{⟨text⟩} 67
Used by \maketitle to typeset ⟨text⟩ as the document date.
dbillpaper Class option for dollar bill stock paper size. 2
\defaultaddspace Default space for \addlinespace. 223
default The default book class chapterstyle. 87
\defaultlists 140
Declaration specifying the default vertical spacing list-based environments.
\defaultsecnum Declaration reversing the effect of \hangsecnum. 97
\deleted{⟨change-id⟩} 326
Change mark for someting deleted; ⟨change-id⟩ is put in the margin.
\DeleteShortVerb{⟨backslash-char⟩} 285
Returns ⟨char⟩ to its normal meaning instead of being a shorthand for \verb⟨char⟩.
demo2 91
A two line chapterstyle with a large sanserif title; the number is above, centered and written (e.g., Six instead of 6) in a bold font. There are rules above and below the title.
demo3 91
A two line chapterstyle with a large sanserif title; the number is above, centered and written (e.g., Six instead of 6) in an italic font. There are rules above and below the title line. It is a modified version of the *demo2* style.
demyvopaper Class option for demy octavo stock paper size. 2

\begin{description} 137
A list of descriptions of \items formatted as regular paragraphs.
\descriptionlabel{⟨label⟩} 137
Specifies the format of the ⟨label⟩ of an \item in a description environment.
\DisemulatePackage{⟨package⟩} 346
Undo a previous \EmulatedPackage or \EmulatedPackageWithOptions for the ⟨package⟩ package.
\doccoltocetc 151
Set the ToC, etc., in one or two columns according to the class option.
\doublerulesep Space between adjacent rules in a tabular, or an array. 222
\begin{DoubleSpace} Environment form of \DoubleSpacing 51
\begin{DoubleSpace*} Environment form of \DoubleSpacing* 51
\DoubleSpacing Declaration doubling the baselineskip. 50
\DoubleSpacing* 50
Same as \DoubleSpacing but also effects page notes and floats.
dowding 91
A centered two line chapterstyle in an italic font and the number is spelled out.
\downbracefill Fills a tabular column with a down brace. 224
draft Class option for draft document. 5
\dropchapter{⟨length⟩} Lowers subsequent chapter heads by ⟨length⟩. 260
\droptitle 65
Length controlling the position of \maketitle on the page (default 0pt).

\easypagecheck 334
\checkoddpage will use a possibly inaccurate but fast method for checking for an odd page number.
ebook Class option for elecronic book stock size. 1
ell 91
A raggedleft large sanserif chapterstyle with the number in the margin. An 'L' shaped rule separates the number and title lines.
\eminnershape{⟨shape⟩} 44
Font shape for emphasized text within emphasized text.
\emph{⟨text⟩} Use a change in font to emphasise ⟨text⟩. 43
\emptythanks Discards any text from previous uses of \thanks. 67
\EmulatedPackage{⟨package⟩}[⟨date⟩] 345
Claim that the ⟨package⟩ package has been loaded.
\EmulatedPackageWithOptions{⟨optionlist⟩}{⟨package⟩}[⟨date⟩] 345
Claim that the ⟨package⟩ package has been loaded with options ⟨optionlist⟩.
\endMakeFramed Ends the MakeFramed environment. 281
\enlargethispage{⟨length⟩} 52
Increase (or decrease) the text height of the current page by ⟨length⟩.
\begin{enumerate}[⟨style⟩] 139
An ordered list of \items. If ⟨style⟩ is given it overrides the default scheme for indicating the item order.
\epigraph{⟨text⟩}{⟨source⟩} 257
Typesets the ⟨text⟩ and ⟨source⟩ of an epigraph.
\epigraphfontsize{⟨fontsize⟩} Font size to be used for epigraphs. 259
\epigraphforheader[⟨distance⟩]{⟨text⟩} 260
Puts ⟨text⟩ into a zero-sized picture (\epigraphpicture) at the coordinate position (0, -metadistance).

\epigraphhead[⟨distance⟩]{⟨text⟩} ... 259
Stores ⟨text⟩ for printing at ⟨distance⟩ below the page header.
\epigraphpicture ... 260
A zero-sized picture holding the result of \epigraphforheader.
\epigraphposition{⟨flush⟩} ... 258
Sets the horizontal placement of epigraphs.
\epigraphrule ... 259
Thickness of the rule drawn between the text and source of an epigraph.
\begin{epigraphs} Environment for several epigraphs. ... 257
\epigraphsourceposition{⟨flush⟩} ... 258
Sets the placement of the source within an epigraph.
\epigraphtextposition{⟨flush⟩} ... 258
Sets the justification for epigraph text.
\epigraphwidth Textwidth for epigraphs. ... 258
executivepaper Class option for executive-paper stock paper size. ... 2
extrafontsizes Class option for using scalable fonts that can exceed 25pt. ... 3
\extrarowheight ... 230
Length added to the normal row height in tabulars and arrays.
\extratabsurround ... 231
Adds additional space for \firsthline and \lasthline.

\fancybreak{⟨text⟩} ... 100
An anonymous division with ⟨text⟩ centered and the following paragraph is not indented.
\fancybreak*{⟨text⟩} ... 100
Like \fancybreak except that the following paragraph is indented.
\begin{fboxverbatim} ... 287
Puts a frame around the verbatim material. Page breaks are not allowed.
\fcardinal{⟨number⟩} ... 336
Typesets {⟨number⟩} as a cardinal number, with \fnumbersep between each triple of digits.
\feetabovefloat Typeset footnotes above bottom floats (the default). ... 238
\feetbelowfloat Typeset footnotes below bottom floats. ... 238
\figurerefname Name for a figure used by \fref. ... 302
final Class option for final document. ... 5
\firmlist ... 140
In a standard list, sets the vertical spacing intermediate between the default and \tightlist(s).
\firmlists ... 140
Declaration for some vertical spacing in list-based environments. There may be some extra space before and after the environments.
\firmlists* ... 140
The same as \firmlists except that there is no space before and after the environments.
\firsthline ... 231
An \hline (the first) that does not effect vertical alignment of an array or tabular.
\flagverse{⟨flag⟩} ... 270
Used at the start of a verse line to put ⟨flag⟩ distance \vleftskip before the start of the line.
\flegfigure The name for a \legend in a figure. ... 194
\flegfloat{⟨name⟩} ... 194
Where float is a float type (e.g. table), defines the ⟨name⟩ used by \namedlegend.
\flegtable The name for a \legend in a table. ... 194

\flegtocfloat{⟨*title*⟩} 194
Where float is a float type (e.g., figure), called by \namedlegend to add ⟨*title*⟩ to a 'List of...'.
fleqn Class option for fixed indentation of displayed math. 5
\begin{labelled}{⟨*name*⟩}{⟨*width*⟩}{⟨*sep*⟩}{⟨*indent*⟩}{⟨*left*⟩}{⟨*right*⟩} 138
A list of descriptions of \items with the labels formatted according to \name and the overall layout specified by the other list length arguments.
\FloatBlock 182
Force LaTeX to place all unplaced floats before proceeding this point.
\FloatBlockAllowAbove 182
Lessens the restriction by \FloatBlock such that a float inserted *after* a \FloatBlock can appear at the top of the same page as \FloatBlock.
\FloatBlockAllowBelow 182
Lessens the restriction by \FloatBlock such that a float inserted *before* a \FloatBlock can appear at the bottom of the same page as \FloatBlock.
\flushbottom 6
Declaration for last line on a page to be at a constant height.
\begin{flushleft} Text to be typeset flushleft and raggedright. 129
\flushleftright 131
Following this declaration paragraphs are set in their usual form.
\begin{flushright} Text to be typeset flushright and raggedleft. 129
\fnsymbol{⟨*counter*⟩} 243
Typesets the representation of the footnote marker.
\fnumbersep Separator between digit triples in numbers. 336
foolscapvopaper Class option for foolscap octavo stock paper size. 2
\footfootmark Typsets the footnote mark at the bottom of the page. 241
\footfudgefiddle 239
Integer number (default 64) to help when typesetting \paragraphfootnotes.
\footmarksep Offset from the footnote mark box for lines after the first. 241
\footmarkstyle{⟨*style*⟩} 241
Style of the footnote mark at the bottom of the page.
\footmarkwidth Width of footnote mark box. 241
\footnote[⟨*num*⟩]{⟨*text*⟩} Put ⟨*text*⟩ as a footnote. 237
\footnotemark[⟨*num*⟩] Typesets a footnote mark. 237
\footnoterule Rule separating footnotes from the main text. 244
\footnotetext[⟨*num*⟩]{⟨*text*⟩} 237
Typesets ⟨*text*⟩ as a footnote at the bottom of the page but does not put a mark in the main text.
\footparindent Paragraph indent for multiparagraph footnote text. 241
\footreflabstr Reference a labelled footnote. 237
\footruleheight 113
Macro specifying the height of a normal rule above a footer.
\footruleskip 113
Macro specifying a distance sufficient to ensure that a rule above a footer will lie in the space between the footer and the typeblock.
\footskip (length) 21
Distance between the bottom of the typeblock and the bottom of a footer.
\foottextfont Font for footnote text. 242
\foottopagenote 256
Declaration which turns \footnotes into \pagenotes.
\fordinal{⟨*number*⟩} 336
Typesets {⟨*number*⟩} as an ordinal number, with \fnumbersep between each triple of digits.

\frame{⟨contents⟩} A picture object drawing a frame about ⟨contents⟩. 420
\framebox(⟨width,height⟩)[⟨pos⟩]{⟨text⟩} 418
A picture object framing ⟨text⟩ (at position ⟨pos⟩) in a box ⟨width⟩ by ⟨height⟩.
\FrameCommand Draws a 'frame'. 281
\begin{framed} 280
Put a ruled box around the contents of the environment; the box can include pagebreaks.
\FrameHeightAdjust 280
Height of the top of a frame in a framed environment above the baseline at the top of a page.
\framepichead 124
Used by the *showlocs* pagestyle to draw a line at the header location.
\framepichook 124
First thing called inside the zero width pictures provided by \framepichead and \framepictextfoot. Empty by default.
\framepictextfoot 124
Used by the *showlocs* pagestyle to draw a box around the textblock and a line at the footer location.
\FrameRestore Restores settings after a 'frame'. 281
\FrameRule Thickness of the rules around an framed environment. 280
\FrameSep 280
Separation between the surrounding box and text in a framed or shaded environment.
\fref{⟨labstr⟩} 302
Prints a named (\figurerefname) reference to a \labeled figure.
\frontmatter 75
Sets folios to be printed in lowercase roman and prohibits sectional number.
\frontmatter* 75
Same as \frontmatter except that folios are unaltered.
fullpt layout 5
Class option to disable point truncation of certain layout lengths.
\fussy 52
Declaration for TeX to minimise interword space variations in justified text lines.

ger 91
A raggedright, large, bold, two line chapterstyle with rules above and below.
\getarrayelement{⟨arrayname⟩}{⟨index⟩}{⟨result⟩} 340
Sets the parameterless macro ⟨result⟩ to the contents of the {⟨index⟩} location in array ⟨arrayname⟩.
\glossary[⟨file⟩](⟨key⟩){⟨term⟩}{⟨description⟩} 319
Adds ⟨term⟩ and its description, ⟨desc⟩, to a glossary file — \jobname.glo by default or to \file.glo. The optional argument ⟨key⟩ can be used to provide a different sort key for ⟨term⟩.
\glossarycolsep Columns separation in a two column glossary. 322
\glossaryintoc Declaration to add glossary title to the ToC. 322
\glossarymark Redefine to specify marks for headers. 322
\glossaryname Name for a glossary. 322
\glossaryrule Width of inter-column rule in a two column glossary. 322
\glossitem{⟨term⟩}{⟨desc⟩}{⟨ref⟩}{⟨num⟩} 319
Glossary entry used in a theglossary environment

\hangcaption 189
Multiline captions will be typeset as a hanging paragraph.
\hangfrom{⟨stuff⟩} Hangs a paragraph from ⟨stuff⟩. 128

\@hangfrom{⟨code⟩} 97
Kernel macro aiding the setting hanging paragraphs.
hangnum 87
Simple chapterstyle looking like a section head but with the number in the margin.
\hangpara{⟨indent⟩}{⟨num⟩} 128
Apply ⟨indent⟩ for ⟨num⟩ lines to the immediately following paragraph.
\begin{hangparas}{⟨indent⟩}{⟨num⟩} 128
Environment form of \hangpara, applying it to every paragraph in the environment.
\hangsecnum 97
Declaration making sectioning numbers hang in the margin.
\hangsubcaption The subcaption version of \hangcaption. 202
\headdrop (length) 21
Distance between the top of the trimmed page and the top of a header.
\headheight (length) Height available for a header. 21
\headnameref Use header title for sectional title references. 303
\headsep (length) 21
Distance between the bottom of a header and the top of the typeblock.
\headstyles{⟨name⟩} 103
Use the ⟨name⟩ set of headstyles for sectional division heads.
\headwith 112
A (scratch) length normally used in the definition of headers and footers.
\heavyrulewidth Default width for a heavy tabular rule. 222
\hideindexmarks 311
The ⟨stuff⟩ argument to \index and \specialindex will not be printed in the margin (the default).
\hmpunct 330
Punctuation between hours and minutes in \printtime (default :)
\hrulefill Fills a tabular column with a rule. 224

\idtextinnotes{⟨id⟩} Prints an endnote's ⟨id⟩ text 254
\ifdraftdoc true if the draft class option has been used. 325
\ifetex true if etex is the underlying engine, otherwise false. 341
\ifluatex true if LuaTeX is being used to process the document. 342
\ifoddpage Result of \checkoddpage. 334
\ifonlyfloats{⟨yes⟩}{⟨no⟩} 121
Processes ⟨yes⟩ on a page containing only floats, otherwise process ⟨no⟩.
\ifsamename Result from \nametest. 333
\ifsidecapleft 203
true if sidecaptions will be set in the left margin, otherwise they will be set in the right margin.
\IfStreamOpen{⟨stream⟩}{⟨yes⟩}{⟨no⟩} 293
If ⟨stream⟩ is open then the ⟨yes⟩ argument is processed otherwise the ⟨no⟩ argument is processed.
\ifxetex true if XeTeX is being used to process the document. 342
\ignorenoidxfile 309
Do not report attempts to use an idx file that has not been declared by \makeindex.
\iiirdstring Ordinal characters for 'rd', e.g., as in 3rd. 337
\iindstring Ordinal characters for 'nd', e.g., as in 2nd. 337
imperialvopaper Class option for imperial octavo stock paper size. 2
\indentafterchapter 87
Specifies that the first paragraph after a \chapter should be indented

`\indentcaption{`⟨*length*⟩`}` ... 189
Multiline captions will be typeset as a hanging paragraph hung by ⟨*length*⟩.
`\indentpattern{`⟨*digits*⟩`}` ... 269
Stanza lines in `patverse` environment are indented according to the ⟨*digits*⟩ pattern.
`\index[`⟨*file*⟩`]{`⟨*stuff*⟩`}` ... 310
Add ⟨*stuff*⟩ and the current page number to the raw index data. By default this is written to file `\jobname.idx`. If the optional argument is given it will be written to file ⟨*file*⟩`.idx`.
`\indexcolsep` (length) Width of the gutter in two column indexes. ... 309
`\indexintoc` Add the index title to the ToC (the default). ... 309
`\indexmark` Can be used in pagestyles for page headers in an index. ... 309
`\indexmarkstyle` Font style for printing index marks in the margin. ... 311
`\indexname` Name used for the theindex title. ... 309
`\indexrule` (length) ... 309
Width of the inter-column rule in two column indexes.
`\insertchapterspace` ... 87
Called by `\chapter` to insert space into LoF and LoT.
`\isopage[`⟨*spine*⟩`]` ... 32
Generates a page layout especially suited to ISO proportioned paper.
`\iststring` Ordinal characters for 'st', e.g., as in 1st. ... 337
`\item[`⟨*label*⟩`]` ... 137
Intoduces a new element in a list. The effect of ⟨*label*⟩ depends on the particular list form.
`\item` Introduces a main index entry. ... 311
`\begin{itemize}[`⟨*marker*⟩`]` ... 138
An unordered list of `\items`. If given, the ⟨*marker*⟩ overrides the default marker for the elements.
`\itshape` Declaration for using an italic font. ... 42

`\jobname` The name of the document's main source file. ... 292
`\justlastraggedleft` ... 133
Following this declaration paragraphs will be set justified except the last line will be set raggedleft.

`\begin{KeepFromToc}` ... 150
Stop the titles of the enclosed `\listof...` commands from being added to the ToC.
`\keepthetitle` ... 67
Makes most aspects of `\maketitle` unavailable but keeps `\thetitle`, `\theauthor` and `\thedate`.
`\killtitle` Makes all aspects of `\maketitle` unavailable. ... 67
komalike A section-like chapterstyle in a sans serif font. ... 91

`\label{`⟨*labstr*⟩`}` ... 301
Associates the current (section, caption, ...) number, and page number, to ⟨*labstr*⟩.
`\label(`⟨*bookmark*⟩`){`⟨*labstr*⟩`}` ... 200
Associates ⟨*labstr*⟩ with the current (section, caption, etc.) number and page number. If used inside a subfloat and with the hyperref package the optional ⟨*bookmark*⟩ (note the parentheses not square brackets) is available to specify a hyperref bookmark.
`\begin{labelled}{`⟨*name*⟩`}` ... 138
A list of descriptions of `\items` with the labels formatted according to `\name`
landscape ... 1
Class option to interchange height and width of stock paper size.
largecrownvopaper Class option for large crown octavo stock paper size. ... 2

largepostvopaper Class option for large post octavo stock paper size. 2
\lasthline 231
An \hline (the last) that does not effect vertical alignment of an array or tabular.
\lastlineparrule 133
The rule used by \lastlinerulefill to eliminate spaces at the ends of lines.
\lastlinerulefill 133
Ending a paragraph with this will cause any spaces at the ends of the lines will be filled with a rule (\lastlineparrule).
lastpage 328
Counter holding the number of the last page processed during the *previous* LaTeX run.
lastsheet 328
Counter holding the number of sheets processed during the *previous* LaTeX run.
\lcminusname Lowercase 'minus' name, default 'minus'. 338
\leadpagetoclevel 83
Holds the default toc level for \newleadpage generated macros, default is chapter.
\leavespergathering{⟨*num*⟩} 330
Ensure that the correct number of pages are output to make up gatherings of ⟨*num*⟩ leaves each.
ledgerpaper Class option for ledger stock paper size. 2
\begin{leftbar} 281
Draws a thick vertical line in the left margin alongside the contents of the environment.
\leftmark 110
Contains the value of the ⟨*left*⟩ argument of the last \markboth.
\leftspringright{⟨*lfrac*⟩}{⟨*rfrac*⟩}{⟨*ltext*⟩}{⟨*rtext*⟩} 134
Sets ⟨*ltext*⟩ flushleft and raggedright, and ⟨*rtext*⟩ raggedleft and flushright with horizontal space between the two texts.
legalpaper Class option for legal-paper stock paper size. 2
\legend{⟨*text*⟩} A legend (an anonymous caption). 192
leqno Class option for numbering equations at the left. 5
letterpaper Class option for letterpaper stock paper size. 2
\lightrulewidth Default width for a light tabular rule. 222
\line(⟨*dx*,*dy*⟩){⟨*distance*⟩} 423
Picture object of a line, slope ⟨*dx*,*dy*⟩ and coordinate length ⟨*distance*⟩.
\linenottooshort[⟨*length*⟩] 132
Following this declaration the last line of a paragraph will not be shorter than ⟨*length*⟩ (default 2em).
\linenumberfont{⟨*fontspec*⟩} Specify the font for line numbers. 289
\linenumberfrequency{⟨*nth*⟩} 289
Number every ⟨*nth*⟩ line in a boxedverbatim or a verse.
\linethickness{⟨*len*⟩} 416
Picture declaration for vertical and horizontal lines to be ⟨*len*⟩ thick.
\begin{list}{⟨*default-label*⟩}{⟨*code*⟩} 140
The general list environment. ⟨*default-label*⟩ is code that is used for an \item with no ⟨*label*⟩ and ⟨*code*⟩ is used to specify the list layout parameters.
\listfigurename The title for the List of Figures. 151
\listoffigures Typeset the LoF, adding its title to the ToC. 150
\listoffigures* Typeset the LoF. 150
\listoftables Typeset the LoT, adding its title to the ToC. 150
\listoftables* Typeset the LoT. 150
\listtablename The title for the List of Tables. 151
\loosesubcaptions Specifies extra vertical space around subcaptions. 201
\lxvchars (length) Approximate length of 65 characters. 15

lyhne 91
A raggedleft bold sanserif chapter title set between two rules, with the name and number above. It requires the graphicx package.

madsen 91
A raggedleft large bold sanserif chapterstyle with the number in the margin and a rule between the number and title lines. It requires the graphicx package.

`\mainmatter` 75
Sets folio numbers to arabic, starting with 1. Floats, etc., will be numbered per chapter and sectional divisions will be numbered.

`\mainmatter*` Same as `\mainmatter` except that folios are unaltered. 75

`\makebox(`⟨*width,height*⟩`)[`⟨*pos*⟩`]{`⟨*text*⟩`}` 418
A picture object putting ⟨*text*⟩ (at position ⟨*pos*⟩) in a box ⟨*width*⟩ by ⟨*height*⟩.

`\makeevenfoot{`⟨*style*⟩`}{`⟨*left*⟩`}{`⟨*center*⟩`}{`⟨*right*⟩`}` 112
Defines the ⟨*left*⟩, ⟨*center*⟩ and ⟨*right*⟩ parts of the even (verso) page footer of the ⟨*style*⟩ pagestyle.

`\makeevenhead{`⟨*style*⟩`}{`⟨*left*⟩`}{`⟨*center*⟩`}{`⟨*right*⟩`}` 112
Defines the ⟨*left*⟩, ⟨*center*⟩ and ⟨*right*⟩ parts of the even (verso) page header of the ⟨*style*⟩ pagestyle.

`\makefootrule{`⟨*style*⟩`}{`⟨*width*⟩`}{`⟨*thickness*⟩`}{`⟨*skip*⟩`}` 113
Specifies the ⟨*width*⟩ and ⟨*thickness*⟩ of the rule drawn ⟨*skip*⟩ (see `\footskip`) above the footers of the ⟨*style*⟩ pagestyle.

`\MakeFramed{`⟨*settings*⟩`}` 281
The `MakeFramed` environment is the workhorse for the `framed`, `shaded`, etc., environments. The ⟨*settings*⟩ argument controls the final appearance and should include a `\FrameRestore` to reset things back to normal.

`\makeglossary[`⟨*file*⟩`]` 319
Opens file `\jobname.glo`, or `\file.glo`, for glossary entries

`\makeheadfootruleprefix{`⟨*style*⟩`}{`⟨*for headrule*⟩`}{`⟨*for footrule*⟩`}` 113
Can be used to add alternative colors to the head/foot rule

`\makeheadposition{`⟨*style*⟩`}{`⟨*eheadpos*⟩`}{`⟨*oheadpos*⟩`}{`⟨*efootpos*⟩`}{`⟨*ofootpos*⟩`}` 113
Specifies the horizontal positioning of the even and odd headers and footers respectively for the ⟨*style*⟩ pagestyle.

`\makeheadrule{`⟨*style*⟩`}{`⟨*width*⟩`}{`⟨*thickness*⟩`}` 113
Specifies the ⟨*width*⟩ and ⟨*thickness*⟩ of the rule drawn below the headers of the ⟨*style*⟩ pagestyle.

`\makeheadstyles{`⟨*name*⟩`}{`⟨*code*⟩`}` 103
Create a new set of sectional division headstyles called ⟨*name*⟩ defined by ⟨*code*⟩.

`\makeindex[`⟨*file*⟩`]` 308
Preamble command to collect raw index information. By default this is written to file `\jobname.idx`. If the optional argument is used it may be written to file ⟨*file*⟩`.idx`.

`\makeoddfoot{`⟨*style*⟩`}{`⟨*left*⟩`}{`⟨*center*⟩`}{`⟨*right*⟩`}` 112
Defines the ⟨*left*⟩, ⟨*center*⟩ and ⟨*right*⟩ parts of the odd (recto) page footer of the ⟨*style*⟩ pagestyle.

`\makeoddhead{`⟨*style*⟩`}{`⟨*left*⟩`}{`⟨*center*⟩`}{`⟨*right*⟩`}` 112
Defines the ⟨*left*⟩, ⟨*center*⟩ and ⟨*right*⟩ parts of the odd (recto) page header of the ⟨*style*⟩ pagestyle.

`\makepagenote` Preamble command for enabling page/end notes. 251

`\makepagestyle{`⟨*style*⟩`}` Used to define a pagestyle ⟨*style*⟩. 112

`\makepsmarks{`⟨*style*⟩`}{`⟨*code*⟩`}` 113
Hook into the ⟨*style*⟩ pagestyle, usually used for the ⟨*code*⟩ setting any marks.

`\makerunningwidth{`⟨*style*⟩`}[`⟨*footwidth*⟩`]{`⟨*headwidth*⟩`}` 112
Sets the width of the ⟨*style*⟩ pagestyle headers to ⟨*headwidth*⟩. The footers are set to ⟨*headwidth*⟩, or ⟨*footwidth*⟩ if it is given.
`\MakeShortVerb{`⟨*backslash-char*⟩`}` 285
Makes ⟨*char*⟩ a shorthand for `\verb`⟨*char*⟩.
`\MakeTextLowercase{`⟨*text*⟩`}` 54
Lower case ⟨*text*⟩ by leave math, references, citations and `\NoCaseChange{`⟨*text*⟩`}` alone.
`\MakeTextUppercase{`⟨*text*⟩`}` 54
Upper case ⟨*text*⟩ by leave math, references, citations and `\NoCaseChange{`⟨*text*⟩`}` alone.
`\makethanksmark` Typesets the thanks marker and text at the foot. 69
`\makethanksmarkhook` Code hook into `\makethanksmark`. 69
`\maketitlehooka` Hook into `\maketitle` applied before the `\title`. 65
`\maketitlehookb` 65
Hook into `\maketitle` applied between the `\title` and `\author`.
`\maketitlehookc` 65
Hook into `\maketitle` applied between the `\author` and `\date`.
`\maketitlehookd` Hook into `\maketitle` applied after the `\date`. 65
`\marg{`⟨*arg*⟩`}` prints `{`⟨*arg*⟩`}`. 350
`\begin{marginfigure}[`⟨*length*⟩`]` 175
Environment which inserts its contents into the margin, and enables figure captions. The optional argument should be a length and is used to perform manual up/down adjustments to the placement.
`\marginpar[`⟨*left-text*⟩`]{`⟨*text*⟩`}` 244
Puts ⟨*text*⟩ into the margin; if given, ⟨*left-text*⟩ will be used instead of ⟨*text*⟩ for the left margin.
`\marginparmargin{`⟨*placement*⟩`}` 244
Provides a more textual interface for the user to specify in which margin the margin note should be placed.
`\marginparpush` (length) 21
Minimum vertical distance between marginal notes.
`\marginparsep` (length) 21
Horizontal distance between the edge of the typeblock and a marginal note.
`\marginparwidth` (length) Maximum width of marginal note. 21
`\begin{margintable}[`⟨*length*⟩`]` 175
Environment which inserts its contents into the margin, and enables figure captions.The optional argument should be a length and is used to perform manual up/down adjustments to the placement.
`\markboth{`⟨*left*⟩`}{`⟨*right*⟩`}` 110
Sets values of two markers to ⟨*left*⟩ and ⟨*right*⟩ respectively (see `\leftmark` and `\rightmark`).
`\markright{`⟨*right*⟩`}` 110
Sets value of one marker to ⟨*right*⟩ (see `\rightmark`).
`\maxsecnumdepth{`⟨*secname*⟩`}` 79
Sets division numbering level in the `\mainmatter` to ⟨*secname*⟩.
`\maxtocdepth{`⟨*secname*⟩`}` 151
Sets the maximum value of the `tocdepth` counter.
mcrownvopaper Class option for metric crown octavo stock paper size. 1
mdemyvopaper Class option for metric demy octavo stock paper size. 1
`\mdseries` Declaration for using a medium font. 42
`\medievalpage[`⟨*spine*⟩`]` 32
Generates a page layout according to the principles of early printers.
mediumvopaper Class option for medium octavo stock paper size. 2
`\medspace` A medium space (4/18 em). 342

\memappchapinfo{⟨thechapter⟩}{⟨fortoc⟩}{⟨forhead⟩}{⟨title⟩} 348
Code hook into an appendix \chapter
\memappchapstarinfo{⟨fortoc⟩}{⟨title⟩} 348
Code hook into an appendix \chapter*
\memapppageinfo{⟨title⟩} Code hook into \appendixpage. 348
\memapppagestarinfo{⟨title⟩} Code hook into \appendixpage*. 348
\membicaptioninfo{⟨type⟩} {⟨thetype⟩} {⟨fortoc1⟩} {⟨title1⟩} {⟨name2⟩} {⟨title2⟩} 349
Code hook into \membicaption.
\membionenumcaptioninfo{⟨type⟩} {⟨thetype⟩} {⟨fortoc1⟩} {⟨title1⟩} {⟨name2⟩} {⟨fortoc2⟩} {⟨title2⟩} 349
Code hook into \membionenumcaption.
\membitwonumcaptioninfo{⟨type⟩} {⟨thetype⟩} {⟨fortoc1⟩} {⟨title1⟩} {⟨name2⟩} {⟨fortoc2⟩} {⟨title2⟩} 349
Code hook into \membitwonumcaption.
\membookinfo{⟨thebook⟩}{⟨fortoc⟩}{⟨title⟩} Code hook into \book 348
\membookstarinfo{⟨title⟩} Code hook into \book* 348
\memcaptioninfo{⟨type⟩}{⟨thetype⟩}{⟨fortoc⟩}{⟨title⟩} 349
Code hook into \caption
\memchapinfo{⟨thechapter⟩}{⟨fortoc⟩}{⟨forhead⟩}{⟨title⟩} 348
Code hook into \chapter
\memchapstarinfo{⟨fortoc⟩}{⟨title⟩} Code hook into \chapter* 348
\memdskips 51
Adjusts the display skips according to \memdskipstretch.
\memdskipstretch The current factor for increasing display skips. 51
\memendofchapterhook 85
Hook executed at the very end of the \chapter command.
\memfontenc 3
Font encoding for the extrafontsizes class option (default T1)
\memfontfamily 3
Font family for the extrafontsizes class option (default lmr)
\memfontpack 3
Font package for the extrafontsizes class option (default lmodern)
\memglodesc{⟨desc⟩} Wrapper round a glossary description. 321
\memglonum{⟨num⟩} Wrapper round glossary numbers. 321
\memgloref{⟨ref⟩} Wrapper round a glossary ref. 321
\memgloterm{⟨term⟩} Wrapper round a glossary term. 321
\memgobble{⟨text⟩} Gobbles its argument. Do *not* redefine it. 345
\memjustarg{⟨text⟩} Definition is just ⟨text⟩. Do *not* redefine it. 345
\memleadpageinfo{⟨pstyle⟩}{⟨cmdname⟩}{⟨title⟩} 348
Code hook into \newleadpage and \renewleadpage.
\memleadpageinfo{⟨pstyle⟩}{⟨cmdname⟩}{⟨title⟩} 348
Code hook into \newleadpage* and \renewleadpage*.
\memlegendinfo{⟨title⟩} Code hook into \legend 349
\memnamedlegendinfo{⟨fortoc⟩}{⟨title⟩} 349
Code hook into \namedlegend
\memorigdbs Stores the original definition of \\. 131
\memorigpar Stores the original definition of \par. 131
\mempartinfo{⟨thepart⟩}{⟨fortoc⟩}{⟨title⟩} Code hook into \part 348
\mempartstarinfo{⟨title⟩} Code hook into \part* 348
\mempoeminfo{⟨title⟩} Code hook into \poemtitle 349
\mempoemstarinfo{⟨title⟩} Code hook into \poemtitle* 349

\memPoemTitleinfo{⟨thepoem⟩}{⟨fortoc⟩}{⟨forhead⟩}{⟨title⟩} 349
Code hook into \PoemTitle
\memPoemTitlestarinfo{⟨fortoc⟩}{⟨title⟩} 349
Code hook into \PoemTitle*
\mempostaddapppagetotochook 162
Hook executed right *after* \appendixpage writes to the ToC
\mempostaddbooktotochook 162
Hook executed right *after* \book writes to the ToC
\mempostaddchaptertotochook 162
Hook executed right *after* \chapter writes to the ToC
\mempostaddparttotochook 162
Hook executed right *after* \part writes to the ToC
\mempreaddapppagetotochook 162
Hook executed right *before* \appendixpage writes to the ToC
\mempreaddbooktotochook 162
Hook executed right *before* \book writes to the ToC
\mempreaddchaptertotochook 162
Hook executed right *before* \chapter writes to the ToC
\mempreaddparttotochook 162
Hook executed right *before* \part writes to the ToC
\memRTLleftskip RTL (bidi) replacement for \leftskip 352
\memRTLraggedleft RTL (bidi) replacement for \raggedleft 352
\memRTLraggedright RTL (bidi) replacement for \raggedright 352
\memRTLrightskip RTL (bidi) replacement for \rightskip 352
\memRTLvrightskip RTL (bidi) replacement for \vrightskip 352
\memRTLvleftskip RTL (bidi) replacement for \vleftskip 352
\memsecinfo{⟨name⟩}{⟨thename⟩}{⟨fortoc⟩}{⟨forhead⟩}{⟨title⟩} 348
Code hook into the \name section command
\memsecstarinfo{⟨name⟩}{⟨title⟩} 348
Code hook into the \name* section command
\memUChead{⟨text⟩} 114
May uppercase ⟨text⟩, depending on \uppercaseheads and \nouppercaseheads.
\meta{⟨arg⟩} prints ⟨arg⟩. 350
\midbicaption{⟨text⟩} 198
In bilingual captions, ⟨text⟩ is inserted after the first \caption and immediately before the second \caption.
\midbookskip Spacing between a \book's number line and the title. 81
\midchapskip Space between chapter name and number and the title. 86
\midpartskip Spacing between a \part's number line and the title. 81
\midPoemTitleskip 272
Vertical space between the number and text of a poem title.
\midrule[⟨width⟩] 222
Draws a rule across a tabular, default width \lightrulewidth.
\midsloppy 52
Declaration for TeX to allow moderate interword space variations in justified text lines.
\begin{midsloppypar} 52
Typeset contents of the enclosed paragraph(s) using \midsloppy.
\minusname Typeset for 'minus' before negative named numbers. 338
mlargecrownvopaper 1
Class option for metric large crown octavo stock paper size.

\movetoevenpage[⟨text⟩] 335
Stops the current page to start typesetting on the next even page. The optional ⟨text⟩ is put on the skipped page (if there is one).
\movetooddpage[⟨text⟩] 335
Stops the current page to start typesetting on the next odd page. The optional ⟨text⟩ is put on the skipped page (if there is one).
\mpjustification Specialized macro to be used within \marginpars 175
ms Class option for 'typewritten manuscript'. 5
msmallroyalvopaper 1
Class option for metric small royal octavo stock paper size.
\multfootsep Separator between adjacent footnote marks. 238
\multiput(⟨x,y⟩)(⟨dx,dy⟩){⟨num⟩}{⟨object⟩} 429
Drawing command to place ⟨num⟩ copies of ⟨object⟩, starting at coordinates ⟨x,y⟩ and stepping ⟨dx,dy⟩ for each copy after the first.

\namedlegend[⟨short⟩]{⟨long⟩} 194
Like \caption but no number and no 'List of…' entry.
\namedsubappendices 78
Precede sub-appendix numbers with the name \appendixname.
\namenumberand The conjunction in named numbers, default ' and '. 338
\namenumbercomma The 'comma' in named numbers, default ', '. 338
\namerefoff Makes \titleref inoperative. 304
\namerefon Makes \titleref operative. 304
\nametest{⟨str1⟩}{⟨str2⟩} 333
Sets \ifsamename true if ⟨str1⟩ is the same as ⟨str2⟩, otherwise false.
\Needspace{⟨length⟩} 335
Starts a new page, leaving the current page short, unless there is actually at least ⟨length⟩ vertical space left on the current page.
\needspace{⟨length⟩} 335
Starts a new page, leaving the current page short, unless there is estimated ⟨length⟩ vertical space left on the current page.
\Needspace*{⟨length⟩} 335
Starts a new page, leaving the current page short unless \flushbottom is in effect, unless there is actually at least ⟨length⟩ vertical space left on the current page.
\newarray{⟨arrayname⟩}{⟨low⟩}{⟨high⟩} 340
Defines a new indexed array datastructure called ⟨arrayname⟩ with the (integer) index ranging from ⟨low⟩ to ⟨high⟩.
\newblock 307
Used in a bibliography to indicate a convenient place in an entry to have a pagebreak.
\newcolumntype{⟨char⟩}[⟨nargs⟩]{⟨spec⟩} 216
Creates a new column type ⟨char⟩ according to ⟨spec⟩ (which has ⟨nargs⟩ number of arguments).
\newcomment{⟨name⟩} 284
Define a new comment environment called ⟨name⟩.
\newcounter{⟨ctr⟩}[⟨within⟩] 331
Creates a new counter ctr, optionally reset when counter within changes.
\newfixedcaption[⟨capcommand⟩]{⟨command⟩}{⟨float⟩} 195
Defines a captioning command \command that may used outside the ⟨float⟩ float as though it was inside it. The \capcommand must have been previously defined as a captioning command for ⟨float⟩.
\newfloat[⟨within⟩]{⟨fenv⟩}{⟨ext⟩}{⟨capname⟩} 173
Creates new float environments, fenv and fenv*, using counter fenv, which may be restarted

by the ⟨*within*⟩ counter, putting captioning information into the file with extension ⟨*ext*⟩, and using ⟨*capname*⟩ as the name for a caption.
`\newfootnoteseries{`⟨*series*⟩`}` Create a new footnote ⟨*series*⟩. 239
`\newinputstream{`⟨*stream*⟩`}` 292
Creates a new input stream called ⟨*stream*⟩.
`\newleadpage[`⟨*page-style*⟩`]{`⟨*cmdname*⟩`}{`⟨*title*⟩`}` 83
Creates a macro to that when used will generate a sort of title page. Very similar to `\appendixpage`.
`\newlistentry[`⟨*within*⟩`]{`⟨*cntr*⟩`}{`⟨*ext*⟩`}{`⟨*level-1*⟩`}` 168
Creates the commands for typesetting an entry in a 'List of…'. ⟨*cntr*⟩ is the new counter for the entry, which may be reset by the ⟨*within*⟩ counter. ⟨*ext*⟩ is the file extension and ⟨*level-1*⟩ is one less than the entry's level.
`\newlistof{`⟨*listofcom*⟩`}{`⟨*ext*⟩`}{`⟨*listofname*⟩`}` 167
Creates two new List of …commands, `\listofcom` and `\listofcom*`, which use a file with extension ⟨*ext*⟩ and ⟨*listofname*⟩ for the title.
`\newloglike{`⟨*cmd*⟩`}{`⟨*string*⟩`}` 332
Creates a new log-like function command ⟨*cmd*⟩ typesetting ⟨*string*⟩.
`\newloglike{`⟨*cmd*⟩`}{`⟨*string*⟩`}` 332
Creates a new log-like function command ⟨*cmd*⟩ typesetting ⟨*string*⟩, which can take limits.
`\newoutputstream{`⟨*stream*⟩`}` 292
Creates a new output stream called ⟨*stream*⟩.
`\newsubfloat{`⟨*float*⟩`}` Creates subcaptions for use in the ⟨*float*⟩ float. 180
`\nobibintoc` Title of `thebibliography` is not added to the ToC. 305
`\nobookblankpage` Do not follow a book title page with a blank page. 82
`\nobvbox` `boxedverbatim` environments will not be framed in any way. 288
`\NoCaseChange{`⟨*text*⟩`}` 55
The argument of this macro is not touched by `\MakeTextUppercase` or `\MakeTextLowercase`.
`\nochangemarks` Do not print change marks. 325
`\nocite{`⟨*labstr*⟩`}` 307
Add entry ⟨*labstr*⟩ to the bibliography, but with no in-text citation.
`\noDisplayskipStretch` No increased display skips. 51
`\noglossaryintoc` 322
Declaration to prohibit adding glossary title to the ToC.
`\noindentafterchapter` 87
Specifies that the first paragraph after a `\chapter` should *not* be indented
`\noindexintoc` Do not add the index title to the ToC. 309
`\nonzeroparskip` 49
Sets the inter-paragraph spacing to a 'perhaps not too unreasonable' non-zero value.
`\nopartblankpage` Do not follow a part title page with a blank page. 82
`\noprelistbreak` 330
Putting this immediately before an `itemize` (or `enumerate`, or …) environment should prevent a pagebreak.
`\normalbottomsection` 77
Cancels any previous `\raggedbottomsection`.
`\normalcaption` 189
Multiline captions will be typeset as a block paragraph.
`\normalcaptionwidth` Captions will be set to the full width. 189
`\normalfont` 41
Declaration setting the font to the normal body font (upright, Roman, and medium weight).

\normalmarginpar ... 244
Sets the normal margins used by \marginpar(the default).
\normalrulethickness ... 113
The normal thickness of a visible rule (default 0.4pt).
\normalsubcaption The subcaption version of \normalcaption. ... 202
\notedivision ... 253
Heading printed by the \printpagenotes and \printpagenotes* macros.
\noteidinnotes{⟨notenum⟩}{⟨id⟩} ... 254
Prints an endnote's number or id in the endnote listing.
\notenuminnotes{⟨num⟩} ... 254
Typesets the number ⟨num⟩ of a page note in the note listing.
\notenumintext{⟨num⟩} ... 254
Typesets the number ⟨num⟩ of a page note in the main text.
\notepageref ... 253
Declaration that page numbers are available to notes in the endnote listing.
\notesname Name for endnotes, default Notes. ... 253
\nouppercaseheads Defines \memUChead as \relax (i.e., do nothing). ... 114
\nouppercaseheads Do not uppercase the titles in the headings. ... 109
ntglike A smaller version of the standard chapterstyle. ... 92
\nthstring Ordinal characters for 'th', e.g., as in 6th. ... 337
\numberline{⟨num⟩} ... 156
Typeset sectional number for \section and siblings in ToC
\numberlinebox{⟨length⟩}{⟨code⟩} ... 156
The box command used to typeset the sectional number within the ToC, note that it will automatically align to the left
\numberlinehook{⟨num⟩} ... 156
The first thing to be called within \numberline, does nothing by default.
\NumberPoemTitle Declaration for \PoemTitle to be numbered. ... 271
\NumtoName{⟨number⟩} ... 337
Typesets ⟨number⟩ as a cardinal using lowercase words, but uppercase for the initial letter of each word.
\numtoName{⟨number⟩} ... 337
Typesets ⟨number⟩ as a cardinal using lowercase words, but uppercase for the initial letter of the first word.
\numtoname{⟨number⟩} ... 337
Typesets ⟨number⟩ as a cardinal using lowercase words.

\oarg{⟨arg⟩} prints [⟨arg⟩]. ... 350
oldfontcommands ... 5
Class option for permitting obsolete, deprecated font commands.
oldpaper Class option for old stock paper size. ... 2
\begin{onecolabstract} ... 73
Environment for typesetting a one column abstract in a two column document.
\onecolglossary Declaration for a single column glossary. ... 322
\twocolglossary Declaration for a two column glossary. ... 322
\onecolindex Typeset index in one column. ... 309
\onecoltocetc Set the ToC, etc., in one column. ... 151
onecolumn Class option for a single column. ... 4
\begin{OnehalfSpace} Environment form of \OnehalfSpacing ... 51
\begin{OnehalfSpace*} Environment form of \OnehalfSpacing* ... 51

\OnehalfSpacing 50
Declaration increasing the baseline to create the illusion of double spacing.
\OnehalfSpacing* 50
Same as \OnehalfSpacing but also effects page note and floats.
\onelineskip (length) 342
Distance between baselines of the document's main font and size.
oneside Class option for text on only one side of the paper. 4
\openany Allow chapters to start on the next new page. 84
openany Class option for chapters to start on a recto or a verso page. 4
openbib Class option for indenting continuation lines in a bibliography. 5
\openinputfile{⟨filename⟩}{⟨stream⟩} 294
Attaches the file ⟨filename⟩ to the input ⟨stream⟩.
\openleft Force chapters to start on a new verso page. 84
openleft Class option for chapters to start on verso pages. 4
\openoutputfile{⟨filename⟩}{⟨stream⟩} 293
Attaches the file ⟨filename⟩ to the output ⟨stream⟩.
\openright Force chapters to start on a new recto page. 84
openright Class option for chapters to start on recto pages. 4
\ordinal{⟨number⟩} Typesets {⟨number⟩} as an ordinal number. 336
\OrdinaltoName{⟨number⟩} 337
Typeset ⟨number⟩ as an ordinal using lowercase words, but uppercase the initial letter of each word.
\ordinaltoName{⟨number⟩} 337
Typeset ⟨number⟩ as an ordinal using lowercase words, but uppercase the initial letter of the first word.
\ordinaltoname{⟨number⟩} 337
Typeset ⟨number⟩ as an ordinal using lowercase words.
\ordscript{⟨chars⟩} Typesets the ordinal characters ⟨chars⟩. 336
\oval(⟨width,height⟩)[⟨portion⟩] 427
Picture object of a rectangular box, size ⟨width⟩ by ⟨height⟩, with rounded corners. The optional ⟨portion⟩ argument controls whether and which a quarter or a half of the object will be drawn (default is everything).
\overridescapmargin{⟨margin⟩} 204
A one-time override of \sidecapmargin.

\pageinnotes{⟨pagenum⟩} 255
Controls the printing of an endnote's page reference number.
\pagenotesubhead{⟨chapapp⟩}{⟨num⟩}{⟨title⟩} 255
Typesets a subheading for notes from chapter or appendix ⟨chapapp⟩ ⟨num⟩ called ⟨title⟩.
\pagenotesubheadstarred{⟨chapapp⟩}{⟨num⟩}{⟨title⟩} 255
Typesets a subheading for notes from unnumbered chapter or appendix ⟨chapapp⟩ ⟨num⟩ called ⟨title⟩.
\pagenumbering{⟨rep⟩} 107
Resets the page number to 1, and causes the folios (page numbers) to be printed using the ⟨rep⟩ representation (e.g., arabic, roman, ...)
\pagenumbering*{⟨rep⟩} 107
Like \pagenumbering except that the page number is not reset.
\pageref{⟨labstr⟩} 301
Prints the page number associated with ⟨labstr⟩ from a \label.
\pagerefname Name for a page used by \pref. 302

\pagerename 255
Holds the name prefix when referring to a page number, defaults to page.
\pagestyle{⟨style⟩} Sets the current pagestyle to ⟨style⟩. 108
\pagetofootnote 256
Declaration which turns \pagenotes into \footnotes.
\paperheight (length) The height of the trimmed stock paper. 12
\paperwidth (length) The width of the trimmed stock paper. 12
\par Ends a paragraph. 49
\paragraphfootnotes Typeset footnotes as a single paragraph. 239
\paragraphfootstyle{⟨style⟩} 239
Set the ⟨series⟩ footnotes to be typeset in single paragraph style.
\parahook 98
Hook called immediately before typesetting the title of a paragraph head.
\parg{⟨arg⟩} prints (⟨arg⟩). 350
\parindent Indentation at the start of the first line of a paragraph. 50
\parskip Space between paragraphs. 49
\parnopar 246
Forces a new paragraph, but with no apparent break in the text.
\parskip (Extra) vertical space between paragraphs (default 0pt). 127
\part{⟨title⟩} 77
Typesets a numbered part ⟨title⟩ on a page by itself, adding ⟨title⟩ to the ToC.
\partblankpage 82
Follow a part title page with a blank page (the default).
\partmark{⟨title⟩} 82
For setting any marks with the title from a \part for a running header.
\partnamefont Font used by \printpartname for the part name. 81
\partnamenum Called between printing a part name and number. 81
\partnumberline{⟨num⟩} Typeset part number in ToC 156
\partnumberlinebox{⟨length⟩}{⟨code⟩} 156
The box command used to typeset the part number within the ToC, note that it will automatically align to the left
\partnumberlinehook{⟨num⟩} 156
The first thing to be called within \partnumberline, does nothing by default.
\partnumfont Font used by \printpartnum for the part number. 81
\partpageend Code to finish off typesetting a part title page. 82
\partrefname Name for a part used by \Pref. 302
\parttitlefont Font used by \printparttitle for the title. 82
\patchcommand{⟨macro⟩}{⟨start-code⟩}{⟨end-code⟩} 333
Inserts ⟨start-code⟩ before the current definition of the ⟨macro⟩ and {⟨end-code⟩} at the end of the current definition.
\begin{patverse} 269
Stanza lines are indented according to the \indentpattern; lines after the pattern is completed are not indented.
\begin{patverse*} 269
Like patverse except that the pattern will keep repeating.
pedersen 92
A single line chapterstyle in large italics with the number set in the righthand margin. The title and/or number may be colored. The graphicx package is required and the color (or xcolor) package if you want to color.
\pfbreak 101
An alternative for \plainfancybreak using \pfbreakskip and \pfbreakdisplay.

\pfbreak* 101
An alternative for \plainfancybreak* using \pfbreakskip and \pfbreakdisplay.
\pfbreakdisplay{⟨text⟩} ⟨text⟩ for a \pfbreak's \fancybreak. 101
\pfbreakskip Space for a \pfbreak's \plainbreak. 101
\phantomsection 151
A macro to be put before \addcontentsline when the hyperref package is used.
\begin{picture}(⟨width, height⟩) 416
Creates a box of ⟨width⟩ times ⟨height⟩ (in terms of \unitlength) in which you can use drawing commands. The origin is at (0,0).
\begin{picture}(⟨width, height⟩)(⟨llx,lly⟩) 416
Creates a box of ⟨width⟩ times ⟨height⟩ (in terms of \unitlength) in which you can use drawing commands. The origin is at (⟨llx⟩,⟨lly⟩).
\plainbreak{⟨num⟩} 100
An anonymous division of ⟨num⟩ blank lines and the following paragraph is not indented.
\plainbreak*{⟨num⟩} 100
Like \plainbreak except that the following paragraph is indented.
\plainfancybreak{⟨space⟩}{⟨num⟩}{⟨text⟩} 100
An anonymous division that acts like \fancybreak at the top of the page, or at the bottom if there is less than ⟨space⟩ left on the page, otherwise it acts like \plainbreak. The following paragraph is not indented.
\plainfancybreak*{⟨space⟩}{⟨num⟩}{⟨text⟩} 100
Like \plainfancybreak except that the following paragraph is indented.
\plainfootnotes 239
Typeset footnotes as separate marked paragraphs (the default).
\plainfootstyle{⟨series⟩} 239
Set the ⟨series⟩ footnotes to be typeset plain style.
\PlainPoemTitle Declaration for \PoemTitle to be unnumbered. 271
\pmname 330
Abbreviation for post meridiem used in \printtime* (default am)
\pnchap 255
Chapter title for \pagenotesubhead. Defaults to the ToC entry.
\pnschap 255
Starred chapter title for \pagenotesubhead. Defaults to the regular title.
\PoemTitle[⟨fortoc⟩][⟨forhead⟩]{⟨title⟩} 271
Typesets the title for a poem and puts it into the ToC.
\PoemTitle*[⟨fortoc⟩][⟨forhead⟩]{⟨title⟩} 271
Typesets an unnumbered title for a poem but does not add it to the ToC.
\PoemTitlefont Font for the text of a poem title 272
\PoemTitleheadstart Called at the start of typesetting a \PoemTitle. 271
\poemtitlemark{⟨forhead⟩} Used to set marks for a \PoemTitle. 271
\PoemTitlenumfont Font for the number of a poem title 272
\poemtitlepstyle Page style for a \PoemTitle. 271
\poemtitlestarmark{⟨forhead⟩} 271
Used to set marks for a \PoemTitle*.
\poemtitlestarpstyle Page style for a \PoemTitle*. 271
\poemtoc Kind of entry for a \PoemTitle in the ToC. 271
\postdate{⟨text⟩} 64
Command processed after the \date in \maketitle.
\postauthor{⟨text⟩} 64
Command processed after the \author in \maketitle.

\postbibhook 306
Called after typesetting the list of of bibliographic entries.
\postcaption{⟨*posttext*⟩} 189
⟨*posttext*⟩ will be processed at the end of a caption.
\postnoteinnotes 254
Called by \noteentry to finish the printing of an endnote.
\postnotetext 255
Called within a page note list entry to finish the printing of an endnote.
\postitle{⟨*text*⟩} 64
Command processed after the \title in \maketitle.
postvopaper Class option for post octavo stock paper size. 2
pottvopaper Class option for pott octavo stock paper size. 2
\preauthor{⟨*text*⟩} 64
Command processed before the \author in \maketitle.
\prebibhook 306
Called between typesetting the title of a bibliography and starting the list of bibliographic entries.
\precaption{⟨*pretext*⟩} 189
⟨*pretext*⟩ will be processed at the start of a caption.
\prechapterprecisshift 93
Spacing between a chapter head and a chapter precis.
\precistocfont Font for typesetting \precistoctext. 93
\precistocformat Format for typesetting \precistoctext. 93
\precistoctext{⟨*text*⟩} ToC entry for chapter precis ⟨*text*⟩ 93
\predate{⟨*text*⟩} 64
Command processed before the \date in \maketitle.
\Pref{⟨*labstr*⟩} 302
Prints a named (\partrefname) reference to a \labeled part.
\pref{⟨*labstr*⟩} 302
Prints a named (\pagerefname) reference to a \label page reference.
\preglossaryhook 322
Vacuous macro called after a glossary title is typeset.
\preindexhook 309
Called between typesetting an index's title and the start of the list.
\prenoteinnotes 254
Called by \noteentry to initialise the printing of an endnote.
\prenotetext 255
Called within a page note list entry to initialise the printing of the text part of an endnote.
\pretitle{⟨*text*⟩} 64
Command processed before the \title in \maketitle.
\printbookname 81
Prints the book name (\bookname) using the \booknamefont.
\printbooknum Prints a book number using the \booknumfont. 81
\printbooktitle Prints the book title using the \booktitlefont. 82
\printchaptername 86
Prints the chapter name using the \chapnamefont.
\printchapternonum 86
Replaces printing the chapter name and number in unnumbered chapters.
\printchapternum 86
Prints the chapter number using the \chapnumfont.

\printchaptertitle{⟨title⟩} 86
Prints the chapter ⟨title⟩ using the \chaptitlefont.
\printglossary[⟨file⟩] 319
Prints the glossary from file \jobname.gls, or \file.gls
\printindex[⟨file⟩] 308
Print the sorted index. By default this is read from file \jobname.ind. If the optional argument is given it will read the data from file ⟨file⟩.ind.
\printpageinnotes{⟨pagenum⟩} 255
Prints an endnote's page reference number.
\printpageinnoteshyperref{⟨pagenum⟩} 255
Prints an endnote's page reference number whenever the hyperref package is used, it will include a hyperlink back to the page in question.
\printpagenotes 251
Input the pagenote ent file for printing, then close it to any more notes.
\printpagenotes* 251
Input the pagenote ent file for printing, then empty it ready for further notes.
\printpartname 81
Prints the part name (\partname) using the \partnamefont.
\printpartnum Prints a part number using the \partnumfont. 81
\printparttitle Prints the part title using the \parttitlefont. 82
\printPoemTitlenonum 271
Used instead of \printPoemTitlenum for an unnumbered \PoemTitle.
\printPoemTitlenum Typesets the number for a \PoemTitle. 271
\printPoemTitletitle{⟨title⟩} Typesets the title of a \PoemTitle. 271
\printtime Prints the time of day using a 24 hour clock. 330
\printtime* Prints the time of day using a 12 hour clock. 330
\printXtitle{⟨title⟩} 152
Generic macro printing ⟨title⟩ as the title for the 'X List of'.
\providecounter{⟨ctr⟩}[⟨within⟩] 332
A 'provide' version of \newcounter.
\provideenvironment{⟨name⟩}[⟨numarks⟩][⟨optarg⟩]{⟨begindef⟩}{⟨enddef⟩} 332
A 'provide' version of \(re)newenvironment.
\providefixedcaption[⟨capcommand⟩]{⟨command⟩}{⟨float⟩} 195
A 'provide' version of \newfixedcaption.
\providelength{⟨cmd⟩} A 'provide' version of \newlength. 332
\provideloglike{⟨cmd⟩}{⟨string⟩} 332
A 'provide' version of \newloglike.
\provideloglike*{⟨cmd⟩}{⟨string⟩} 332
A 'provide' version of \newloglike*.
\put(⟨x,y⟩){⟨object⟩} 417
Drawing command to place ⟨object⟩ at coordinates (⟨x,y⟩). In the simplest case ⟨object⟩ is just text.

\qbezier[⟨num⟩](⟨Xs,Ys⟩)(⟨Xm,Ym⟩)(⟨Xe,Ye⟩) 431
Picture command to draw a quadratic Bezier curve from ⟨Xs,Ys⟩ to ⟨Xe,Ye⟩ passing near ⟨Xm,Ym⟩. If the optional argument is present exactly ⟨num⟩ segments will be used in drawing the curve.
\qbeziermax 433
The maximum number of segments for drawing a Bezier curve.
\qitem{⟨text⟩}{⟨source⟩} 257
Typesets the ⟨text⟩ and ⟨source⟩ of an epigraph in an epigrpahs environment.

\quarkmarks 328
Trim marks in the style of Quark Xpress registration marks, typeset with \registrationColour.
\begin{quotation} 130
Contents set justified in a narrower measure with normal \parindent.
\begin{quote} 130
Contents set justified in a narrower measure, with zero \parindent.

\raggedbottom 6
Declaration allowing the last line on a page to be at a variable height.
\raggedbottomsection 77
Pages will be typeset short because of a moved subhead as if \raggedbottom was in effect.
\raggedleft Declaration for text to be set raggedleft and flushright. 129
\raggedright Declaration for text to be set flushleft and raggedright. 129
\leftcenterright 133
Following this declaration paragraphs will be set with the first line raggedright, the last raggedleft, and those in the middle centered.
\raggedrightthenleft 133
Following this declaration paragraphs will be set with the first line raggedright and the rest raggedleft.
\raggedyright[⟨space⟩] 130
Version of \raggedright with ⟨space⟩ providing control over the amount of raggedness.
\ragrparindent The \parindent for \raggedyright paragraphs. 130
\readaline{⟨stream⟩} 294
Reads a single line from the file associated with the input ⟨stream⟩.
\readboxedverbatim{⟨stream⟩} 294
Acts like boxedverbatim except the contents is read from the file associated with the input ⟨stream⟩.
\readboxedverbatim*{⟨stream⟩} 294
Acts like boxedverbatim* except the contents is read from the file associated with the input ⟨stream⟩.
\readstream{⟨stream⟩} 294
Reads the entire contents of the file associated with the input ⟨stream⟩.
\readverbatim{⟨stream⟩} 294
Acts like verbatim except the contents is read from the file associated with the input ⟨stream⟩.
\readverbatim*{⟨stream⟩} 294
Acts like verbatim* except the contents is read from the file associated with the input ⟨stream⟩.
\ref{⟨labstr⟩} 301
Prints the (section, or other) number associated with ⟨labstr⟩ from a \label.
\refixpagelayout 26
Command to reemit the document size to the output.
\registrationColour{⟨mark⟩} Typesets \quarkmarks. 328
\renewfixedcaption[⟨capcommand⟩]{⟨command⟩}{⟨float⟩} 195
A 'renew' version of \newfixedcaption.
\renewleadpage[⟨page-style⟩]{⟨cmdname⟩}{⟨title⟩} 83
Redefines an existing new lead page macro.
reparticle 88
With the article class option it replicates a section head in the article class.
\reportnoidxfile 309
Report attempts to use an idx file that has not been declared by \makeindex.

\RequireAtEndClass{⟨class⟩}{⟨code⟩} 347
Inserts ⟨code⟩ just after the ⟨class⟩ class is used, or immediately if ⟨class⟩ has already been used.
\RequireAtEndPackage{⟨pack⟩}{⟨code⟩} 347
Inserts ⟨code⟩ just after the ⟨pack⟩ package is used, or immediately if ⟨pack⟩ has already been used.
\RequireXeTeX 342
Generates an error if the document is not being processed by XeTeX.
\resetbvlinenumber Resets the boxedverbatim line number to zero. 289
\restorepagenumber 108
Sets the page number to that saved by the most recent \savepagenumber.
\restoretrivseps 143
Sets the current \topsep and \partopsep to the values saved by \savetrivseps.
\reversemarginpar 244
Reverses the normal margins used by \marginpar.
\rightmark 110
Contains the value of the ⟨right⟩ argument of the first \markboth or \markright on the page; if there is none then the value of the most recent ⟨right⟩ argument.
\rmfamily Declaration for using a Roman font. 42
royalvopaper Class option for royal octavo stock paper size. 2
\russianpar 132
Ending a paragraph with \russianpar causes it to be set following Russian typographic rules.

\savebox⟨box⟩(⟨width, height⟩)[⟨pos⟩]{⟨text⟩} 422
Picture command to save ⟨text⟩ in a (pre-existing) storage box ⟨box⟩ making it size ⟨width⟩ times ⟨height⟩. The optional argument controls the position of the ⟨text⟩.
\savepagenumber Saves the current page number. 108
\savetrivseps 143
Stores the current \topsep and \partopsep for trivlists.
\saythanks 73
Following a onecolabstract it ensures that \thanks are printed.
\scshape Declaration for using a small caps font. 42
\@seccntformat{⟨code⟩} 97
Kernel macro that formats the number in a sectional head.
\sechook 98
Hook called immediately before typesetting the title of a section head.
\section[⟨toc-title⟩][⟨head-title⟩]{⟨title⟩} 76
Typesets a section subhead ⟨title⟩, adding ⟨title⟩ to the ToC and possibly the running headers. If given ⟨toc-title⟩ is used instead of ⟨title⟩ for the ToC and running header. If given ⟨head-title⟩ is used for a running header.
\section*{⟨title⟩} 76
Typesets an unnumbered section subhead ⟨title⟩. There are no ToC or running header entries.
section Simple chapterstyle looking like a section head. 87
\sectionrefname Name for a section used by \Sref. 302
\see *see* entry in an index using \seename for the wording. 314
\seealso *see also* entry in an index using \alsoname for the wording. 314
\seename Wording for a *see* index entry. 314
\semiisopage[⟨spine⟩] 32
Generates a page layout especially suited to ISO proportioned paper but with smaller margins than \isopage.
\setafterparaskip{⟨skip⟩} 96
Sets the \afterskip for a paragraph head.

\setaftersecskip{⟨skip⟩} Sets the \afterskip for a section head. 96
\setafterSskip{⟨skip⟩} Sets the \afterskip for an S head. 96
\setaftersubparaskip{⟨skip⟩} 96
Sets the \afterskip for a subparagraph head.
\setaftersubsecskip{⟨skip⟩} 96
Sets the \afterskip for a subsection head.
\setaftersubsubsecskip{⟨skip⟩} 96
Sets the \afterskip for a subsubsection head.
\setarrayelement{⟨arrayname⟩}{⟨index⟩}{⟨text⟩} 340
Makes ⟨text⟩ the contents of the {⟨index⟩} location in array ⟨arrayname⟩.
\setbeforeparaskip{⟨skip⟩} 94
Sets the \beforeskip for a \paragraph head.
\setbeforesecskip{⟨skip⟩} 94
Sets the \beforeskip for a \section head.
\setbeforeSskip{⟨skip⟩} Sets the \beforeskip for an S head. 94
\setbeforesubparaskip{⟨skip⟩} 94
Sets the \beforeskip for a \subparagraph head.
\setbeforesubsecskip{⟨skip⟩} 94
Sets the \beforeskip for a \subsection head.
\setbeforesubsubsecskip{⟨skip⟩} 94
Sets the \beforeskip for a \subsubsection head.
\setbiblabel{⟨style⟩} 307
Define the look of the bibliographic entry identifiers.
\setbinding{⟨length⟩} Adds a binding allowance to the spine margin. 19
\setbvlinenums{⟨first⟩}{⟨startat⟩} 289
The first line of the following boxedverbatim is number {⟨first⟩} and the first printed line number should be ⟨startat⟩.
\setcolsepandrule{⟨colsep⟩}{⟨thickness⟩} 22
Sets the width of the gutter and the thickness of the rule in the gutter.
\setDisplayskipStretch{⟨factor⟩} 51
Increase the display skips by gmetafactor.
\setfloatadjustment{⟨floatname⟩}{⟨code⟩} 174
Add global internal adjustment to a given type of float. Empty by default, but can easily be used to make all floats centered or all tables in a smaller font size.
\setFloatBLockFor{⟨sectional name⟩} 182
Adds \FloatBlock within the \⟨section name⟩ macro.
\setfloatlocations{⟨float⟩}{⟨locs⟩} 181
Sets the default location for the ⟨float⟩ (e.g., table) to ⟨locs⟩ (default tbp).
\setFloatSpacing Explicitly set the spacing used inside floats. 51
\setfootins{⟨length for normal⟩}{⟨length for minipage⟩} 244
Sets \skip\footins and its minipage counterpart.
\sethangfrom{⟨code⟩} User macro redefining \@hangfrom to ⟨code⟩. 97
\setheaderspaces{⟨headdrop⟩}{⟨headsep⟩}{⟨ratio⟩} 21
Sets the spacing above and below a header.
\setheadfoot{⟨headheight⟩}{⟨footskip⟩} 21
Sets the headheight and footskip.
\setlrmargins{⟨spine⟩}{⟨edge⟩}{⟨ratio⟩} 17
Sets the spine and fore-edge margins for the current typeblock width.
\setlrmarginsandblock{⟨spine⟩}{⟨edge⟩}{⟨ratio⟩} 18
Sets the spine and fore-edge margins, modifying the typeblock width to match.

\setlxvchars[⟨fontspec⟩] 15
Sets cslxvchars to the length of 65 characters in the ⟨fontspec⟩ font (default \normalfont).
\setmarginnotes{⟨separation⟩}{⟨width⟩}{⟨push⟩} 21
Sets the length parameters for marginal notes.
\setmpjustification{⟨at left of textblock⟩}{⟨at right of textblock⟩} 175
Loads the \mpjustification command to execute the left part when placed at the left of the text block and vice versa.
\setpagebl{⟨height⟩}{⟨width⟩}{⟨ratio⟩} 33
Specifies a page of the given dimensions positioned at the bottom left of the stock; see \settrimmedsize.
\setpageml{⟨height⟩}{⟨width⟩}{⟨ratio⟩} 33
Specifies a page of the given dimensions positioned at the middle left of the stock; see \settrimmedsize.
\setpagetl{⟨height⟩}{⟨width⟩}{⟨ratio⟩} 33
Specifies a page of the given dimensions positioned at the top left of the stock; see \settrimmedsize.
\setpagebm{⟨height⟩}{⟨width⟩}{⟨ratio⟩} 33
Specifies a page of the given dimensions positioned at the bottom middle of the stock; see \settrimmedsize.
\setpagebr{⟨height⟩}{⟨width⟩}{⟨ratio⟩} 33
Specifies a page of the given dimensions positioned at the bottom right of the stock; see \settrimmedsize.
\setpagecc{⟨height⟩}{⟨width⟩}{⟨ratio⟩} 33
Specifies a page of the given dimensions positioned at the center of the stock; see \settrimmedsize.
\setpagemr{⟨height⟩}{⟨width⟩}{⟨ratio⟩} 33
Specifies a page of the given dimensions positioned at the middle right of the stock; see \settrimmedsize.
\setPagenoteSpacing 51
Explicitly set the spacing used inside page notes such including footnotes.
\setpagetm{⟨height⟩}{⟨width⟩}{⟨ratio⟩} 33
Specifies a page of the given dimensions positioned at the top middle of the stock; see \settrimmedsize.
\setpagetr{⟨height⟩}{⟨width⟩}{⟨ratio⟩} 33
Specifies a page of the given dimensions positioned at the top right of the stock; see \settrimmedsize.
\setparaheadstyle{⟨font⟩} Sets the style (font) for a paragraph head. 96
\setparahook{⟨text⟩} Redefines \parahook to be ⟨text⟩. 98
\setparaindent{⟨length⟩} Sets the \indent for a \paragraph head. 96
\setpnumwidth{⟨length⟩} 153
Sets the width of the page number box (\@pnumwidth) in a 'List of…' to ⟨length⟩.
\setrmarg{⟨length⟩} 153
Sets the right hand title margin (\@tocrmarg) in a 'List of…' to ⟨length⟩.
\setsecheadstyle{⟨font⟩} Sets the style (font) for a section head. 96
\setsechook{⟨text⟩} Redefines \sechook to be ⟨text⟩. 98
\setsecindent{⟨length⟩} Sets the \indent for a \section head. 96
\setsecnumdepth{⟨secname⟩} 79
Sets division numbering level to ⟨secname⟩.
\setsecnumformat{⟨code⟩} Redefines \@seccntformat to ⟨code⟩. 97
\setSheadstyle{⟨font⟩} Sets the style (font) for an S head. 96
\setShook{⟨text⟩} Redefines \Shook to be ⟨text⟩. 98

\setsidcappos{⟨pos⟩} ... 203
Declaration of the vertical position of a sidecaption with respect to the float.
\setsidebarheight{⟨height⟩} ... 247
Sets the height of sidebars. The default is \textheight.
\setsidebars{⟨hsep⟩}{⟨width⟩}{⟨vsep⟩}{⟨topsep⟩}{⟨font⟩}{⟨height⟩} ... 248
Sets the several sidebar parameters.
\setsidecaps{⟨sep⟩}{⟨width⟩} ... 202
Sets the lengths \sidecapsep and \sidcapwidth to the given values.
\setSindent{⟨length⟩} Sets the \indent for an S head. ... 96
\setSingleSpace{⟨factor⟩} Change the baselineskip by ⟨factor⟩. ... 51
\setstocksize{⟨height⟩}{⟨width⟩} ... 8
Set the stock paper size to be ⟨height⟩ by ⟨width⟩.
\setsubparaheadstyle{⟨font⟩} ... 96
Sets the style (font) for a subparagraph head.
\setsubparahook{⟨text⟩} Redefines \subparahook to be ⟨text⟩. ... 98
\setsubparaindent{⟨length⟩} ... 96
Sets the \indent for a \subparagraph head.
\setsubsecheadstyle{⟨font⟩} ... 96
Sets the style (font) for a subsection head.
\setsubsechook{⟨text⟩} Redefines \subsechook to be ⟨text⟩. ... 98
\setsubsecindent{⟨length⟩} ... 96
Sets the \indent for a \subsection head.
\setsubsubsecheadstyle{⟨font⟩} ... 96
Sets the style (font) for a subsubsection head.
\setsubsubsechook{⟨text⟩} Redefines \subsubsechook to be ⟨text⟩. ... 98
\setsubsubsecindent{⟨length⟩} ... 96
Sets the \indent for a \subsubsection head.
\settocdepth{⟨secname⟩} ... 151
Sets the value of the tocdepth counter in the toc file.
\settocpreprocessor{⟨type⟩}{⟨code⟩} ... 162
Provide a method for preprocessing certain TOC entries before they are written to the .toc file.
\settrimmedsize{⟨height⟩}{⟨width⟩}{⟨ratio⟩} ... 12
Sets the size of the trimmed stock paper.
\settrims{⟨top⟩}{⟨foredge⟩} ... 13
Sets the amount to be trimmed from the top and fore-edge of the stock paper.
\settypeblocksize{⟨height⟩}{⟨width⟩}{⟨ratio⟩} ... 17
Sets the size of the typeblock.
\settypeoutlayoutunit ... 26
Allows the user to set the unit in which to typeset the layout listing. Supported units: pt, mm, cm, in, bp, dd and cc.
\setulmargins{⟨upper⟩}{⟨lower⟩}{⟨ratio⟩} ... 19
Sets the upper and lower margins with the current typeblock height.
\setulmarginsandblock{⟨upper⟩}{⟨lower⟩}{⟨ratio⟩} ... 20
Sets the upper and lower margins, modifying the typeblock height to match.
\setverbatimfont{⟨fontspec⟩} ... 286
Sets the font to be used for verbatim text.
\setverselinenums{⟨first⟩}{⟨startat⟩} ... 270
The first line of the following verse is number {⟨first⟩} and the first printed line number should be ⟨startat⟩.

\setxlvchars[⟨fontspec⟩] 15
Sets \xlvchars to the length of 45 characters in the ⟨fontspec⟩ font (default \normalfont).
\sffamily Declaration for using a Sans serif font. 42
\begin{shaded} 280
Put a colored background behind the contents of the environment, which can include page-breaks. The color extends into the margins a little.
sheetsequence Counter for sheets (similar to page for pages). 328
\Shook 98
Hook called immediately before typesetting the title of an S head.
\shortstack[⟨pos⟩]{⟨text⟩} 421
Vertically stacks each line of ⟨text⟩ into a column, normally centered but can be left or right aligned via ⟨pos⟩. Usually used as a picture object, but can be used outside the environment.
\shortsubcaption The subcaption version of \shortcaption. 202
\showcols 217
Writes a list of all \newcolumntypes to the terminal and log file.
\showheadfootlocoff 124
Prevents \framepichead and \framepictextfoot from drawing lines at the header and footer locations.
\showindexmarks 311
The ⟨stuff⟩ argument to \index and \specialindex will be printed in the margin (for use in noting what has been indexed where).
\showtextblockoff 124
Prevents \framepictextfoot from drawing a box around the textblock.
showtrims Class option for printing trimming marks. 5
\showtrimsoff Switch off any trim marks. 326
\showtrimson 326
If the showtrims option has been used, switch on any trim marks (this is the default).
\sidebar{⟨text⟩} Typesets ⟨text⟩ in a sidebar. 247
\sidebarfont Font used for sidebars. 247
\sidebarform Form (e.g., \raggedright) used for sidebars. 247
\sidebarhsep Space between the edge of the main text and sidebars. 247
\sidebarmargin{⟨margin⟩} Specifies the ⟨margin⟩(s) for sidebars. 247
\sidebartopsep 247
Controls the vertical position of a sidebar. The default is 0pt which aligns the tops of the typeblock and the sidebar.
\sidebarvsep 247
Vertical space between sidebars that fall on the same page.
\sidebarwidth Width of sidebars. 247
\sidecapfloatwidth{⟨length⟩} 204
Macro holding the width of a float with a sidecaption.
\sidecapmargin{⟨margin⟩} Sets the margin for sidecaptions. 203
\sidecapraise 204
Vertical distance added to the default vertical placement of a sidecaption.
\sidecapsep 202
Length specifying the horizontal separation between a sidecaption and the float.
\sidecapstyle Style settings for a sidecaption. 204
\begin{sidecaption}[⟨fortoc⟩]{⟨title⟩}[⟨label⟩] 202
Environment for setting a sidecaption.
\sidecapwidth 202
Length specifying the maximum width of a sidecaption.

\begin{sidecontcaption}{⟨title⟩}[⟨label⟩] ... 203
Environment for setting a continued sidecaption.
\begin{sidelegend}{⟨title⟩}[⟨label⟩] ... 203
Environment for setting a legend kind of sidecaption.
\begin{sidenamedlegend}{⟨title⟩}[⟨label⟩] ... 203
Environment for setting a named legend kind of sidecaption.
\sidepar[⟨left⟩]{⟨right⟩} ... 245
Like \marginpar except that the note does not move vertically.
\siderparfont The font which the \sidepars are typeset. ... 245
\sideparform Macro holding placement like \raggedright and such. ... 245
\sideparmargin{⟨placement⟩} ... 245
Specify into which margin the \sidepar goes.
\sideparvshift Move a \sidepar up/down by this amount. ... 245
\begin{SingleSpace} Environment form of \SingleSpacing ... 51
\SingleSpacing ... 51
Declaration restoring normal single spacing (or that set by \setSingleSpace). Note, this added a skip at the end.
\SingleSpacing* ... 51
Same as \SingleSpacing, but does not add a skip at the end.
\slashfrac{⟨top⟩}{⟨bottom⟩} ... 339
Typesets like 3/4, using the \slashfracstyle font.
\slashfracstyle{⟨num⟩} ... 339
Typesets ⟨num⟩ in a particular (font, size) style.
\sloppy ... 52
Declaration for TeX to allow large interword space variations in justified text lines.
\sloppybottom ... 53
Declaration for TeX to allow an extra line at the bottom of a page. The \topskip must have been increased beforehand.
\begin{sloppypar} ... 52
Typeset contents of the enclosed paragraph(s) using \sloppy.
\slshape Declaration for using a slanted font. ... 42
smalldemyvopaper Class option for small demy octavo stock paper size. ... 2
smallroyalvopaper Class option for small royal octavo stock paper size. ... 2
\begin{snugshade} Like shaded but does not bleed into the margins. ... 280
\sourceatright[⟨length⟩]{⟨text⟩} ... 130
At the end of a paragraph puts ⟨text⟩ at the end of the line if the line is short enough for a space ⟨length⟩ and the ⟨text⟩, otherwise puts ⟨text⟩ flushright on the following line.
southall ... 92
A raggedright chapterstyle with the number and title on the same line and a rule below.
\begin{SingleSpace}{⟨factor⟩} ... 51
Environment form of \setSingleSpace
\specialindex{⟨file⟩}{⟨counter⟩}{⟨stuff⟩} ... 311
Add ⟨stuff⟩ and the current value of ⟨counter⟩ to the raw index data file ⟨file⟩.idx.
\specialrule{⟨width⟩}{⟨abovespace⟩}{⟨belowspace⟩} ... 223
Draws a rule with the given parameters across a tabular.
\Sref{⟨labstr⟩} ... 302
Prints a named (\sectionrefname) reference to a \labeled section.
\stanzaskip Vertical space between verse stanzas. ... 266
statementpaper Class option for statement stock paper size. ... 2
\stockheight (length) Height of the stock paper. ... 8
\stockwidth (length) Width of the stock paper. ... 8

\strictpagecheck 334
\checkoddpage will use an accurate but time and space consuming method for checking for an odd page number.
\stringtoarray{⟨arrayname⟩}{⟨string⟩} 340
Puts each character from ⟨string⟩ sequentially into array ⟨arrayname⟩, starting at index 1.
\begin{subappendices} 78
Like the appendices environment but used at the end of a chapter for per-chapter sub-appendices.
\subbottom[⟨list-entry⟩][⟨subtitle⟩]{⟨text⟩} 199
Puts a subcaption identifier, and optionally ⟨subtitle⟩, below ⟨text⟩.
\subcaption[⟨list-entry⟩]{⟨subtitle⟩} 199
Analagous to \caption but for an identified subcaption within a float.
\subcaptionfont{⟨fontspec⟩} Font for subcaption titles. 201
\subcaptionlabelfont{⟨fontspec⟩} Font for subcaption identifiers. 201
\subcaptionref{⟨labstr⟩} 200
Print the subcaption identifer for a ⟨labstr⟩ labelled subcaption.
\subcaptionsize{⟨size⟩} Font size for subcaptions. 201
\subcaptionstyle{⟨style⟩} Paragraph ⟨style⟩ for subcaptions. 202
\subconcluded 200
Indicates (to LaTeX) that a continued subfloat is finished.
\subitem Introduces a subsidiary index entry. 311
\subparahook 98
Hook called immediately before typesetting the title of a subparagraph head.
\subsechook 98
Hook called immediately before typesetting the title of a subsection head.
\subsubitem Introduces a third level index entry. 311
\subsubsechook 98
Hook called immediately before typesetting the title of a subsubsection head.
\subtop[⟨list-entry⟩][⟨subtitle⟩]{⟨text⟩} 199
Puts a subcaption identifier, and optionally ⟨subtitle⟩, on top of ⟨text⟩.
superroyalvopaper Class option for super royal octavo stock paper size. 2
\suppressfloats[⟨pos⟩] 182
Suppresses any floats on the current page at the given ⟨pos⟩ placement.
\symbolthanksmark 67
Set the thanks marks to be printed using the footnote series of symbols.

\tabcolsep Half the space between columns in a tabular. 230
\tableofcontents Typeset the ToC, adding its title to the ToC itself. 150
\tableofcontents* Typeset the ToC. 150
\tablerefname Name for a table used by \tref. 302
\tabsoff Ignore extra TAB spaces in a verbatim. 286
\tabson[⟨number⟩] 286
Set ⟨number⟩ of spaces in a verbatim for a TAB character; default 4.
\begin{tabular}[⟨pos⟩]{⟨preamble⟩} 211
Environment for setting text elements in a tabular form.
\begin{tabular*}{⟨width⟩}[⟨pos⟩]{⟨preamble⟩} 211
Environment for setting text elements in a tabular form within an overall ⟨width⟩; intercolumn spacing is adjusted to suit.
\begin{tabularx}{⟨width⟩}[⟨pos⟩]{⟨preamble⟩} 211
Environment for setting text elements in a tabular form within an overall ⟨width⟩; column widths are adjusted to suit.

\tabularxcolumn Column type for an X column in a tabularx. 228
tandh A simple section-like chapterstyle in a bold font. 92
\tensunitsep 338
The separator/conjoiner between tens and units in named numbers, default '-'.
\textbf{⟨*text*⟩} Typeset ⟨*text*⟩ with a bold font. 42
\textheight (length) The height of the typeblock. 17
\textit{⟨*text*⟩} Typeset ⟨*text*⟩ with an italic font. 42
\textmd{⟨*text*⟩} Typeset ⟨*text*⟩ with a medium font. 42
\textrm{⟨*text*⟩} Typeset ⟨*text*⟩ with a Roman font. 42
\textsc{⟨*text*⟩} Typeset ⟨*text*⟩ with a small caps font. 42
\textsf{⟨*text*⟩} Typeset ⟨*text*⟩ with a Sans serif font. 42
\textsl{⟨*text*⟩} Typeset ⟨*text*⟩ with a slanted (oblique) font. 42
\textsubscript{⟨*sub*⟩} Typesets ⟨*sub*⟩ as a subscript. 339
^{⟨*super*⟩} Typesets ⟨*super*⟩ as a superscript. 339
\texttt{⟨*text*⟩} Typeset ⟨*text*⟩ with a Typewriter (monospaced) font. 42
\textup{⟨*text*⟩} Typeset ⟨*text*⟩ with an upright font. 42
\textwidth (length) The width of the typeblock. 17
\thanksfootmark Handle the inner part of the thanks mark at the foot. 69
\thanksfootmark Typesets a thanks mark at the foot. 69
\thanksheadextra{⟨*pre*⟩}{⟨*post*⟩} 68
Inserts ⟨*pre*⟩ and ⟨*post*⟩ before and after thanks markers in the titling code.
\thanksmark{⟨*n*⟩} 68
Prints a thanks mark identical to the n'th (previously) printed mark.
\thanksmarksep 69
Indentation of second and subsequent thanks text lines at the foot.
\thanksmarkseries{⟨*format*⟩} 67
Thanks marks will be printed using ⟨*format*⟩ series of symbols.
\thanksmarkstyle{⟨*defn*⟩} 68
Sets the style for the thanks marks at the foot.
\thanksmarkwidth Width of box for the thanks marks at the foot. 69
\thanksrule The rule to be typeset before the thanks in the foot. 70
thatcher 92
A centered small caps chapterstyle with the number line separated from the title by a short rule.
\theauthor Copy of ⟨*text*⟩ from \author. 67
\begin{thebibliography}{⟨*exlabel*⟩} 305
Environment for typesetting a bibliography. ⟨*exlabel*⟩ is an arbitrary piece of text as wide as the widest label for the bibliographic items.
\thectr Typesets the value of the counter ctr. 331
\thedate Copy of ⟨*text*⟩ from \date. 67
\begin{theglossary} Environment for typesetting a glossary. 319
\begin{theindex} Environment for typesetting an index 309
\thepoem Typeset the current Poem Title number 271
\thepoemline 270
The numeric representation of verse line numbers (default arabic).
\thesheetsequence Typesets the current sheet sequence number. 328
\thetitle Copy of ⟨*text*⟩ from \title. 67
\theTitleReference{⟨*num*⟩}{⟨*text*⟩} 304
Called by \titleref and \currenttitle with the number and text of the title to print the values.
\thicklines Picture declaration for lines to be thick. 416

\thinlines Picture declaration for lines to be thin. ... 416
\thinspace A thin space (3/18 em). ... 342
\thispagestyle{⟨style⟩} ... 108
Sets the pagestyle to ⟨style⟩ for the current page only.
\threecolumnfootnotes Typeset footnotes in three columns. ... 239
\threecolumnfootstyle{⟨series⟩} ... 239
Set the ⟨series⟩ footnotes to be typeset in three column style.
\tightlist In a standard list, removes extra vertical spacing. ... 140
\tightlists ... 140
Declaration removing extra vertical space from list-based environments.
\tightsubcaptions ... 201
Specifies the default vertical space around subcaptions.
\title{⟨text⟩} ... 67
Used by \maketitle to typeset ⟨text⟩ as the document title.
\titleref{⟨labstr⟩} ... 303
Prints the (section, or other) title of the number associated with ⟨labstr⟩ from a \label.
\begin{titlingpage} ... 65
Environment for a title page, resets the page counter to 1 after it
\begin{titlingpage*} ... 65
Like \begin{titlingpage}, but does not reset the page counter.
\titlingpageend ... 65
Can be used to set what kind of page clearing is issued at the end of a titling page. The default for the two args are \cleardoublepage and \clearpage.
\tmarkbl Trim mark for bottom left of trimmed page. ... 327
\tmarkbm Trim mark for bottom middle of trimmed page. ... 327
\tmarkbr Trim mark for bottom right of trimmed page. ... 327
\tmarkml Trim mark for middle left of trimmed page. ... 327
\tmarkmr Trim mark for middle right of trimmed page. ... 327
\tmarktl Trim mark for top left of trimmed page. ... 327
\tmarktm Trim mark for top middle of trimmed page. ... 327
\tmarktr Trim mark for top right of trimmed page. ... 327
\tocnameref Use ToC title for sectional title references. ... 303
\toprule[⟨width⟩] ... 222
Draws a rule across a tabular, default width \heavyrulewidth.
\topskip (length) ... 23
The height of the first line of text on a page. This is usually less than the \baselineskip.
\tracingtabularx ... 228
Writes information about the changing column widths while setting a tabularx.
\traditionalparskip ... 49
Sets the interparagraph spacing to its traditional value.
\tref{⟨labstr⟩} ... 302
Prints a named (\tablerefname) reference to a \labeled table.
\trimedge (length) ... 13
Amount to be trimmed from the fore-edge of the stock paper.
\trimFrame ... 326
Trim mark of a frame drawn round the trimmed page boundary.
\trimLmarks ... 326
Trim marks of 'L' shapes at the four corners of the trimmed page.
\trimmark Cross mark used by \trimXmarks. ... 327
\trimmarks ... 327
Displays 8 (in)visible trim marks round the boundary of the trimmed page.

\trimmarkscolor ... 326
Empty macro that can be redefined to give a specific color to the trimmarks.
\trimNone No trim marks. ... 326
\trimtop (length) ... 13
Amount to be trimmed from the top of the stock paper.
\trimXmarks ... 326
Trim marks of crosses at the four corners of the trimmed page.
\ttfamily Declaration for using a Typewriter (monospaced) font. ... 42
\twocolindex Typeset index in two columns (the default). ... 309
\twocoltocetc Set the ToC, etc., in two columns. ... 151
twocolumn Class option for two columns. ... 4
\twocolumnfootnotes Typeset footnotes in two columns. ... 239
\twocolumnfootstyle{⟨series⟩} ... 239
Set the ⟨series⟩ footnotes to be typeset in two column style.
twoside Class option for text on both sides of the paper. ... 4
\TX@verb A poor man's \verb for use in a tabularx. ... 229
\typeoutlayout ... 26
Outputs the current values of the class layout parameters to the terminal and log file.
\typeoutstandardlayout ... 26
Outputs the current values of the standard LaTeX layout parameters to the terminal and log file.

\ucminusname ... 338
Lowercase 'minus' name with initial uppercase letter, default 'Minus'.
\undodrop ... 260
Following a \dropchapter restores subsequent chapter heads to their normal position.
\unitlength The unit of length in a picture environment. Default 1pt. ... 415
\unnamedsubappendices ... 78
Do not precede the sub-appendix number with any name (the default).
\upbracefill Fills a tabular column with an up brace. ... 224
\uppercaseheads ... 114
Defines \memUChead as equivalent to \MakeUppercase.
\uppercaseheads Set the titles in the headings pagestyle in Uppercase. ... 109
\upshape Declaration for using an upright font. ... 42
\usethanksrule ... 70
Specifies that the \thanksrule is to be typeset in the titlingpage environment.

\vector(⟨dx,dy⟩){⟨distance⟩} ... 425
Picture object of a line with an arrowhead at the end, slope ⟨dx,dy⟩ and coordinate length ⟨distance⟩.
veelo ... 92
A raggedleft large bold chapterstyle with a large black square in the margin by the number line. It requires the graphicx package.
\begin{verbatim} Typeset the contents verbatim. ... 286
\verbatimbreakchar{⟨char⟩} ... 287
Character indicating a verbatim line is being wrapped.
\verbatimindent Indent for wrapped overlong verbatim lines. ... 287
\verbatiminput{⟨file⟩} ... 294
Acts like verbatim except the contents is read from the ⟨file⟩ file.
\verbatiminput*{⟨file⟩} ... 294
Acts like verbatim* except the contents is read from the ⟨file⟩ file.

`\begin{verbatimoutput}{`⟨*file*⟩`}` 293
The contents of the environment are written verbatim to the ⟨*file*⟩ file, overwriting anything previously in the file.
`\verbfootnote[`⟨*num*⟩`]{`⟨*text*⟩`}` 238
Like `\footnote` except that ⟨*text*⟩ can contain verbatim material.
`\begin{verse}[`⟨*length*⟩`]` 266
Environment for typesetting verse; if given the midpoint of ⟨*length*⟩ is placed at the center of the typeblock measure.
`\verselinebreak[`⟨*length*⟩`]` 267
Makes a break in a verse line, indenting the next part by twice `\vgap`, or by ⟨*length*⟩ if it is given.
`\verselinenumbersleft` 270
Following this declaration verse line numbers are set at the left of the verse lines.
`\verselinenumbersright` 270
Following this declaration verse line numbers are set at the right of the verse lines.
`\versewidth` Scratch length, typically for use in verse typesetting. 266
verville 92
A single line, large, centered, chapterstyle with rules above and below.
`\vinphantom{`⟨*text*⟩`}` Leaves a space as wide as ⟨*text*⟩. 267
`\vleftmargin` General verse indent from the left of the typeblock. 266
`\vleftofline{`⟨*text*⟩`}` 'Hanging left' ⟨*text*⟩ at the start of a verse line. 268
`\vleftskip` 270
Space between the argument to `\flageverse` and `\flagverse`.
`\begin{vplace}[`⟨*num*⟩`]` 345
The contents of this environment are centered vertically. The optional ⟨*num*⟩ argument can be used to specify the ratio of the upper space to the lower space.
`\vrightskip` 270
Verse line numbers are set distance `\vrightskip` into the margin.

wilsondob A one line flushright chapterstyle in a large italic font. 92
`\wrappingoff` 287
The normal behaviour of not wrapping overlong verbatim lines.
`\wrappingon` Wrap overlong verbatim lines. 287
`\begin{writeverbatim}{`⟨*stream*⟩`}` 293
The contents of the environment are written verbatim to the ⟨*stream*⟩ stream.

`\Xheadstart` Generic macro called before printing a 'X List of' title. 152
`\xlvchars` (length) Approximate length of 45 characters. 15
`\Xmark` Generic macro setting the marks for the 'X List of'. 152

`\zerotrivseps` Eliminate space before and after a `trivlist`. 143

Bibliography

CTAN is the Comprehensive TeX Archive Network. Information on how to access CTAN is available at http://www.tug.org.

[AHK90] Paul W. Abrahams, Kathryn Hargreaves and Karl Berry. *TeX for the Impatient*. Addison-Wesley, 1990. (Available at ftp://tug.org/tex/impatient)

[Ars99] Donald Arseneau. *The url package*. February, 1999. (Available from CTAN via /macros/latex/contrib/url/)

[Ars01a] Donald Arseneau. *The titleref package*. April, 2001. (Available from CTAN via /macros/latex/contrib/titleref/)

[Ars01b] Donald Arseneau. *The chapterbib package*. September, 2001. (Available from CTAN via /macros/latex/contrib/cite/)

[Ars07] Donald Arseneau. *The framed package* v0.95. October, 2007. (Available from CTAN via /macros/latex/contrib/framed/)

[Ber02] Jens Berger. *The titlesec and titletoc packages*. September, 2002. (Available from CTAN via /macros/latex/contrib/titlesec/)

[Bez99] Javier Bezos. *The titlesec and titletoc packages*. February, 1999. (Available from CTAN via /macros/latex/contrib/titlesec/)

[Bir04] Derek Birdsall. *notes on book design*. Yale University Press, 2004. ISBN 0–300–10347–6.

[Bra97] Johannes Braams. *The alltt environment*. June, 1997. (Available as `alltt.dtx` and `alltt.ins` from CTAN via /macros/latex/base/)

[Bri99] Robert Bringhurst. *The Elements of Typographic Style*. Hartley & Marks, second edition, 1999. ISBN 0–88179–033–8.

[Car14] David Carlisle. *The delarray package*. October, 2014. (Available from CTAN via /macros/latex/required/tools/)

[Car95] David Carlisle. *The afterpage package*. October, 1995. (Available from CTAN via /macros/latex/required/tools/)

[Car98b] David Carlisle. *The longtable package*. May, 1998. (Available from CTAN via /macros/latex/required/tools/)

[Car98c] David Carlisle. *The enumerate package*. August, 1998. (Available from CTAN via /macros/latex/required/tools/)

[Car16] David Carlisle. *The tabularx package*. February, 2016. (Available from CTAN via /macros/latex/required/tools/)

[CR99] David Carlisle and Sebastian Rahtz. *The graphicx package*. February, 1999. (Available from CTAN via /macros/latex/required/graphics/)

[Car14] David Carlisle. *The dcolumn package*. May, 2001. (Available from CTAN via /macros/latex/required/tools/)

[Car19] David Carlisle. *The textcase package*. September, 2019. (Available from CTAN in /macros/latex/contrib/textcase)

[Car05] David Carlisle. *Packages in the graphics bundle* (includes the color package). November, 2005. (Available from CTAN via /macros/latex/required/graphics/)

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

[CH88] Pehong Chen and Michael A. Harrison. 'Index Preparation and Processing'. *Software: Practice and Experience*, 19:8, pp. 897–915, September, 1988. (Available from CTAN via /indexing/makeindex/paper/)

[Chi93] *The Chicago Manual of Style*, Fourteenth Edition. The University of Chicago, 1993. ISBN 0–226–10389–7.

[Dal99a] Patrick W. Daly. *Natural Sciences Citations and References*. May, 1999. (Available from CTAN via /macros/latex/contrib/natbib/)

[Dal99b] Patrick W. Daly. *Customizing Bibliographic Style Files*. August, 1999. (Available from CTAN via /macros/latex/contrib/custom-bib)

[Dow96] Geoffrey Dowding. *Finer Points in the Spacing & Arrangement of Type*. Hartley & Marks, 1996. ISBN 0–88179–119–9.

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

[Eij92] Victor Eijkhout. *TeX by Topic*. Addison-Wesley, 1992. ISBN 0–201–56882–9. (Available from http://www.eijkhout.net/tbt/).

[Eij99] Victor Eijkhout. *comment.sty* October, 1999. (Available from CTAN via /macros/latex/contrib/comment/)

[Fai00] Robin Fairbairns. *footmisc — a portmanteau package for customising footnotes in LaTeX2e*. March, 2000. (Available from CTAN via /macros/latex/contrib/footmisc/)

[FAQ] Robin Fairbairns. *The UK TeX FAQ*. (Available from CTAN via http://faq.tug.org/)

[Far90] Gerald Farin. *Curves and Surfaces for Computer Aided Geometric Design — A Practical Guide*. Academic Press, 2 edition, 1990.

[FP80] I. D. Faux and M. J. Pratt. *Computational Geometry for Design and Manufacture*. Ellis Horwood, 1980.

[Fea16] Simon Fear. *Publication quality tables in LaTeX*. April, 2016. (Available from CTAN via /macros/latex/contrib/booktabs)

[Fly98] Peter Flynn. *Formatting Information: A Beginner's Introduction to Typesetting with LaTeX2*. 2002. (Available from CTAN via /info/beginlatex/)

[GM+07] Michel Goossens, Frank Mittelbach, et al. *The LaTeX Graphics Companion: Second edition*. Addison-Wesley, 2007. ISBN 0–321–50892–0.

[GR99] Michel Goossens and Sebastian Rahtz (with Eitan Gurari, Ross Moore and Robert Sutor). *The LaTeX Web Companion: Integrating TeX, HTML and XML*. Addison-Wesley, 1999. ISBN 0–201–43311–7.

[Hoe98] Alan Hoenig. *TeX Unbound: LaTeX and TeX strategies for fonts, graphics, and more*. Oxford University Press, 1998. ISBN 0–19–509686–X.

[Jon95] David M. Jones. *A new implementation of LaTeX's indexing commands*. September, 1995. (Available from CTAN via /macros/latex/contrib/camel)

[Keh98] Roger Kehr. *xindy: A flexible indexing system*. February, 1998. (Available from CTAN via /indexing/xindy/)

[Ker07] Uwe Kern. *Extending LaTeX's color facilities: the xcolor package*. January, 2007. (Available from CTAN via /macros/latex/contrib/xcolor/)

[Kha10] Vafa Khalighi. *The Bidi package*. 2010. (Available from CTAN via /macros/latex/contrib/bidi/)

[Knu84] Donald E. Knuth. *The TeXbook*. Addison-Wesley, 1984. ISBN 0–201–13448–9.

[Knu86] Donald E. Knuth. *TeX: The Program*. Addison-Wesley, 1986. ISBN 0–201–13437–3.

[Knu87] Donald E. Knuth. *Computer Modern Typefaces*. Addison-Wesley, 1987. ISBN 0–201–134446–2.

[Knu92] Donald E. Knuth. *The METAFONT Book*. Addison-Wesley, 1992. ISBN 0–201–13444–6.

[Lam94] Leslie Lamport. *LaTeX: A Document Preparation System*. Addison-Wesley, 1994. ISBN 0–201–52983–1.

[LEB04] Leslie Lamport, Victor Eijkhout and Johannes Braams. *NTG document classes for LaTeX version 2e*. June, 2004. (Available from CTAN via /macros/latex/contrib/ntgclass/)

[Lea03] Christopher League. *TeX support for the FontSite 500 CD*. May 2003. (Available from http://contrapunctus.net/fs500tex)

[Leh04] Philipp Lehman. *The Font Installation Guide*. December 2004. (Available from CTAN via /info/Type1fonts/fontinstallationguide)

[Leu92] Mary-Claire van Leunen. *A Handbook for Scholars*. Oxford University Press, 1992. ISBN 0–19–506954–4.

[Lon91] F. W. Long. *multind*. August, 1991. (Available from CTAN as /macros/latex209/contrib/misc/multind.sty)

[Mad06] Lars Madsen. *Various chapter styles for the memoir class*. July, 2006. (Available from CTAN via /info/latex-samples/MemoirChapStyles/)

[Mad07] Lars Madsen. *The Memoir Experimental Support Package*. 2007. (Available from CTAN via /macros/latex/contrib/memexsupp/)

[McD98] Rowland McDonnell. *The sectsty package*. November, 1998. (Available from CTAN via /macros/latex/contrib/sectsty/)

[McL80] Ruari McLean. *The Thames & Hudson Manual of Typography*. Thames & Hudson, 1980. ISBN 0–500–68022–1.

[Mit18] Frank Mittelbach. *An environment for multicolumn output*. April, 2018. (Available from CTAN (the `multicol` package) via /macros/latex/required/tools/)

[Mit2020a] Frank Mittelbach *The ltfilehook package* July 2020.

[Mit2020b] Frank Mittelbach *The lthooks package* July 2020.

[Mit2020c] Frank Mittelbach *The ltshipout package* July 2020.

[MC18] Frank Mittelbach and David Carlisle. *A new implementation of LaTeX's tabular and array environment*. May, 2018. (Available from CTAN (the `array` package) via /macros/latex/required/tools/)

[MC00] Frank Mittelbach and David Carlisle. *The fixltx2e package*. December, 2016. As of 2015 the functionality of this package has been merged with the LaTeX kernel. (Available from CTAN via /macros/latex/base/)

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

[Mor85] Michael E. Mortenson. *Geometric Modeling*. John Wiley & Sons, 1985.

[NG98] Rolf Niespraschk and Hubert Gäßlein. *The sidecap package*. June, 1998. (Available from CTAN via /macros/latex/contrib/sidecap/)

[Oet] Tobias Oetiker. *The Not So Short Introduction to LaTeX2e*. (Available from CTAN via /info/lshort/)

[Oos96] Piet van Oostrum. *Page Layout in LaTeX*. June, 1996. (Available from CTAN via /macros/latex/contrib/fancyhdr/)

[Pak01] Scott Pakin. *The Comprehensive LaTeX Symbol List*. July, 2001. (Available from CTAN via /info/symbols/comprehensive/)

[dP84] H. de Parville. Recreations mathematique: La Tour d'Hanoi et la question du Tonkin. *La Nature*, part I:285–286, Paris 1884.

[Pat88a] Oren Patashnik. *BibTeXing*. February, 1988. (Available from CTAN as /bibliography/bibtex/distribs/doc/btxdoc.tex)

[Pat88b] Oren Patashnik. *Designing BibTeX Styles*. February, 1988. (Available from CTAN as /bibliography/bibtex/distribs/doc/btxhak.tex)

[Pug02] Diego Puga. *The Pazo Math fonts for mathematical typesetting with the Palatino fonts*. May, 2002. (Available from CTAN via /fonts/mathpazo/)

[Rahtz01] Sebastian Rahtz. *Section name references in LaTeX*. January, 2001. (Available from CTAN (the `nameref` package) via /macros/latex/contrib/hyperref/)

[Rahtz02] Sebastian Rahtz. *Hypertext marks in LaTeX*. May, 2002. Now maintained and frequently updated by Heiko Oberdiek. (Available from CTAN via /macros/latex/contrib/hyperref/)

[Rec97] Keith Reckdahl. *Using Imported Graphics in LaTeX2e*. December, 1997. Updated in 2006. (Available from CTAN via /info/epslatex.pdf)

[Rei07] Edward M. Reingold. 'Writing numbers in words in TeX'. TUGboat, 28, 2 pp 256–259, 2007.

[RBC74] W. W. Rouse Ball and H. S. M. Coxeter. *Mathematical Recreations and Essays*. University of Toronto Press, twelfth edition, 1974.

[SW94] Douglas Schenck and Peter Wilson. *Information Modeling the EXPRESS Way*. Oxford University Press, 1994. ISBN 0–19–508714–3.

[SRR99] Rainer Schöpf, Bernd Raichle and Chris Rowley. *A New Implementation of LaTeX's verbatim and verbatim* Environments*. December, 1999. (Available from CTAN via /macros/latex/required/tools/)

[Sch07] Martin Scharrer, *Version Control of LaTeX Documents with svn-multi*. *The PracTEX Journal*, 3, 2007. ISSN 1556-6994.

[Sch09] Martin Scharrer, *The svn-multi package*, 2009. (Available from CTAN via /macros/latex/contrib/svn-multi/)

[Sne04] Maarten Sneep. *The atmosphere in the laboratory: cavity ring-down measurements on scattering and absorption*. Phd thesis, Vrijie Universiteit, Amsterdam, 2004.

[Tal06] Nicola L. C. Talbot. *datetime.sty: Formatting Current Date and Time*. December, 2006. (Available from CTAN via /macros/latex/contrib/datetime/)

[Thi98] Christina Thiele. 'Hey — it works: Ornamental rules'. *TUGboat*, vol. 19, no. 4, p 427, December 1998.

[Thi99] Christina Thiele. 'The Treasure Chest: Package tours from CTAN', *TUGboat*, vol. 20, no. 1, pp 53–58, March 1999.

[TJ05] Kresten Krab Thorup, Frank Jensen (and Chris Rowley). *The calc package — Infix notation arithmetic in LaTeX*. August, 2005. (Available from CTAN via /macros/latex/required/tools/)

[Tob00] Geoffrey Tobin. *setspace.sty*. December, 2000. (Available from CTAN via /macros/latex/contrib/setspace/)

[Tsc91] Jan Tschichold. *The Form of the Book*. Lund Humphries, 1991. ISBN 0–85331–623–6.

[Ume99] Hideo Umeki. *The geometry package*. November, 1999. (Available from CTAN in /macros/latex/contrib/geometry/)

[Wil00] Graham Williams. *The TeX Catalogue*. (Latest version on CTAN as /help/Catalogue/catalogue.html)

[Wil93] Adrian Wilson. *The Design of Books*. Chronicle Books, 1993. ISBN 0–8118–0304–X.

[Wil99b] Peter Wilson. *The tocvsec2 package*. January, 1999. (Available from CTAN via /macros/latex/contrib/tocvsec2/)

[Wil00a] Peter Wilson. *The epigraph package*. February, 2000. (Available from CTAN via /macros/latex/contrib/epigraph/)

[Wil00b] Peter Wilson. *LaTeX files for typesetting ISO standards*. February, 2000. (Available from CTAN via /macros/latex/contrib/isostds/iso/)

[Wil00c] Peter Wilson. *The nextpage package*. February, 2000. (Available from CTAN as /macros/latex/contrib/misc/nextpage.sty)

[Wil00e] Peter Wilson. *The xtab package*. April 2000. (Available from CTAN via /macros/latex/contrib/xtab)

[Wil01a] Peter Wilson. *The abstract package*. February, 2001. (Available from CTAN via /macros/latex/contrib/abstract/)

[Wil01d] Peter Wilson. *The ccaption package*. March, 2001. (Available from CTAN via /macros/latex/contrib/ccaption/)

[Wil01e] Peter Wilson. *The chngcntr package*. April, 2001. (Available from CTAN via /macros/latex/contrib/chngcntr/)

[Wil01f] Peter Wilson. *The hanging package*. March, 2001. (Available from CTAN via /macros/latex/contrib/hanging/)

[Wil01g] Peter Wilson. *The titling package*. March, 2001. (Available from CTAN via /macros/latex/contrib/titling/)

[Wil01h] Peter Wilson. *The tocbibind package*. April, 2001. (Available from CTAN via /macros/latex/contrib/tocbibind/)

[Wil01i] Peter Wilson. *The tocloft package*. April, 2001. (Available from CTAN via /macros/latex/contrib/tocloft/)

[Wil03b] Peter Wilson. *ledmac: A presumptuous attempt to port EDMAC and TABMAC to LaTeX* November, 2003. (Available from CTAN via /macros/latex/contrib/ledmac/)

[Wil04a] Peter Wilson. *The bez123 and multiply packages*, April 2004. (Available from CTAN in /macros/latex/contrib/bez123/)

[Wil04b] Peter Wilson. *The pagenote package* September, 2004. (Available from CTAN via /macros/latex/contrib/pagenote/)

[Wil04c] Peter Wilson. *The layouts package* October, 2004. (Available from CTAN in /macros/latex/contrib/layouts/)

[Wil07a] Peter Wilson. *Some Examples of Title Pages*. Herries Press, 2007. (Available from CTAN via /info/latex-samples/TitlePages/)

[Wil07e] Peter Wilson. 'Glisterings', *TUGboat*, 28(2):229–232, 2007.

[Wil08a] Peter Wilson. *The changepage package*. March, 2008. (Available from CTAN via /macros/latex/contrib/changepage/)

[Wil08b] Peter Wilson. 'Glisterings', *TUGboat*, 29(2):324–327, 2008.

[Wil09a] Peter Wilson. *The fonttable package* April, 2009. (Available from CTAN via /macros/latex/contrib/fonttable/)

[Wil19a] Peter Wilson (with the assistance of Lars Madsen). *The LaTeX memoir class for configurable book typesetting: source code* November, 2019. (Available from CTAN via /macros/latex/contrib/memoir/)

[Wil19b] Peter Wilson (with the assistance of Lars Madsen). *The Memoir Class for Configurable Typesetting — User Guide* November, 2019. Regularly updated. (Available from CTAN via /macros/latex/contrib/memoir/)

[Wil18] Peter Wilson. *A Few Notes on Book Design* September, 2018. (Available from CTAN via /info/memdesign/)

[Wil??] Peter Wilson. *A Rumour of Humour: A scientist's commonplace book*. To be published.

[Wri19] Joseph Wright. *Siunitx — A comprehensive (SI) units package* October, 2019. (Available from CTAN via /macros/latex/contrib/siunitx)

[Zan98] Timothy Van Zandt. *Documentation for fancybox.sty: Box tips and tricks for LaTeX*, November, 1998. (Available from CTAN via /macros/latex/contrib/fancybox/)

Index

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

Analphabetics

\!, 342, 343
! (level specifier), 312
!h float specifier ..., 469
\#, 456, 462
\&, 457
\(, 455, 464
\), 464
*pt (option), 3, 4, 474
\+, 468
\,, 342
\-, 463, 468
... allowed only in math mode, 464
... at index ... in pattern ..., 474
... is negative, 471
... is not a counter, 471
... is not a macro, 474
... is not an input stream, 474
... is not an output stream, 474
... is zero or negative, 471
\:, 342
\;, 342
\<, 464, 468
<{...}at wrong position ..., 471
\=, 467, 468
=, 440
\>, 468
>{...}at wrong position ..., 471
@,
in macro code, 149, 170, 191, 207, 362, 368, 439, 461, 462
@ (actual specifier), 314
@-expression, 466
\@@end, 460
\@@wrglom@m, 320
\@MM, 443
\@Mi, 443
\@Mii, 443
\@Miii, 443
\@Miv, 443
\@bsphack, 299
\@caption, 206, 207
\@captype, 206
\@cclv, 443
\@cclvi, 443
\@cftasnum, 163
\@cftasnumb, 163
\@cftbsnum, 163
\@chapapp, 88, 116
\@dblfpbot (length), 185, 186
\@dblfpsep (length), 185
\@dblfptop (length), 185, 186
\@dblftop (length), 186
\@dotsep, 146, 149, 156
\@dottedtocline, 148
\@esphack, 299
\@fibnext, 449
\@fibseries, 449–451
\@fnsymbol, 243
\@fpbot (length), 185, 186
\@fpsep (length), 185, 186
\@fptop (length), 185, 186
\@hangfrom, 97, 128, 359
\@m, 443
\@makecaption, 206–209

\@makefnmark, 240
\@minus, 444
\@namedef, 441
\@nameuse, 441
\@ne, 443
\@plus, 444
\@pnumwidth, 146, 153, 160
\@pnumwidth (length), 157
\@seccntformat, 97
\@setref, 299
\@startsection, 97, 170
\@tempdima, 447
\@tempdimb, 447
\@thefnmark, 240, 241
\@tocrmarg, 146, 148, 153, 160
\@whiledim, 450
\@whilenum, 450, 451
\@xfloat, 206
\@xxxii, 443
\@zeroseps, 291, 388
\[, 211, 218, 455, 464
\\, 77, 131, 189–191, 211, 222, 223, 228, 263, 265–267, 387, 421, 422, 455, 458, 459, 461, 467
\ (escape specifier), 315
" (quote specifier), 315
*, 189, 265, 266
\\>, 267, 275
\\!, 266
\< in mid line, 464
\LoadClass in package file, 465
\include cannot be nested, 465
\verb illegal in command argument, 279
\], 211, 218, 464
| (encap specifier), 314
10pt (option), 2, 3, 5, 46
11pt (option), 2, 46
12pt (option), 2, 6, 46
14pt (option), 3, 46, 113
17pt (option), 3, 46, 473
20pt (option), 3, 46
25pt (option), 3, 46, 473
30pt (option), 3, 46
36pt (option), 3, 46
48pt (option), 3, 46
60pt (option), 3, 46
9pt (option), 2, 46

A

A box was supposed to be here, 454
A pattern has not been specified, 471
a3paper (option), 1
A4 paper, 8
a4paper (option), 1, 27, 29
a5paper (option), 1
a6paper (option), 1
a7paper (option), 1
\abnormalparskip, 49
\abovecaptionskip, 210
\abovecaptionskip (length), 207
\aboverulesep (length), 222–224
\abslabeldelim, 72
\absleftindent (length), 72
\absnamepos, 72, 73
\absparindent, 72, 73
\absparindent (length), 72, 73
\absparsep, 72
\absparsep (length), 72
\absrightindent (length), 72
\abstitleskip (length), 73
abstract, 66, 71
  heading, 71
  one column, 73–74
  styling, 71–73
abstract (environment), xxiii, 71–74, 344
abstract (package), xxix, 71, 345
\abstractcol, 71, 72
\abstractintoc, 71, 72
\abstractname, 71, 72, 344
\abstractnamefont, 72
\abstractnum, 71, 72
\abstractrunin, 71–73
\abstracttextfont, 72
Adams, William, 90, 328
add to a macro, 332
add to contents, 148
\addappheadtotoc, 78, 79
\addcontentsline, 145, 146, 148, 149, 151, 171, 193, 194
\added, 325, 326
\addlinespace, 223
\addpenalty, 467
\addperiod, 343

\addtocontents, 148, 149, 161, 171
\addtocounter, 107, 463, 466
\addtodef, 168, 332, 371, 474
\addtodef*, 333
\addtoiargdef, 332, 333, 474
\addtoiargdef*, 333
\addtolength, 22, 72, 86, 142, 266, 378, 446
\addtonotes, 255
\addtopsmarks, 116
\addtostream, 293
\addvspace, 87, 467
adjustwidth (environment), 90, 135, 136, 140, 142, 284
adjustwidth* (environment), 66, 135
\advance, 443, 444, 446
\afterbookskip, 80, 82
\afterchapskip (length), 85, 86
\afterchapternum, 85, 86, 89
\afterchaptertitle, 85, 86, 152, 358
\afterepigraphskip (length), 259
\afternextrecto, 171
\afternextverso, 171
\afterpage, 171
afterpage (package), 171, 196
\afterpartskip, 80, 83, 262
\afterPoemTitle, 271, 272
\afterPoemTitlenum, 271, 272
\afterPoemTitleskip (length), 272
\afterXtitle, 152
pagenote, 254
mainmatter, 80
\aliaspagestyle, 112
alltt (environment), 263, 264
alltt (package), 263, 385
Alph, 107
alph, 107
alphabet length, 15
alphabetic numbering, 107
\alsoname, 315, 344
Alte Schwabacher, 479
altverse (environment), 267, 269, 273
\amname, 330, 344
amsfonts (package), 465
amsmath (package), 385
\and, 67
\andnext, 67
anonymous division, 100
    styling, 101–102
ans (file), 298
answer (environment), 298, 299
answer (package), 300
\anyptfilebase, 4, 474
\anyptsize, 4, 474
appendices (environment), 78
appendix, 78, 82, 88, 150, 159, 160
    subappendix, 78
\appendix, 78, 116, 372
appendix (package), xxix, 345
\appendixname, 78, 79, 116, 255, 344
\appendixpage, 78, 82, 161, 344, 412, 414
\appendixpage*, 78
\appendixpagename, 78, 344
\appendixrefname, 302
\appendixtableofcontents, 414
\appendixtocname, 78, 344
arabic, 107
\Aref, 302
marginpar, 175
marginparmargin, 175
argument,
    optional, 300
Argument of ... has an extra }, 454
Argument to \overridesidecapmargin neither ..., 472
Argument to \setsidecappos is not ..., 472
Arial, 363, 379
Arithmetic overflow, 454
array, 455
    intercolumn space, 230
    row fill, 224–226
    row spacing, 230
array (environment), 211, 213, 217, 218, 224, 228, 230, 232, 455, 457, 458, 463, 465, 466, 471, 472
array (package), xxvii, xxx, 211, 212, 345
\arraybackslash, 228
\arraycolsep (length), 230
\arrayrulewidth (length), 230
\arraystretch, 230
\arraytostring, 340
arrow, *see* vector

Arseneau, Donald, 53, 243, 246, 280, 282, 345
article (chapterstyle), 88
article (class), xxv, 5, 71, 88, 148
article (option), 5, 77, 84, 88, 93
asu (chapterstyle), 372
asu (pagestyle), 370
asuappendix (chapterstyle), 372
\At...File, 347
\AtBeginClass, 347
\AtBeginDocument, 346, 348
\AtBeginEnvironment, 140
\AtBeginFile, 346
\AtBeginPackage, 347
\atcentcr, 131
\atcentercr, 131
patchcmd, 472
\AtEndClass, 347
\AtEndDocument, 346
\AtEndFile, 346
\AtEndOfClass, 346
\AtEndOfPackage, 346
\AtEndPackage, 347
\atendtheglossaryhook, 322, 323
ATF Clearface, 479
\author, 57, 64, 66, 67, 469, 475
\autocols, 234
\autorows, 233, 234
aux (file), 292, 301, 437, 438
Avant Garde Gothic, 16, 38

B

b (position argument), 174, 181, 186, 419, 428
b3paper (option), 1
b4paper (option), 1
b5paper (option), 1
b6paper (option), 1
b7paper (option), 1
babel (package), xxv
back matter, 75, 305
\backmatter, 75, 77, 78
Bad \line or \vector argument, 464
Bad \sidebarmargin argument, 474
Bad math environment delimeter, 464
Barbedour, 479
Basel, 479
\baselineskip, 50, 51, 86
\baselineskip (length), 21, 23, 25, 50, 51, 128, 207, 259, 342, 388
Basilia, 479
Baskerville, 15, 479
bch (fontfamily), 39
\beforebookskip, 80
\beforechapskip, 86
\beforechapskip (length), 85, 86, 405
\beforeepigraphskip (length), 259
\beforepartskip, 80, 262
\beforePoemTitleskip (length), 272
\begin, 164, 290, 464, 465
\begintheglossaryhook, 322
\begin{...} ended by \end{...}, 464
\belowcaptionskip, 210
\belowcaptionskip (length), 207
\belowrulesep (length), 222, 223
Bembo, 479
Benguiat, Ed, 39
Benton, Morris, 40
Bergamo, 479
Bernhard Modern, 479
Bezier curve, *see also* picture object
  control points, 432
  linearized rendition, 433
Bezier, Pierre, 431
\bfseries, 42, 81, 82, 86, 96, 138
bianchi (chapterstyle), 90
Bianchi, Stefano, 90
bib (file), 307, 308
\bibindent, 5
\bibintoc, 305
\bibitem, 305, 307, 468, 469, 475
bibitemlist (environment), 305–307
\bibitemsep (length), 307
\biblabel, 470
bibliographic database, 307
bibliography, 5, 160, 170, 262, 305–308
  explanatory text, 306
  flushleft entries, 306
  heading, 306
  label styling, 307
  list styling, 306–307
  title in ToC, 305
\bibliography, 307

\bibliographystyle, 307
\biblistextra, 306, 373
\bibmark, 306
\bibname, 305, 344
\bibsection, 305, 306, 373
BibTeX database, 308
BibTeX (program), xxv, 307, 308
BibTeX style, 307, 308
  abbrv, 308
  alpha, 308
  changing, 308
  plain, 308
  unsrt, 308
\bicaption, 197, 198
\bicontcaption, 198
bidi (package), xxx, 352, 353
big point, xxxvi, xxxvii
bilingual captions, 196–199
binding allowance, 19, 23, 25
\bionenumcaption, 196, 197
bitmap, 35
  font, 35, 49
\bitwonumcaption, 196, 197
blank space, 267
blockdescription (environment), 137
\blockdescriptionlabel, 137
Bodoni, 479
body font, 41, 45
boek (class), 92, 105
\boldmath, 469
book, 76
  number, 77, 79
\book, 76, 77, 79, 80, 82, 84, 110, 153, 154, 156, 161, 162, 261, 262, 344, 348, 412
book (class), xxv, 71, 87, 88, 103, 105, 116, 148
book (pagestyle), 80, 109, 110
\book*, 348
\bookblankpage, 82
Bookman, 15, 16, 38
bookmark (package), 172
\bookname, 81, 344
\booknamefont, 81, 103
\booknamenum, 81
\booknumberline, 156
\booknumberlinebox, 156
\booknumberlinehook, 156
\booknumfont, 81, 103
\bookpageend, 82
\bookpagemark, 82
\bookrefname, 302, 344
booktabs (package), xxvii, xxix, 211, 222, 345
\booktitlefont, 82, 103
\bottomfraction, 185–187
bottomnumber (counter), 185
\bottomrule, 222, 223
\bottomsectionpenalty, 77, 78
\bottomsectionskip (length), 77, 78
box, 173, 231, 279, 418
  framed, 280–284, 418
  include pagebreak, 280
  picture, 416
  reference point, 418
  rounded, *see also* oval
    half, 427
    quarter, 427
  saved object, 422
  shaded background, 280
  unbreakable, 416
  using saved object, 422
\box, 454
\box255, 459
boxed verbatim, 287
boxedverbatim (environment), 287–289, 294, 295
boxedverbatim* (environment), 287
\boxedverbatiminput, 294
\boxedverbatiminput*, 294
bp, xxxvi
\break, 131
\Bref, 302, 344
bringhurst (chapterstyle), 90, 105
bringhurst (headstyles), 104, 105
bringhurst (package), 362
Bringhurst, Robert, 13
\bringpicl, 360
\bringpicr, 360
broadsheetpaper (option), 2
brotherton (chapterstyle), 90
\bs, 349
bst (file), 308
buffer size, 463

\bvbox, 287–289
\bvboxsep (length), 287, 288
\bvendofpage, 288
\bvendrulehook, 288
\bvleftsidehook, 288
\bvnumberinside, 289
\bvnumbersinside, 289
\bvnumbersoutside, 289
\bvperpagefalse, 288
\bvperpagetrue, 288
\bvrightsidehook, 288
\bvsides, 287–289
\bvtopandtail, 287–289
\bvtopmidhook, 288
\bvtopofpage, 288
\bvtoprulehook, 288
by (keyword), 443

C

c (position argument), 232, 233
calc (package), 25, 27, 248, 409, 410
\calccentering, 66, 136, 137
Can only be used in the preamble, 464
\cancelthanksrule, 70
Cannot change a macro that has delimited arguments, 472
caption, 75, 160, 173, 187–210
  LaTeX methods, 206–209
  anonymous, 187, 192–195
  bilingual, 196–199
    in list of, 196
    styling, 198
  continuation, 187
  continued, 191–192
    outside a float, 195
  delimeter, 188, 208
  fixed, 178, 195–196, 232
  font, 188, 190, 207, 208
  footnote, 209–210
  multiline, 188, 189, 191
  multiple, 178
  new subcaption, 180
  on opposite page, 195
  outside a float, 195–196
  paragraph style, 188
  ruled, 189
  short style, 188
  side, 202–205
  side caption, 189
  space above, 207
  style, 173, 187–191
  subcaption, 199–202
    continued, 200
    in list of, 199
    referencing, 201
    styling, 201
  width, 189
\caption, 146, 154, 160, 171, 173, 191, 192, 194–196, 198, 199, 201, 206, 209, 232, 301, 303, 304, 323, 344, 349, 361, 464, 465
\caption outside float, 465
caption (package), 187, 210
\captiondelim, 188
\captionnamefont, 188, 204
\captionstyle, 188, 189
\captiontitlefinal, 189
\captiontitlefont, 188
\captionwidth, 189
cardinal,
\cardinal, 336
Carnase, Tom, 38
Carter, Matthew, 39
\cases, 219
Caslon, 479
cc, xxxvi
ccaption (package), xxvii, xxix, 187, 345
\cdot, 214
Centaur, 479
\center, 129
center (environment), 73, 129, 136, 143, 144, 207, 387
\centerfloat, 129
centering,
  float, 129
  vertical, 345
\centering, 128–131, 188, 190, 202, 205, 207, 228, 272, 387
\centerlastline, 188, 191, 202
centimetre, xxxvi
Century, 363
Century Old Style, 63, 479
\cftaddnumtitleline, 160

\cftaddtitleline, 160
\cftbeforeKskip (length), 154
\cftbookbreak, 154
\cftchapterbreak, 154
\cftchapterleader, 157
\cftdot, 153
\cftdotsep, 153, 156
\cftinsertcode, 161
\cftinserthook, 161
\cftKafterpnum, 157, 158, 163
\cftKaftersnum, 155, 156, 158, 163
\cftKaftersnumb, 155, 156, 158, 163
\cftKdotsep, 156
\cftKfont, 155, 158, 163
\cftKformatpnum, 157
\cftKformatpnumhook, 157
\cftKindent (length), 154
\cftKleader, 156, 158, 159
\cftKname, 155
\cftKnumwidth (length), 154, 155
\cftKpagefont, 157
\cftKpresnum, 155, 156, 163
\cftlocalchange, 160
\cftnodots, 153, 156
\cftpagenumbersoff, 159, 170
\cftpagenumberson, 159
\cftparfillskip, 159
\cftparskip (length), 153
\cftpartbreak, 154
\cftsetindents, 157, 169
\cftwhatismyname, 156, 410
\changecaptionwidth, 189
\changed, 325, 326
\changeglossactual, 321
\changeglossnum, 321, 322
\changeglossnumformat, 321
\changeglossref, 321
\changemarks, 325
changepage (package), 334
Chantilly, 479
\chapindent (length), 89, 90
\chapnamefont, 86, 103
\chapnumfont, 86, 103
chappell (chapterstyle), 90, 402
chapter, 4, 5, 75, 76, 78, 146
    number, 77, 78
    precis, 92–94
\chapter, 5, 75–80, 84–88, 93, 110, 113, 116, 120, 148, 150, 154, 156, 161, 162, 168, 259–261, 271, 301, 304, 344, 348, 410, 412
chapter (counter), 296
chapter (pagestyle), 77, 84, 109, 110, 152
\chapter*, 77, 86, 259, 260, 348
chapterbib (package), 306
\chapterheadstart, 85, 86
\chaptermark, 110, 119
\chaptername, 78, 116, 255, 344
\chapternamenum, 86
\chapternumberline, 156
\chapternumberlinebox, 156
\chapternumberlinehook, 156
\chapterprecis, 92, 93, 325, 413
\chapterprecishere, 93
\chapterprecistoc, 93, 162, 413
\chapterrefname, 302, 344
chapterstyle, xxxi, 87–92, 389–406
    article, 88
    asu, 372
    asuappendix, 372
    bianchi, 90
    bringhurst, 90, 105
    brotherton, 90
    chappell, 90, 402
    companion, 87, 90, 387
    crosshead, 91, 105
    culver, 91
    dash, 91
    default, 87–90, 260
    demo, 403
    demo2, 91, 403
    demo3, 91, 104, 105, 403
    dowding, 91, 105
    ell, 91
    fred, 87
    ger, 91
    hangnum, 87, 89
    komalike, 91, 105
    lyhne, 91
    madsen, 91
    ntglike, 92, 105
    pedersen, 75, 92, 104, 403
    reparticle, 88
    section, 87–89

southall, 92, 404
tandh, 92, 105
thatcher, 92
veelo, 92, 405, 406
verville, 92
wilsondob, 92, 105
\chapterstyle, 87, 88
\chaptertoc, 413
\chaptitlefont, 86, 103
Characters dropped after \end{...}, 475
Charter, 16, 39
\checkandfixthelayout, 22, 23, 25, 26, 29, 33, 53, 202, 248, 356
\checkarrayindex, 340
\checkifinteger, 341
\checkoddpage, 334
\checkthelayout, 23, 24, 473
Cheltenham, 479
chngcntr (package), xxix, 331, 345
chngpage (package), xxix, 135, 334
cicero, xxxvi
circle, *see also* picture object
disk, 425
open, 425
reference point, 425
restricted diameter, 426
\circle, 425, 426, 470
\circle*, 425, 426
Citation ... on page ..., 468
Citation ... undefined, 469
\cite, 54, 305, 307, 468–470
cite bibliographic item, 305
\citeindexfile, 318
\citeindextrue, 318
Clarendon, 479
class, xxv, xxxi
article, xxv, 5, 71, 88, 148
boek, 92, 105
book, xxv, 71, 87, 88, 103, 105, 116, 148
iso, 325
memoir, xxv, xxxi, xxxiii, 1, 102, 148, 150, 151, 169, 265, 303, 342
report, xxv, 64, 71, 88, 103, 148
scrbook, 91, 105
class option, 45
class options, 1
article, 5
bibliography, 5
fonts, 5
layout truncation, 5
math, 5
printing, 4
stock size, 1
type size, 2
\cleaders, 101
\cleardoublepage, 108, 110, 181, 196, 261, 335
cleared (pagestyle), 109, 110
\clearforchapter, 84, 85
\clearmark, 116
\clearpage, 84, 85, 181, 182, 196, 335, 463
\clearplainmark, 116
\cleartoevenpage, 196, 261, 335
\cleartooddpage, 335
\cleartorecto, 84, 110, 335
\cleartoverso, 84, 110, 335
\cline, 230
clo (file), 474
\closeinputstream, 293, 294
\closeoutputstream, 293
\club, 102
cm, xxxvi
\cmd, 349
\cmdname, 83
\cmdname*, 83
\cmdprint, 349
\cmidrule, 223, 224
\cmidrulekern (length), 223, 224
\cmidrulesep (length), 223
\cmidrulewidth (length), 223
colophon, 136
color (package), 92, 280, 385
\colorchapnum, 404
\colorchaptitle, 404
column, 13
double, 4, 13, 22, 71, 73, 121, 385
multiple, 14, 181
single, 4, 14, 71, 73
Column ... is already defined, 475
\columnsep (length), 22, 24
\columnseprule (length), 22, 24

Command ... invalid ..., 465, 469
Command ... not provided ..., 465
Command \... already defined ..., 351, 465
comment (environment), 284, 285
comment out text, 284
comment (package), 285
\commentsoff, 284
\commentson, 284
companion (chapterstyle), 87, 90, 387
companion (pagestyle), 109, 119–122, 387
Computer Modern, 3, 15, 16
Computer Sans, 15, 16
Computer Typewriter, 15, 16
concordance, 309
Concrete Roman, 15, 16
conditional, 447
configuration file
  MakeIndex, 312
Connes, Frederic, 312
\contcaption, 191
\contentsline, 146, 148, 160
\contentsname, 151, 152, 344
\continuousmarks, 68
\continuousnotenums, 253, 256
control sequence, 436
\contsubbottom, 200
\contsubcaption, 200
\contsubtop, 200
\contsuptop, 200
coordinate, 415
coordinate pair, 415
coordinate system, 415
\copy, 454
copyfitting, 13
\copypagestyle, 112
count, 442
\count, 466
counter, 173, 442
  bottomnumber, 185
  chapter, 296
  dbltopnumber, 185
  footnote, 68, 237
  increment, 298
  lastpage, 328
  lastsheet, 328
  maxsecnumdepth, 77
  page, 107
  pagenote, 253, 254, 256
  poemline, 270
  pseudo, 296, 297
  question, 300
  secnumdepth, 77, 79
  sheetsequence, 328
  tocdepth, 145, 151, 165, 168, 174
  topnumber, 185
  totalnumber, 185
  Xdepth, 174
Counter ... already defined, 475
counter representation, 107
  Alph, 107
  alph, 107
  arabic, 107
  Roman, 107
  roman, 107
Counter too large, 465
\counterwithin, 331
\counterwithin*, 331
\counterwithout, 331
\counterwithout*, 331
Courier, 15, 16, 39
\cplabel, 334
\cr, 455, 457, 458
\createmark, 114, 116, 473
\createplainmark, 114–116, 473
\Cref, 302, 344
crop (package), xxx, 345, 407
cross reference, 301
  automatic, 301
  specified, 301
crosshead (chapterstyle), 91, 105
crosshead (headstyles), 105
crownvopaper (option), 2
\cs, 349
\csname, 441, 455, 458
\cstyle, 387
ctabular (environment), 232
CTAN, xxx, xxxi, 35
CTT, xxxi, 352, 405
culver (chapterstyle), 91
Culver, Christopher, 91
\currenttitle, 304

D

dash (chapterstyle), 91
\dashbox, 419, 420
\date, 57, 64, 66, 67, 475
datetime (package), 330
dbillpaper (option), 2
\dblfloatpagefraction, 185
\dblfloatsep (length), 185
\dbltextfloatsep (length), 185
\dbltopfraction, 185
dbltopnumber (counter), 185
dcolumn (package), xxvii, xxx, 211, 214, 345
dd, xxxvi
\def, 440, 461
default,
  page layout, 7
  printing options, 5
  stock, 2
  type size, 3
default (chapterstyle), 87–90, 260
default (headstyles), 103, 105
\defaultaddspace, 223
\defaultaddspace (length), 223, 224
\defaultlists, 140
\defaultsecnum, 97
delarray (package), xxvii, xxx, 211, 345
\delcode, 459
\deleted, 325, 326
\DeleteShortVerb, 285, 286
\delimeter, 459
Della Robbia, 479
demo (chapterstyle), 403
demo2 (chapterstyle), 91, 403
demo3 (chapterstyle), 91, 104, 105, 403
demyvopaper (option), 2
description (environment), 137, 138, 140, 142
\descriptionlabel, 137
diagram, 415
\diamond, 102
didot point, xxxvi
\dimen, 443–445
dimension, 443
Dimension too large, 454
\ding, 101
\DisemulatePackage, 346
disk, *see* circle
Display math should end with $$, 455
\divide, 443–446
divide by zero, 454
division,
  anonymous, 100
  sectional, *see also* subhead, 79
\do, 450
Do not use \footnote ..., 475
\doccoltocetc, 150, 151
document (environment), 5, 346, 437, 438
\documentclass, 1, 4, 5, 16, 41, 465, 468
double column,
  index, 308, 309
double spacing, 50–52
Double subscript, 455
Double superscript, 455
\doublerulesep (length), 222, 223, 230
\DoubleSpacing, 50, 51, 368
\DoubleSpacing*, 50
dowding (chapterstyle), 91, 105
dowding (headstyles), 105
\downbracefill, 224
Downes, Michael, 53, 333
\dp, 442
draft document, 116
draft (option), 5, 116, 298, 325
draw, 415
\dropchapter, 260
\droptitle (length), 65
dvi (file), 292, 436, 438
Dye, Thomas, 92, 404

E

headstyles, 103
\easypagecheck, 135, 334
ebook (option), 1, 6
sectionname, 79
\edef, 440
seename, 314
leftmargini (length), 266
Egyptian, 479
ell (chapterstyle), 91
Els, Danie, 83, 155
\else, 447, 450, 451
em, xxxvii
\eminnershape, 44

\emph, 43, 44
emphasis, 43
Empty 'thebibliography' environment, 475
empty (pagestyle), 66, 83, 109, 112, 113, 116, 152, 370, 372
Empty preamble: 'l' used, 472
\emptythanks, 67
\EmulatedPackage, 345, 346
\EmulatedPackageWithOptions, 345, 346
en, xxxvii
end
  line, 421
\end, 164, 290, 455, 460, 464
end floats, 294–297
\end occurred inside a group ..., 455
\end occurred when ..., 455
\endcsname, 441, 455, 458
\endgroup, 455, 458
\endinput, 362
\endlinechar, 166
\endlist, 143
\endMakeFramed, 281
endnote
  mark, *see also* reference mark
\endnote, 251
endnotes, 294, 300, 374
endnotes (package), 251
\endwriteverbatim, 297
English Serif, 479
Engravers Litho, 479
Engravers Old English, 479
\enlargethispage, 52, 378, 466
\enlargthispage, 467
ent (file), 251–253, 255
enumerate (environment), 128, 137–140, 142, 301, 476
enumerate (package), xxix, 138, 345
enumitem (package), 139, 140
environment,
  abstract, xxiii, 71–74, 344
  adjustwidth, 90, 135, 136, 140, 142, 284
  adjustwidth*, 66, 135
  alltt, 263, 264
  altverse, 267, 269, 273
  answer, 298, 299
  appendices, 78
  array, 211, 213, 217, 218, 224, 228, 230, 232, 455, 457, 458, 463, 465, 466, 471, 472
  bibitemlist, 305–307
  blockdescription, 137
  boxedverbatim, 287–289, 294, 295
  boxedverbatim*, 287
  center, 73, 129, 136, 143, 144, 207, 387
  comment, 284, 285
  ctabular, 232
  description, 137, 138, 140, 142
  document, 5, 346, 437, 438
  enumerate, 128, 137–140, 142, 301, 476
  epigraph, xxvi
  epigraphs, 257–260
  eqnarray, 467
  equation, 301
  fboxverbatim, 287, 297
  figure, 173, 175, 193, 195, 206, 207, 295, 296, 415, 465, 466
  flexlabelled, 138
  flushleft, 73, 129, 144
  flushright, 73, 129, 144
  framed, 175, 280–282, 284
  framedminipage, 279, 280
  frametitle, 283
  framewithtitle, 282, 283
  hangparas, 128
  itemize, 137, 138, 140, 247, 330, 467
  KeepFromToc, 150
  labelled, 138
  lcode, 290, 291, 388
  leftbar, 281
  list, 72, 137, 140, 142, 143, 266, 388
  lrbox, 279
  MakeFramed, 281
  marginfigure, 175
  margintable, 175
  midsloppypar, 52
  minipage, 52, 178, 195, 199, 204, 205, 237, 279, 360
  multicols, 164
  new, 298

onecolabstract, 73, 74
patverse, 267, 269, 270, 275, 340, 471
patverse*, 269, 270, 471
picture, 415–419, 429, 438, 463, 470
qfame, 284
qframe, 284
qshade, 284
quotation, 71, 130, 135, 137, 140, 284, 467
quote, 93, 130, 135, 137, 140
shaded, 280, 281, 284
sidecaption, 202, 203, 205
sidecontcaption, 203
sidelegend, 203
sidenamedlegend, 203
sloppypar, 52
snugshade, 280, 281
snugshaded, 280
subappendices, 78, 79
symbols, 142
syntax, 387
tabbing, 464, 466, 467
table, 173, 175, 193, 194, 232, 465
tabular, 178, 205, 211, 213, 217, 226–228, 230, 232, 237, 387, 455, 457, 458, 463, 465, 466, 471, 472
tabular*, 211, 226, 227, 230
tabularx, 211, 226–229, 476
thebibliography, 110, 305, 306, 322, 467, 475
theglossary, 319, 322
theindex, 110, 309, 311, 322
tightcenter, 387
titledframe, 283
titlepage, 65
titlingpage, 65, 66, 70, 107, 108, 110
trivlist, 143, 144, 291, 387
verbatim, 144, 264, 285–287, 290, 291, 294, 388, 463, 475
verbatim*, 285–287
verbatimoutput, 293
verse, xxviii, 263–266, 268–270
vminipage, 52
vplace, 345
writefigure, 296, 297
writeverbatim, 293, 297
Environment ... already defined, 475
Environment ... undefined, 465
epigraph, 85, 257–262
\epigraph, 257–260
epigraph (environment), xxvi
epigraph (package), xxix, 257, 345
epigraph (pagestyle), 110, 262
\epigraphfontsize, 259
\epigraphforheader, 260
\epigraphhead, 110, 259, 260
\epigraphpicture, 260
\epigraphposition, 258
\epigraphrule (length), 259
epigraphs (environment), 257–260
\epigraphsourceposition, 258
\epigraphtextposition, 258
\epigraphwidth (length), 258, 260
eqnarray (environment), 467
equation (environment), 301
error, 453
  LaTeX, 464–468
  memoir class, 471–474
  TeX, 454–463
error message,
  response, 453
    continue, 453
    edit, 453
    exit, 453
    help, 453
    insert, 453
    quiet, 453
    run, 453
    scroll, 453
descriptionlabel, 137
eso-pic (package), 409
etex, 341
etex (package), xxx, 385
etoolbox (package), 140, 409
\evensidemargin (length), 24
\everydisplay, 51
\everypar, 133
ex, xxxvii
exception dictionary, 463
executivepaper (option), 2
\ext@type, 206
extend a macro, 332

Extra ..., 455
Extra ... or forgotten ..., 455
Extra alignment tab ..., 455
Extra \else, 455
Extra \endcsname, 455
Extra \fi, 455
Extra \or, 455
Extra \right, 455
\extracolsep, 217
extrafontsizes (option), 3, 49, 473
\extrarowheight (length), 230
\extratabsurround, 231
\extratabsurround (length), 231
textwidth, 250

F

fancybox (package), 281
\fancybreak, 100, 101
\fancybreak*, 100
fancyhdr (package), xxv, xxvi, xxx, 108, 345
\fbox, 279–281, 287
\fboxrule (length), 300
\fboxsep (length), 300
fboxverbatim (environment), 287, 297
\fcardinal, 336
\feetabovefloat, 238
\feetbelowfloat, 238
Fette Fraktur, 479
Fette Gotisch, 479
\fi, 447, 450
\fibseries, 448–450
\fibtogo, 449
figure, 5, 121, 136, 148, 157, 160, 169, 173, 174, 177, 187, 195, 196, 199, 200, 207, 210, 261, 355, 361, 415
  float definition, 173
  reference, 302
  sub-, 169
  subfigure, 199–202
figure (environment), 173, 175, 193, 195, 206, 207, 295, 296, 415, 465, 466
\figurename, 173, 208, 209, 344
\figurerefname, 302, 344
figures,
file, 292–300
  ans, 298
  aux, 292, 301, 437, 438
  bib, 307, 308
  bst, 308
  clo, 474
  close, 293, 294
  dvi, 292, 436, 438
  ent, 251–253, 255
  idx, 308–310, 312, 318
  ind, 308, 309, 312, 317
  ist, 312, 315
  jobname.idx, 310
  jobname.ind, 309
  lof, 146
  log, 292, 345, 438, 453
  lot, 146
  mem10.clo, 474
  open, 293, 294
  pdf, 292
  read, 279, 292–294, 297
    single line, 294
    verbatim, 294
  source2e.tex, 440
  tex, 292
  toc, 93, 145, 146, 162, 251, 292, 438
  write, 279, 292–293, 296, 298
    verbatim, 293
File ended while scanning ..., 456
File not found ..., 465
filll, 446
fill, 446
fil, 446
final (option), 5, 298, 325
\finishwritefigure, 297
\firmlist, 140
\firmlists, 140
\firmlists*, 140
\firsthline, 231
fixltx2e (package), 121, 385
\fixthelayout, 23, 25, 477
flafter (package), 182
\flagverse, 270, 276
Flanders, 479
\flegfigure, 194
\flegfloat, 194
\flegtable, 194
\flegtocfigure, 194

\flegtocfloat, 194
\flegtoctable, 195
fleqn (option), 5
fleuron, xxvii, 101, 480
flexlabelled (environment), 138
float, 6, 75, 173, 187, 189, 191–195, 198, 200, 206, 209, 210, 237, 335, 361, 385
  bottom, 181, 182, 185
  centering, 129
  double column, 173, 181, 185, 187
  flush, 181
  framed, 175
  here, 181, 182, 185
  multiple, 177–181
    table and figure, 178
  new, 173–175
  new diagram, 174
  new subfloat, 180
  page, 121–124, 181, 182
  parameters, 182
  placement, 181–187
  position, 174
  ruled, 176
  set off, 175–177
  single column, 173, 181
  subfloat, 199–202
  suppress, 182
  suppress bottom, 182
  suppress top, 182
  top, 181, 182, 185
Float too large ..., 469
Float(s) lost, 465
\FloatBlock, 182
\FloatBlockAllowAbove, 182
\FloatBlockAllowBelow, 182
floatcomp (pagestyle), 122
\floatpagefraction, 185–187
\floatsep (length), 185, 187
\flushbottom, 6, 23, 24, 77, 335
flushleft, 129
flushleft (environment), 73, 129, 144
\flushleftright, 131
flushright, 129
flushright (environment), 73, 129, 144
\fnsymbol, 243
\fnum@figure, 207, 208
\fnum@type, 206, 207
\fnumbersep, 336
folio, xxxv, 20, 75, 107, 109, 121, 170, 171, 355, 359
  changing representation, 107
font, xxxv, xxxvii
  bitmap, 35, 49
  Computer Modern, 35
  family, xxxv
  Knuthian, 15
  measuring, 13
  Metafont, 15
  Metafont, 35
  monospaced, 15
  Opentype, 35
  outline, 35, 44, 49
  PostScript, 15, 35, 479
  TrueType, 35, 479
Font ... not loadable ..., 456
font characteristic,
  family, 41, 42
  series, 41, 42
  shape, 41, 42
Font command ... is not supported, 472
font commands, 41
font declarations, 41
Font shape ..., 469
font size, 44
fontenc (package), 385
fontfamily, 35
  bch, 39
  pag, 38
  pbk, 39
  pcr, 39
  phv, 39
  pnc, 40
  ppl, 40
  psy, 41
  ptm, 40
  put, 40
  pzc, 41
  pzd, 41
FontSite, 63
fonttable (package), 386
foolscapvopaper (option), 2
footer, 7, 20–21, 108–113, 119–122, 325, 355

\footfootmark, 241
\footfudgefiddle, 239
\footmarksep (length), 240–242
\footmarkstyle, 241
\footmarkwidth (length), 240–242
footmisc (package), 238, 240
footnote, 45, 68–70, 102, 237–244
  bottom float, 238
  fragile, 209
  in caption, 209–210, 454
  in float, 209
  in heading, 102, 454
  mark, *see also* reference mark, 68, 102, 210
  marker, 240
    multiple, 242
    styling, 240
  marker separator, 238
  new series, 239
  reference, 237
  style, 240
  styles, 239
  styling, 240–244
  symbol, 243
    order, 243
  text, 210, 240, 241
    font, 242
  too long, 239
  verbatim text, 238
\footnote, 102, 209, 237–240, 249, 251, 252, 256, 475
footnote (counter), 68, 237
\footnotemark, 68, 209, 237, 240, 256
\footnoterule, 70, 240, 243, 244
footnotes,
  as a paragraph, 239
  as paragraphs, 239
  as three columns, 239
  as two columns, 239
\footnotesatfoot, 249
\footnotesinmargin, 249
\footnotesize, 45, 46, 70, 140, 242, 250, 259
\footnotetext, 209, 210, 237, 249, 256
\footparindent (length), 241
\footref, 237
\footruleheight, 113
\footruleskip, 113
\footskip is too large ..., 472
\footskip (length), 21, 23, 24, 27, 356, 472
\foottextfont, 242, 368
\foottopagenote, 256
Forbidden control sequence found ..., 456
Ford, Matthew, 268
\fordinal, 336
fore-edge, 20
\foremargin (length), 23, 24, 246, 409, 473
\frac, 339
fragile, 164, 304, 454
fragile command, 102
frame,
  box, 280–284
    styling, 281
  minipage, 279
    verbatim, 279
  narrow box, 281
  rounded corners, 281
  title, 282–283
  verbatim, 287, 294
    styling, 288
\frame, 420
\framebox, 418, 419
\FrameCommand, 281
framed,
  box, 418
  float, 175
  verbatim, 297
framed (environment), 175, 280–282, 284
framed (package), xxix, 280, 283, 345
framedminipage (environment), 279, 280
\FrameHeightAdjust, 281
\FrameHeightAdjust (length), 280
\framepichead, 124
\framepichook, 124
\framepictextfoot, 124
\FrameRestore, 281
\FrameRule (length), 280
\FrameSep (length), 280
\FrameTitle, 282
frametitle (environment), 283

framewithtitle (environment), 282, 283
Franklin Gothic, 479
fred (chapterstyle), 87
\freetabcaption, 232
\fref, 302, 344
front matter, 75
frontispiece, 160
\frontmatter, 72, 75, 80, 86, 114, 348
\frontmatter*, 75
Frutiger, Adrian, 39
full stop, *see also* period
fullptlayout (option), 5
Function, 479
\fussy, 52
Futura, 479
\futurelet, 333

G

Garamond, 363, 379, 479
Garcia, Gerardo, 91
gathering, 330
\gdef, 440
geometry (package), xxvi, xxx, 7, 26, 345
ger (chapterstyle), 91
\getarrayelement, 340
\gfibseries, 448, 450
Gill Sans, 479
\global, 440
\glossary, 318–320, 324, 463
\glossarycolsep (length), 322
\glossaryentry, 321
\glossaryintoc, 322
\glossarymark, 322
\glossaryname, 322, 344
\glossaryrule (length), 322
\glossitem, 319, 320, 322, 323
glue, 444
Glypha, 479
Glytus, 479
golden section, 27
Goudy Old Style, 479
\gparindent (length), 291, 292, 388
graphicx (package), 91, 92, 385, 397, 398, 401, 406
group, 440
gutter, 22
    width, 22

H

h (position argument), 181, 186
h float specifier ..., 469
half-title page, 57
\halign, 455, 458
\hangcaption, 188, 189
\hangfrom, 128
hanging,
hanging (package), 128
hangnum (chapterstyle), 87, 89
\hangpara, 128
hangparas (environment), 128
\hangsecnum, 97
\hangsubcaption, 202
chapter, 80
hash size, 463
\hb@xt@, 441
\hbox, 101, 102, 441, 454, 459, 461
\headdrop (length), 21
header, 7, 20–21, 27, 76, 102, 108–113, 119–122, 259, 325, 355, 359
    specifying size, 21, 23
\headheight and/or \headsep are too large ..., 472
\headheight (length), 19, 21, 23, 24, 27, 472
heading, 6, 150, 182, 258
    abstract, 71
    book, 80–83
    chapter, 5, 84–87, 92, 160, 259, 261
    part, 5, 80–83
    sections, 5, 94–100
headings (pagestyle), 7, 109, 115, 116, 118
\headmargin (length), 23, 24, 27
\headnameref, 303
\headsep (length), 19, 21, 24, 27, 359, 472
\headskip, 23
headstyles,
    bringhurst, 104, 105
    crosshead, 105
    default, 103, 105
    dowding, 105
    komalike, 105

memman, 103, 105
ntglike, 105
tandh, 105
wilsondob, 105
\headstyles, 103
\headwidth (length), 112
\heavyrulewidth (length), 222, 223
Helvetica, 15, 16, 39
\hfil, 101, 155, 159, 446
\hfill, 159, 178, 191, 446
\hfilneg, 446
\hhrule, 178
\hideindexmarks, 311
\hline, 230
\hmpunct, 330
horizontal mode, 441
\hrule, 178, 288
\hrulefill, 224
\hsize (length), 281
\hskip, 446
\hss, 446
\ht, 442
\HUGE, 45, 46, 49
\Huge, 45, 46, 82, 86
\huge, 45, 46, 81, 86
Huge page cannot be shipped out, 456
Hurenkinder, *see* widow
hyperindex, 312
hyperref (package), xxv, xxix, 107, 151, 155, 162, 166, 172, 201, 254, 255, 302, 311, 312, 349
hyphenation, 247
\hyphenation, 457, 459, 463
Høgholm, Morten, 14, 15

I

I can't find file ..., 456
I can't go on meeting you like this, 456
I can't write on file ..., 456
sidecapfloatwidth, 204
\idtextinnotes, 254
idx (file), 308–310, 312, 318
\idxmark, 121
if-then-else, 447
\ifbounderror, 340
\ifcase, 450, 451
\ifdim, 447
\ifdraftdoc, 116, 298, 325
\ifetex, 341
\ifinteger, 341
\ifluatex, 342
ifmtarg (package), xxix, 345
\ifnum, 447
\ifodd, 447
\ifoddpage, 334
\ifonlyfloats, 121
\ifpdf, 341
ifpdf (package), 345
\ifsamename, 333
\ifscapmargleft, 203
\IfStreamOpen, 293, 299
iftex (package), xxx, 341, 342
\ifxetex, 342
\ignorenoidxfile, 309
\iiirdstring, 337
\iindstring, 337
Illegal character ..., 465
Illegal parameter number ..., 456
Illegal pream-token ..., 472
Illegal unit of measure ..., 456, 457
illustration, 85, 108, 160, 170, 177, 178, 195, 207
multiple, 177
imperialvopaper (option), 2
Improper \hyphenation ..., 457
in, xxxvi
inch, xxxvi
\include, 311, 346, 347, 465, 466
\includegraphics, 178
\includeonly, 311
Incomplete ..., 457
ind (file), 308, 309, 312, 317
\indentafterchapter, 87
\indentcaption, 188, 189
vindent (length), 266
\indentpattern, 269, 275, 277
index, 121, 160, 170, 262, 265, 386, 387
double column, 308, 309
gutter, 309
entry levels, 311
explanatory text, 309
main entry, 311
multiple, 265, 308, 310

name, 309
preparation, 309–311
print, 309
printing, 308–309
see also reference, 310, 314
see reference, 310, 314
show indexed items, 311
single column, 308, 309
subentry, 311
subsubentry, 311
title in ToC, 309
\index, 310–312, 314, 463
Index ... for pattern ..., 475
Index ... outside limits ..., 472
index (package), xxix, 265, 345, 346
index (pagestyle), 121
\indexcolsep (length), 309
indexing, 121
\indexintoc, 309
\indexmark, 309
\indexmarkstyle, 311
\indexname, 309, 344
\indexrule (length), 309
Infinite glue shrinkage ..., 457
\input, 294, 346, 347, 456, 465
input stack size, 463
Input stream ... is already defined, 475
Input stream ... is not open, 475
Input stream ... is open, 475
\InputIfFileExists, 346
insert in contents, 148
\insertchapterspace, 87, 168, 371
interlinear space, *see* leading
\intextsep (length), 185
iso (class), 325
\isopage, 29, 32
ist (file), 312, 315
\iststring, 337
\item, 137–139, 142, 258, 301, 307, 311, 466, 467
itemize (environment), 137, 138, 140, 247, 330, 467
\itemsep (length), 307
\itshape, 42, 94

J

Jenson Recut, 479
Jessica, 479
Joanna, 479
\jobname, 292
jobname.idx (file), 310
jobname.ind (file), 309
jurabib (package), 308
\justlastraggedleft, 133

K

KeepFromToc (environment), 150
\keepthetitle, 67
Kettler, Howard, 39
keyword,
by, 443
minus, 444
plus, 444
\killtitle, 67
komalike (chapterstyle), 91, 105
komalike (headstyles), 105
kyus, xxxvi

L

l (position argument), 232, 233, 418, 421, 428
\l@book, 154
\l@kind, 146
\l@section, 148
label, 301
\label, 54, 171, 178, 195, 200, 201, 237, 298, 300–304, 442, 463, 469, 470
Label ... multiply defined, 469
Label(s) may have changed ..., 469
labelled (environment), 138
landscape (option), 1, 2
Lanston Bell, 479
Lanston Koch, 479
\LARGE, 45, 46
\Large, 45, 46, 358
\large, 45, 46, 96, 128, 272
largecrownvopaper (option), 2
largepostvopaper (option), 2
\lasthline, 231
\lastlineparrule, 133
\lastlinerulefill, 133

lastpage (counter), 328
lastsheet (counter), 328
LaTeX,
  error, 464–468
  warning, 468–470
latexsym (package), 385, 465
Latin Modern, 3
layouts (package), 29, 32, 153, 386
\lccode, 459
\lcminusname, 338, 344
lcode (environment), 290, 291, 388
leading, xxxv, 50, 342
\leadpagetoclevel, 83
leaf, xxxv
League, Christopher, 63, 479
\leavespergathering, 330
\leavevmode, 102, 442
ledgerpaper (option), 2
ledmac (package), 238, 240
\left, 218, 219, 455
leftbar (environment), 281
\leftcenteright, 133
\leftcenterright, 133
\leftmargini (length), 266
\leftmark, 110, 117, 119, 121
\leftskip, 352
\leftspringright, 134
legalpaper (option), 2
legend, 192
  in list of, 193
  named, 194
\legend, 160, 192, 195, 304, 349, 361
length, 443
  \@dblfpbot, 185, 186
  \@dblfpsep, 185
  \@dblfptop, 185, 186
  \@dblftop, 186
  \@fpbot, 185, 186
  \@fpsep, 185, 186
  \@fptop, 185, 186
  \@pnumwidth, 157
  \abovecaptionskip, 207
  \aboverulesep, 222–224
  \absleftindent, 72
  \absparindent, 72, 73
  \absparsep, 72
  \absrightindent, 72
  \abstitleskip, 73
  \afterchapskip, 85, 86
  \afterepigraphskip, 259
  \afterPoemTitleskip, 272
  \arraycolsep, 230
  \arrayrulewidth, 230
  \baselineskip, 21, 23, 25, 50, 51, 128, 207, 259, 342, 388
  \beforechapskip, 85, 86, 405
  \beforeepigraphskip, 259
  \beforePoemTitleskip, 272
  \belowcaptionskip, 207
  \belowrulesep, 222, 223
  \bibitemsep, 307
  \bottomsectionskip, 77, 78
  \bvboxsep, 287, 288
  \cftbeforeKskip, 154
  \cftKindent, 154
  \cftKnumwidth, 154, 155
  \cftparskip, 153
  \chapindent, 89, 90
  \cmidrulekern, 223, 224
  \cmidrulesep, 223
  \cmidrulewidth, 223
  \columnsep, 22, 24
  \columnseprule, 22, 24
  \dblfloatsep, 185
  \dbltextfloatsep, 185
  \defaultaddspace, 223, 224
  \doublerulesep, 222, 223, 230
  \droptitle, 65
  leftmargini, 266
  \epigraphrule, 259
  \epigraphwidth, 258, 260
  \evensidemargin, 24
  \extrarowheight, 230
  \extratabsurround, 231
  \fboxrule, 300
  \fboxsep, 300
  \floatsep, 185, 187
  \footmarksep, 240–242
  \footmarkwidth, 240–242
  \footparindent, 241
  \footskip, 21, 23, 24, 27, 356, 472
  \foremargin, 23, 24, 246, 409, 473
  \FrameHeightAdjust, 280
  \FrameRule, 280

\FrameSep, 280
\glossarycolsep, 322
\glossaryrule, 322
\gparindent, 291, 292, 388
\headdrop, 21
\headheight, 19, 21, 23, 24, 27, 472
\headmargin, 23, 24, 27
\headsep, 19, 21, 24, 27, 359, 472
\headwidth, 112
\heavyrulewidth, 222, 223
\hsize, 281
vindent, 266
\indexcolsep, 309
\indexrule, 309
\intextsep, 185
\itemsep, 307
\leftmargini, 266
\lightrulewidth, 222, 223
\linewidth, 148, 204
\lowermargin, 472, 474
\lxvchars, 15, 16
\marginparpush, 21, 23, 24
\marginparsep, 21, 23, 24, 202, 245, 359, 360
\marginparwidth, 21–24, 202, 245, 359, 360
\midchapskip, 85, 86, 405
\midPoemTitleskip, 272
\normalrulethickness, 113
\oddsidemargin, 24
\onelineskip, 50, 342, 388
\paperheight, 12, 22, 24, 473, 474
\paperwidth, 12, 22, 24, 136, 356, 473
\parindent, 16, 50, 127, 130, 132, 213
\parsep, 307
\parskip, 49, 50, 127, 153, 207
\partopsep, 143
\pfbreakskip, 101
\prechapterprecisshift, 93
\pwlayii, 359, 360
\pwlayi, 359, 360
\ragrparindent, 130
rigid, 443, 445
rubber, 443–445
\sidebarhsep, 247, 248
\sidebartopsep, 247, 248
\sidebarvsep, 247, 248
\sidebarwidth, 247, 248
\sidecapraise, 204
\sidecapsep, 202, 204
\sidecapwidth, 202
\sidefootadjust, 249
\sidefoothsep, 249
\sidefootmarksep, 250
\sidefootmarkwidth, 250
\sidefootparindent, 250
\sidefootvsep, 249
\sidefootwidth, 249
\sideparvshift, 245
\spinemargin, 23, 24, 136, 473
\stanzaskip, 266
\stockheight, 22, 24, 473
\stockwidth, 22, 24, 356, 473
\tabcolsep, 230
\textfloatsep, 185, 187
\textheight, 17, 23–25, 248, 474
\textwidth, 17–19, 23, 24, 113, 136, 231, 473
\thanksmarksep, 69
\thanksmarkwidth, 68, 69
\topmargin, 19, 24
\topsep, 143
\topskip, 23, 25, 53
\trimedge, 22, 24, 27, 356, 473
\trimtop, 22, 24, 27, 473
\unitlength, 137, 167, 260, 359, 415, 416
\uppermargin, 19, 23, 24, 472, 474
\verbatimindent, 287
\versewidth, 265, 266
\vgap, 266, 267, 269
\vindent, 266
\vleftmargin, 266
\vleftskip, 270
\vrightskip, 270
\width, 281
\xlvchars, 15, 16
Length ... already defined, 475
leqno (option), 5
\let, 163, 440
\letcountercounter, 331
letterpaper, 8

letterpaper (option), 2, 5, 27, 29
\lightrulewidth (length), 222, 223
Limit controls ..., 457
\limits, 457
Limits for array ... , 472
Linden, 479
line, , *see also* picture object
  arrowhead, *see* vector
  end, 421
  length, 423
  orphan, 52–54
  reference point, 423
  restricted slope, 423, 424
  short, 423
  slope, 423
  thickness, 416
  widow, 52–54
\line, 423, 425, 426, 464, 470
line number, 266, 267, 276, 287, 295
  font, 270, 289
  frequency, 270, 289
  position, 270, 289
  reset, 289
line too long, *see* overfull lines
\linebreak, 461
\linenottooshort, 132
\linenumberfont, 270, 289
\linenumberfrequency, 270, 289
\linethickness, 416
\linewidth (length), 148, 204
lining,
Linotype, 39
list, 137–144
  new, 140–143
  new list of, 167–171
  spaces, 291
  tight, 140
\list, 143
list (environment), 72, 137, 140, 142, 143, 266, 388
\listanswername, 167
\listfigurename, 151, 152, 344
\listofanswers, 167
\listoffigugres, 344
\listoffigures, 110, 145, 150, 153
\listoffigures*, 145, 150
\listofplates, 170
\listoftables, 110, 145, 148, 150, 153, 344
\listoftables*, 145, 150
\listplatename, 170
\listtablename, 151, 152, 344
Litho Antique, 479
\llap, 89
lmodern (package), 3
\LoadClass, 347, 465, 467, 468
LoF, 145, 146, 149–153, 155, 160, 168, 170, 355, 361, 371, 463
lof (file), 146
\lofmark, 151
log (file), 292, 345, 438, 453
Lonely \item ..., 466
\long, 440
longtable (package), xxvii, 232
\loop, 450
\loosesubcaptions, 201
LoT, 145, 146, 149–153, 168, 170, 355, 361, 371
lot (file), 146
\lotmark, 151
lowercase,
\lowermargin, 23
\lowermargin (length), 472, 474
LR
  mode, 417
lrbox (environment), 279
alsoname, 314
LTR, 352
Lubalin, Herb, 38
Lucas, Édouard, 448
Lucida Bright, 363, 379
Luecking, Daniel, 157
\lxvchars (length), 15, 16
Lydian, 479
lyhne (chapterstyle), 91
Lyhne, Anders, 91

M

\m@ne, 443
\m@th, 102
madsen (chapterstyle), 91
Madsen, Lars, 83, 90, 91, 161, 404
main matter, 75, 305

main memory size, 463
\mainmatter, 75, 77, 78, 80, 114, 117, 374
\mainmatter*, 75, 80
\makeatletter, 149, 191, 207, 243, 362, 368, 439, 461, 462
\makeatother, 149, 191, 207, 243, 362, 368, 439, 462
\makebox, 418, 419, 422
makebst (package), 308
\makechapterstyle, 88, 89
\makeevenfoot, 112
\makeevenhead, 112
\makefootrule, 112, 113
\MakeFramed, 281
MakeFramed (environment), 281
\makeglossary, 318, 319, 322
\makeheadfootruleprefix, 112, 113
\makeheadfootstrut, 113
\makeheadposition, 113, 333
\makeheadrule, 112, 113
\makeheadstyles, 103
makeidx (package), xxix, 345
MakeIndex,
\makeindex, 308, 309
MakeIndex (program), xxv, xxix, 312, 314, 315, 317–321, 324, 387
  configuration file, 312
  output styling, 315–317
  raw data, 312–315
\MakeLowercase, 54, 358
\makeoddfoot, 112
\makeoddhead, 112
\makeother, 461
\makepagenote, 251
\makepagestyle, 112, 113
\makepsmarks, 113, 115, 119, 122
\makerunningwidth, 112
\MakeShortVerb, 285, 286
\MakeTextLowercase, 54
\MakeTextUppercase, 54, 109, 115, 162
\makethanksmark, 69
\makethanksmarkhook, 69
\maketitle, 57, 63–67, 70, 74, 110, 239, 466, 469, 475
\maketitlehooka, 65
\maketitlehookb, 65
\maketitlehookc, 65
\maketitlehookd, 65
\MakeUppercase, 54, 55, 117, 155
\marg, 349
margin, 5, 7, 17, 18, 23, 72, 135–137, 149, 160, 326, 359
  foredge
    fore-edge, 356
  fore-edge, *see also* outer, 8, 17, 18, 27, 109, 119, 355
  fore-edge, 17
  inner, *see also* spine, 17, 27, 119
  left, 87, 94, 96, 97, 127, 129, 135, 146, 148, 154, 266
  lower, 17, 19, 20, 27
  outer, *see also* fore-edge, 17, 20, 27, 119, 121, 136
  right, 129, 135, 136, 146, 149, 153, 159, 259
  specify width, 17
  specifying size, 17–20, 23
  spine, *see also* inner, 8, 17, 18, 23, 27, 136
  top, *see also* upper, 356
  upper, *see also* top, 17, 19–21, 23, 27, 355
margin note, 244
  moved, 245
  wrong margin, 244
marginalia, 7, 21–23, 27, 119, 193, 355, 356, 359
\marginbox, 99
marginfigure (environment), 175
\marginhead, 99
\marginpar, 175, 237, 244, 245, 247–249, 465–467, 469, 475
Marginpar on page ..., 469, 475
\marginparmargin, 22, 244
\marginparpush (length), 21, 23, 24
\marginparsep, 249
\marginparsep (length), 21, 23, 24, 202, 245, 359, 360
\marginparwidth, 249
\marginparwidth (length), 21–24, 202, 245, 359, 360
margintable (environment), 175
\markboth, 109, 110
markers, 110, 119

\markright, 109, 110
math,
in caption or title, 454
math mode, 441
math unit, 146
\mathindent, 5
mathpazo (package), 16
\max, 332
\maxdeadcycles, 459
maximum length, 454
maximum number, 454
\maxsecnumdepth, 79, 80
maxsecnumdepth (counter), 77
\maxtocdepth, 151
mcrownvopaper (option), 1
mdemyvopaper (option), 1
\mdseries, 42
measure, xxxvi
45 characters, 15
65 characters, 15
\medievalpage, 29, 32
mediumvopaper (option), 2
\medspace, 342
Melior, 479
mem10.clo (file), 474
\memappchapinfo, 348
\memappchapstarinfo, 348
\memapppageinfo, 348
\memapppagestarinfo, 348
\membicaptioninfo, 349
\membionenumcaptioninfo, 349
\membitwonumcaptioninfo, 349
\membookinfo, 348
\membookstarinfo, 348
\memcaptioninfo, 349
\memchapinfo, 348
\memchapstarinfo, 348
\memdskips, 51
\memdskipstretch, 51
\memendofchapterhook, 84, 85
\memfontenc, 3
\memfontfamily, 3
\memfontpack, 3
\memglodesc, 321
\memglonum, 321
\memgloref, 321
\memgloterm, 321
\memgobble, 345
memhfixc (package), xxix
\memjustarg, 345
\memleadpageinfo, 348
\memleadpagestarinfo, 348
\memlegendinfo, 349
memman (headstyles), 103, 105
\memnamedlegendinfo, 349
memoir (class), xxv, xxxi, xxxiii, 1, 102, 148, 150, 151, 169, 265, 303, 342
error, 471–474
warning, 474–476
\memorigdbs, 131
\memorigpar, 131
\mempartinfo, 348
\mempartstarinfo, 348
\mempoeminfo, 349
\mempoemstarinfo, 349
\memPoemTitleinfo, 349
\memPoemTitlestarinfo, 349
\mempostaddapppagetotochook, 162
\mempostaddbooktotochook, 162
\mempostaddchaptertotochook, 162
\mempostaddparttotochook, 162
\mempreaddapppagetotochook, 162
\mempreaddbooktotochook, 162
\mempreaddchaptertotochook, 162
\mempreaddparttotochook, 162
\MemRestoreOrigMakecase, 54
\memRTLleftskip, 352
\memRTLraggedleft, 352
\memRTLraggedright, 352
\memRTLrightskip, 352
\memRTLvleftskip, 352
\memRTLvrightskip, 352
\memsecinfo, 348
\memsecstarinfo, 348
memsty (package), 164, 320, 386
\memUChead, 114, 115
\mergepagefloatstyle, 122
\meta, 349
Metafont, 35
\midbicaption, 198, 199
\midbookskip, 81
\midchapskip (length), 85, 86, 405
midpage (package), 196
\midpartskip, 81

\midPoemTitleskip (length), 272
\midrule, 222
\midsloppy, 52
midsloppypar (environment), 52
Miedinger, Max, 39
Miller & Richard, 38
millimetre, xxxvi
Minion, 355
minipage,
in float, 178
minipage (environment), 52, 178, 195, 199, 204, 205, 237, 279, 360
\miniscule, 45, 46, 49
minus (keyword), 444
\minusname, 338, 344
minus, 444, 446
Misplaced &, 457
Misplaced \cr, 457
Misplaced \noalign, 457
Misplaced \omit, 457
Misplaced \span, 457
Missing = inserted ..., 458
Missing @-exp ..., 466
Missing # inserted ..., 458
Missing $ inserted, 458
Missing \cr inserted, 458
Missing \endcsname inserted, 458
Missing \endgroup inserted, 458
Missing \right inserted, 458
Missing { inserted, 458
Missing }inserted, 458
Missing \begin{document}, 466
Missing arg: token ignored, 472
Missing control sequence inserted, 459
Missing delimeter(. inserted)., 459
Missing number ..., 459
Missing p-arg ..., 466
mixing fonts, 43
\mkern, 102
mlargecrownvopaper (option), 1
mm, xxxvi
mode, 441
display math, 441
horizontal, 441
internal vertical, 441
LR, 417
math, 441
paragraph, 422
picture, 417, 418, 421, 422
restricted horizontal, 441
vertical, 441
Modern, 479
monospaced font, 15
\morecmidrules, 223
moreverb (package), xxix, 345
Morison, Stanley, 40
\movetoevenpage, 335
\movetooddpage, 335
moving argument, 304, 454
\mpjustification, 175
ms (option), 5
msmallroyalvopaper (option), 1
\multfootsep, 238
multicol (package), 163, 181, 280
multicols (environment), 164
\multicolumn, 217, 229, 465, 466, 473
multind (package), 265
\multiply, 443–446
\multiput, 417, 428, 431, 463
myheadings (pagestyle), 109, 110, 118

N

\n@me@number, 339
name,
index, 309
named,
number, 336, 337
\namedlegend, 194, 195
\namedsubappendices, 78, 79
\namenumberand, 338, 339, 344
\namenumbercomma, 338, 344
nameref (package), 302
\namerefoff, 304
\namerefon, 304
\nametest, 333, 334
natbib (package), 306, 308, 318
\Needspace, 335
\needspace, 265, 276, 335
needspace (package), xxix, 345
\Needspace*, 335
new,
counter, 296
environment, 296

New Baskerville, 479
New Century Schoolbook, 3, 16, 40, 63
\newarray, 340, 472
\newblock, 307
\newcolumntype, 214, 216, 217, 228, 475, 476
\newcommand, 216, 332, 435, 440, 456
\newcommand*, 333, 460
\newcomment, 284
\newcount, 442
\newcounter, 168, 331, 442, 466
\newdimen, 443, 444
\newenvironment, 456, 463
newfile (package), xxix, 345
\newfixedcaption, 195
\newfloat, 173, 174, 181
\newfootnoteseries, 239, 240
\newfootseries, 239
\newif, 447
\newinputstream, 292, 475
\newleadpage, 83
\newlength, 444, 445
\newline, 77, 461, 467
\newlistentry, 168–170
\newlistof, 77, 167, 168
\newloglike, 332
\newloglike*, 332
\newoutputstream, 292, 476
\newpage, 335
News Gothic, 479
\newsavebox, 279, 422, 431
\newskip, 443–445
\newsubfloat, 180, 199
\newtheorem, 463, 466
nextpage (package), xxix, 335, 345
\nextpageref, 171
Nicholas Cochin, 479
No array called ..., 472
No \author given, 469
No counter ... defined, 466
No more to read from stream ..., 475
No positions in optional float specifier ..., 469
No room for a new ..., 466
No room for a new read, 292
No room for a new write, 292
No \title given, 466
\noalign, 457
\nobibintoc, 305
\nobookblankpage, 82
\nobvbox, 287–289
\NoCaseChange, 55
\nochangemarks, 325
\nocite, 307
\noDisplayskipStretch, 51
\nofiles, 437, 438
\noglossaryintoc, 322
\noindent, 94, 258
\noindentafterchapter, 87
\noindexintoc, 309
\nolimits, 457
\nonzeroparskip, 49
\nopartblankpage, 82
\noprelistbreak, 330
\normalbottomsection, 77, 78
\normalcaption, 188, 189
\normalcaptionwidth, 189
\normalfont, 16, 41, 245
\normalmarginpar, 244
\normalrulethickness, 122
\normalrulethickness (length), 113
\normalsize, 45, 46, 128, 245
\normalsubcaption, 202
Not a letter, 459
Not defined: ..., 472
Not in outer par mode, 466
Not redefinable: ..., 473
\notedivision, 253, 256, 344
\noteidinnotes, 254
\notenuminnotes, 254
\notenumintext, 254
\notepageref, 253, 254
\notesname, 253, 344
\nouppercaseheads, 109, 114, 115
ntglike (chapterstyle), 92, 105
ntglike (headstyles), 105
\nthstring, 337
number, 336
  named, 336, 337
  numeric, 336
  representation, 336
Number too big, 459
numbered lines, 287
\numberline, 146, 156, 160, 163

\numberlinebox, 156
\numberlinehook, 156
\NumberPoemTitle, 270, 271
numeric,
    number, 336
\NumToName, 337
\numtoName, 337
\numtoname, 337, 338
Nyman, Patrick, 128

O

\oarg, 349
\oddsidemargin (length), 24
old-style,
Olde English, 479
oldfontcommands (option), 5, 472
oldpaper (option), 2
colorchapnum, 404
colorchaptitle, 404
\omit, 457
onecolabstract (environment), 73, 74
\onecolglossary, 322
\onecolindex, 309
\onecoltocetc, 150, 151
onecolumn (option), 4, 5, 151, 173
\OnehalfSpacing, 50, 368
\OnehalfSpacing*, 50
\onelineskip, 50, 210, 249
\onelineskip (length), 50, 342, 388
oneside (option), 4, 6, 26, 84
Only one column-spec. allowed, 473
footnotesinmargin, 249
\openany, 84
openany (option), 4, 66, 84
openbib (option), 5, 307
\openinputfile, 293, 294
\openleft, 84
openleft (option), 4, 84
\openoutputfile, 293
\openright, 84
openright (option), 4, 5, 84
Opentype, 35
Optima, 479
option, , xxxi
    *pt, 3, 4, 474
    10pt, 2, 3, 5, 46
    11pt, 2, 46
    12pt, 2, 6, 46
    14pt, 3, 46, 113
    17pt, 3, 46, 473
    20pt, 3, 46
    25pt, 3, 46, 473
    30pt, 3, 46
    36pt, 3, 46
    48pt, 3, 46
    60pt, 3, 46
    9pt, 2, 46
    a3paper, 1
    a4paper, 1, 27, 29
    a5paper, 1
    a6paper, 1
    a7paper, 1
    article, 5, 77, 84, 88, 93
    b3paper, 1
    b4paper, 1
    b5paper, 1
    b6paper, 1
    b7paper, 1
    broadsheetpaper, 2
    crownvopaper, 2
    dbillpaper, 2
    demyvopaper, 2
    draft, 5, 116, 298, 325
    ebook, 1, 6
    executivepaper, 2
    extrafontsizes, 3, 49, 473
    final, 5, 298, 325
    fleqn, 5
    foolscapvopaper, 2
    fullptlayout, 5
    imperialvopaper, 2
    landscape, 1, 2
    largecrownvopaper, 2
    largepostvopaper, 2
    ledgerpaper, 2
    legalpaper, 2
    leqno, 5
    letterpaper, 2, 5, 27, 29
    mcrownvopaper, 1
    mdemyvopaper, 1
    mediumvopaper, 2
    mlargecrownvopaper, 1
    ms, 5

msmallroyalvopaper, 1
oldfontcommands, 5, 472
oldpaper, 2
onecolumn, 4, 5, 151, 173
oneside, 4, 6, 26, 84
openany, 4, 66, 84
openbib, 5, 307
openleft, 4, 84
openright, 4, 5, 84
postvopaper, 2
pottvopaper, 2
royalvopaper, 2
sectionbib, 306
showtrims, xxvi, 5, 326
smalldemyvopaper, 2
smallroyalvopaper, 2
statementpaper, 2
superroyalvopaper, 2
titlepage, 65, 71
twocolumn, 4, 6, 71–73, 151, 173
twoside, 4–6, 26, 34, 84, 246
Option clash for ..., 466
Optional argument is not one of: ..., 473
Optional argument of \twocolumn ..., 470, 475
Opus, 479
\or, 450
ordinal,
\ordinal, 336
\OrdinalToName, 337
\ordinaltoName, 337
\ordinaltoname, 337
\ordscript, 336, 337
orphan
line, 52–54
postbibhook, 306
postnotetext, 255
outline,
font, 35, 44, 49
\output, 459
Output loop ..., 459
Output stream ... is already defined, 476
Output stream ... is not open, 476
Output stream ... is open, 476
oval, *see also* picture object
reference point, 427
\oval, 427, 428, 470
\oval, \circle, or \line size unavailable, 470
Overfull \hbox ..., 459
Overfull \vbox ..., 459
overfull lines, 52
overfull, 441
\overridescapmargin, 204
\overridesidecapmargin, 472

P

p (position argument), 174, 181, 186
package, , xxxi, 73, 296, 361, 362
abstract, xxix, 71, 345
afterpage, 171, 196
alltt, 263, 385
amsfonts, 465
amsmath, 385
answer, 300
appendix, xxix, 345
array, xxvii, xxx, 211, 212, 345
babel, xxv
bidi, xxx, 352, 353
bookmark, 172
booktabs, xxvii, xxix, 211, 222, 345
bringhurst, 362
calc, 25, 27, 248, 409, 410
caption, 187, 210
ccaption, xxvii, xxix, 187, 345
changepage, 334
chapterbib, 306
chngcntr, xxix, 331, 345
chngpage, xxix, 135, 334
color, 92, 280, 385
comment, 285
crop, xxx, 345, 407
datetime, 330
dcolumn, xxvii, xxx, 211, 214, 345
delarray, xxvii, xxx, 211, 345
endnotes, 251
enumerate, xxix, 138, 345
enumitem, 139, 140
epigraph, xxix, 257, 345
eso-pic, 409
etex, xxx, 385
etoolbox, 140, 409
fancybox, 281

fancyhdr, xxv, xxvi, xxx, 108, 345
fixltx2e, 121, 385
flafter, 182
fontenc, 385
fonttable, 386
footmisc, 238, 240
framed, xxix, 280, 283, 345
geometry, xxvi, xxx, 7, 26, 345
graphicx, 91, 92, 385, 397, 398, 401, 406
hanging, 128
hyperref, xxv, xxix, 107, 151, 155, 162, 166, 172, 201, 254, 255, 302, 311, 312, 349
ifmtarg, xxix, 345
ifpdf, 345
iftex, xxx, 341, 342
index, xxix, 265, 345, 346
jurabib, 308
latexsym, 385, 465
layouts, 29, 32, 153, 386
ledmac, 238, 240
lmodern, 3
longtable, xxvii, 232
makebst, 308
makeidx, xxix, 345
mathpazo, 16
memhfixc, xxix
memsty, 164, 320, 386
midpage, 196
moreverb, xxix, 345
multicol, 163, 181, 280
multind, 265
nameref, 302
natbib, 306, 308, 318
needspace, xxix, 345
newfile, xxix, 345
nextpage, xxix, 335, 345
pagenote, 345
parskip, xxix, 345
patchcmd, xxix, 333, 345
pifont, 41, 101, 385
placeins, 182
ragged2e, 175
sectsty, 75
setspace, xxix, xxx, 50, 345
shortvrb, xxix, 286, 345
showidx, xxix, 345
sidecap, xxx, 189, 345
siunitx, 214
subfig, xxvii
subfigure, xxx, 199, 345
tabularx, xxvii, xxx, 211, 345
tcolorbox, 280, 289
textcase, xxx, 54, 109, 115, 482
titleref, xxix, 302, 304, 345
titlesec, xxx, 75, 345
titling, xxix, 63
tocbibind, xxix, 145, 345
tocloft, xxv, xxix, 145, 345
tocvsec2, 80, 151
url, 368, 385
verbatim, xxix, 286, 345
verse, xxix, 345
wrapfig, 205
xcolor, 280, 385
xtab, xxvii, 232
packages before class, 16
pag (fontfamily), 38
page, xxxv, 13
  break, 231, 232
  current number, 301
  height, 12, 17, 19, 20
  location, 22
  of floats, 121
  reference, 302
  size, 8, 12, 22
  specifying size, 12, 20
  width, 12, 17, 18
page break,
  asynchronous, 437
page color, 6
page (counter), 107
Page height already too large, 466
page layout, 7, 22
  change, 370
  check parameters, 23
  class parameters, 8, 22, 26
  default, 7
  design, 8
  LaTeX parameters, 8, 22, 25, 26
  parameters, 23
\pageaiii, 1
\pageaiv, 1

\pageav, 1
\pageavi, 1
\pageavii, 1
\pagebiii, 1
\pagebiv, 1
\pagebroadsheet, 2
\pagebv, 1
\pagebvi, 1
\pagebvii, 1
\pagecrownvo, 2
\pagedbill, 2
\pagedemyvo, 2
\pageexecutive, 2
\pagefoolscapvo, 2
\pageimperialvo, 2
\pageinnotes, 254, 255
\pagelargecrownvo, 2
\pagelargepostvo, 2
\pageledger, 2
\pagelegal, 2
\pageletter, 2
\pagemdemyvo, 1
\pagemediumvo, 2
\pagemetriccrownvo, 1
\pagemlargecrownvo, 1
\pagemsmallroyalvo, 1
\pagename, 344
\pagenote, 251, 252, 254–256
pagenote (counter), 253, 254, 256
pagenote (package), 345
\pagenotesubhead, 255, 256
\pagenotesubheadstarred, 255, 256
\pagenumbering, 107
\pagenumbering*, 107
\pageold, 2
\pagepostvo, 2
\pagepottvo, 2
\pageref, 171, 301, 469, 470
\pagerefname, 255, 302, 344
\pageroyalvo, 2
\pagesmalldemyvo, 2
\pagesmallroyalvo, 2
\pagestatement, 2
pagestyle, xxxi, 7
  asu, 370
  book, 80, 109, 110
  chapter, 77, 84, 109, 110, 152
  class, 108
  cleared, 109, 110
  combining, 122
  companion, 109, 119–122, 387
  empty, 66, 83, 109, 112, 113, 116, 152, 370, 372
  epigraph, 110, 262
  float pages, 121–124
  floatcomp, 122
  headings, 7, 109, 115, 116, 118
  index, 121
  index pages, 121
  myheadings, 109, 110, 118
  part, 80, 109, 110
  plain, 64, 66, 109, 111, 116, 121, 260, 370, 371
  Ruled, 109
  ruled, 7, 109, 120
  section, 87
  showlocs, 124, 326
  simple, 109
  specifying, 108–120
  title, 64, 109, 110
  titlingpage, 65, 66, 109, 110
\pagestyle, 87, 108, 113, 152, 168
\pagesuperroyalvo, 2
\pagetofootnote, 256
pagination, xxxv, 107
  changing, 107
  interrupt, 108
Palatino, 15, 16, 35, 40, 479
paper, xxxv, 1, 8, 29, 170
  end, 160
  size, 1, 2
    A3, 1
    A4, 1, 27, 136
    A5, 1
    A6, 1
    A7, 1
    B3, 1
    B4, 1
    B5, 1
    B6, 1
    B7, 1
    broadsheet, 2
    crown octavo, 2
    demy octavo, 2

dollar bill, 2
executive, 2
foolscap octavo, 2
imperial octavo, 2
large crown octavo, 2
large post octavo, 2
ledger, 2
legal, 2
letterpaper, 2, 27, 136, 356
medium octavo, 2
metric crown octavo, 1
metric demy octavo, 1
metric large crown octavo, 1
metric small royal octavo, 1
old, 2
post octavo, 2
pott octavo, 2
royal octavo, 2
small demy octavo, 2
small royal octavo, 2
statement, 2
super royal octavo, 2
stock, 8
\paperheight and/or \trimtop are too large ..., 473
\paperheight (length), 12, 22, 24, 473, 474
\paperwidth, 18, 19
\paperwidth and/or \trimedge are too large ..., 473
\paperwidth (length), 12, 22, 24, 136, 356, 473
\par, 166, 167, 454, 460
paragraph, 6, 127–130, 136, 159, 263, 355, 358
block, 97, 127, 188, 359
reasons against, 127
hanging, 97, 128–129, 136, 359
in list, 128
indentation, 72, 100, 127, 130, 189, 258
mode, 422
outdentation, 127
typeset as a unit, 128, 130
\paragraph, 76, 79, 98, 99, 154, 343
paragraph break,
invisible, 246
Paragraph ended before ..., 460
paragraph indentation, 50
\paragraphfootnotes, 239
\paragraphfootstyle, 239
\paraheadstyle, 103
parallel texts, 232
\parbox, 99, 212, 228
\parfillskip, 159
\parg, 349
\parindent, 72
\parindent (length), 16, 50, 127, 130, 132, 213
\parnopar, 246
\parsep (length), 307
\parshape, 136, 265
\parskip, 72
parskip (package), xxix, 345
\parskip (length), 49, 50, 127, 153, 207
part, 76, 146, 148, 149
number, 77
\part, 5, 76, 77, 79–84, 110, 154, 156, 157, 161, 162, 261, 344, 348, 412
part (pagestyle), 80, 109, 110
\part*, 348
\partblankpage, 82
\partmark, 82, 84
\partname, 81, 344
\partnamefont, 81, 103
\partnamenum, 81
\partnumberline, 156
\partnumberlinebox, 156
\partnumberlinehook, 156
\partnumfont, 81, 103
\partopsep (length), 143
\partpageend, 82, 83
\partrefname, 302, 344
\parttitlefont, 82, 103
de Parville, Henri, 448
\patchcmd, 472, 473
patchcmd (package), xxix, 333, 345
\patchcommand, 333
patverse (environment), 267, 269, 270, 275, 340, 471
patverse* (environment), 269, 270, 471
pbk (fontfamily), 39
pc, xxxvi
pcr (fontfamily), 39

\Pcstyle, 387
pdf (file), 292
pdfLaTeX, 341
pedersen (chapterstyle), 75, 92, 104, 403
Pedersen, Troels, 92
period, *see also* full stop
\pfbreak, 101
\pfbreak*, 101
\pfbreakdisplay, 101
\pfbreakskip (length), 101
\phantomsection, 151
Phemister, Alexander, 39
phv (fontfamily), 39
pica, xxxvi
picture, 415
  box, 416
  mode, 417, 418, 421, 422
  zero-sized, 417
picture (environment), 415–419, 429, 438, 463, 470
picture object, 417
  Bezier curve, *see also* Bezier curve
  box, 418
  circle, *see also* circle
  dashed box, 419
  frame, 420
  framed box, 418
    text position, 418
  line, *see also* line
  oval, *see also* box, rounded
  picture, 429
  regular pattern, 428
  stack, 421
  text, 417
  two-dimensional pattern, 431
  unframed box, 419
    text position, 419
    zero size, 419
  vector, *see also* vector
pifont (package), 41, 101, 385
pilcrow (¶), 138
placeins (package), 182
plain (pagestyle), 64, 66, 109, 111, 116, 121, 260, 370, 371
\plainbreak, 100
\plainbreak*, 100
\plainfancybreak, 100, 101
\plainfancybreak*, 100
\plainfootnotes, 239
\plainfootstyle, 239
\PlainPoemTitle, 270, 271, 273
plates, 170
Please type a command ..., 460
plus (keyword), 444
plus, 444, 446
\pmname, 330
\pmnane, 330, 344
pnc (fontfamily), 40
\pnchap, 255
\pnschap, 255
poem,
  author, 274
poem title, 271, 273
  in ToC, 271, 273
  styling, 275
poemline (counter), 270
\PoemTitle, 270, 271, 348
\poemtitle, 273, 302, 348
\PoemTitle*, 271
\PoemTitlefont, 272
\PoemTitleheadstart, 271, 272
\poemtitlemark, 271
\PoemTitlenumfont, 272
\poemtitlepstyle, 271
\poemtitlestarmark, 271
\poemtitlestarpstyle, 271
\poemtoc, 271, 273
point, xxxv, xxxvi
pool size, 463
\poptabs, 466
\postauthor, 64
\postbibhook, 306
\postcaption, 189, 198
\postchapterprecis, 93
\postdate, 64
\postnoteinnotes, 254
\postnotetext, 254, 255
PostScript, 35, 479
\posttitle, 64
postvopaper (option), 2
pottvopaper (option), 2
ppl (fontfamily), 40
\Ppstyle, 386, 387

preamble, 5, 6, 27, 70, 80, 121, 149, 153, 157, 189, 243, 296, 308, 361, 362, 386, 464, 468
\preauthor, 64
\prebibhook, 262, 306
\precaption, 189, 198
\prechapterprecis, 93
\prechapterprecisshift (length), 93
\precisfont, 93
\precistocfont, 93, 94, 162
\precistocformat, 93, 94, 162
\precistoctext, 93, 94, 162
\predate, 64
\Pref, 302, 344
\pref, 302, 344
\preglossaryhook, 322
\preindexhook, 262, 309
\prenoteinnotes, 254
\prenotetext, 254, 255
\pretitle, 64
\printanswer, 298
\printanswers, 298
\printbookname, 81
\printbooknum, 81
\printbooktitle, 82
\printchaptername, 86, 89
\printchapternonum, 86, 89
\printchapternum, 86
\printchaptertitle, 86, 152, 358
\printfibterm, 449, 450
\printfigures, 297
\printglossary, 319, 324
\printindex, 308, 309
\printpageinnotes, 254, 255
\printpageinnoteshyperref, 254, 255
\printpagenotes, 251–253
\printpagenotes*, 251–253, 256
\printpartname, 81
\printpartnum, 81
\printparttitle, 82
\printPoemTitlenonum, 271, 272
\printPoemTitlenum, 271, 272
\printPoemTitletitle, 271, 272
\printtime, 330
\printtime*, 330
\printXtitle, 152
\priorpageref, 171
program,
  BibTeX, xxv, 307, 308
  MakeIndex, xxv, xxix, 312, 314, 315, 317–321, 324, 387
  xindy, 312
proportion
  book, 12
  margin, 17
  typeblock, 13
protect, 102, 299
\protect, 55, 102, 148, 149, 164, 209, 255, 293, 295, 304, 454
\providecommand, 332, 440, 456
\providecommand*, 440
\providecounter, 332, 475
\provideenvironment, 332, 475
\providefixedcaption, 195
\providelength, 332, 475
\provideloglike, 332
\provideloglike*, 332
pseudo (counter), 296, 297
psnfss, 15, 35
\pstyle, 386, 387
psy (fontfamily), 41
pt, xxxvi
ptm (fontfamily), 40
\published, 65
\pushtabs, 466
\pushtabs and \poptabs don't match, 466
\put, 417, 423, 425, 429
put (fontfamily), 40
\pwlayii (length), 359, 360
\pwlayi (length), 359, 360
pzc (fontfamily), 41
pzd (fontfamily), 41

Q

\qbezier, 417, 431, 433, 463
\qbeziermax, 433
qfame (environment), 284
qframe (environment), 284
\qitem, 257, 258
qshade (environment), 284
quad, xxxvii

\quarkmarks, 328, 330
question (counter), 300
questions and answers, 297–300
\quietref, 299
quotation, 127, 130, 257, 259, 260
quotation (environment), 71, 130, 135, 137, 140, 284, 467
quote (environment), 93, 130, 135, 137, 140

R

r (position argument), 232, 233, 419, 421, 428
ragged2e (package), 175
\raggedbottom, 6, 52, 78, 238, 335
\raggedbottomsection, 77, 78
raggedleft, 129
\raggedleft, 129, 130, 188, 202, 205, 228, 352
raggedright, 129, 247
\raggedright, 129, 130, 188, 202, 205, 228, 352
\raggedrightthenleft, 133
\raggedyright, 130
\ragrparindent (length), 130
ranging,
\readaline, 294
\readboxedverbatim, 294
\readboxedverbatim*, 294
\readstream, 294
\readverbatim, 294
\readverbatim*, 294
recto, xxxv
Redefining primitive column ..., 476
\ref, 54, 55, 201, 298–301, 469, 470
reference,
  by name, 302–304
  by number, 301–302
  current value, 301
  set current value, 301, 302
  to appendix, 302
  to book, 302
  to chapter, 302
  to counter, 311
  to figure, 302
  to label, 301
  to page, 301, 302, 311
  to part, 302
  to section, 302
  to table, 302
  unexpected result, 301, 303
Reference ... on page ..., 470
reference mark, *see also* endnote mark, footnote mark, 243
reference point, 417
  box, 418
  circle, 425
  line, 423
  oval, 427
  stack, 421
  text, 417
  vector, 425
\refixpagelayout, 26
\refstepcounter, 298, 301, 302, 442
registered trademark, 343
\registrationColour, 328
\relax, 443
\renewcommand, 89, 113, 146, 155–157, 194, 195, 207, 293, 331, 333, 334, 433, 440, 456
\renewcommand*, 204, 333, 440
\renewenvironment, 456
\renewfixedcaption, 195
\renewleadpage, 83
reparticle (chapterstyle), 88
\repeat, 450
report (class), xxv, 64, 71, 88, 103, 148
\reportnoidxfile, 309
representation,
  number, 336
\RequireAtEndClass, 347
\RequireAtEndPackage, 347
\RequirePackage, 16, 467
\RequirePackage or \LoadClass in Options Section, 467
\RequireXeTeX, 342
\resetbvlinenumber, 289
\resizebox, 406
\restorepagenumber, 107, 108
\restoretrivseps, 143
\reversemarginpar, 244
\right, 218, 219, 455, 458
\rightmark, 110, 117, 119, 121

\rightskip, 352
\rmfamily, 42
roman numerals, 107
Roman, 107
roman, 107
Rosslaire, 479
royalvopaper (option), 2
RTL, 352
rule,
in margin, 281
invisible, 122
thickness, 22, 113, 122
ruled,
float, 176
Ruled (pagestyle), 109
ruled (pagestyle), 7, 109, 120
Runaway argument, 460
Runaway definition, 460
Runaway preamble, 460
Runaway text, 460
running BibTeX, 308
\russianpar, 132

S

Sabon, 479
Sanderson, Donald, 63
save stack size, 463
\savebox, 279, 422, 431
\savepagenumber, 107, 108
\savetrivseps, 143
Savoy, 479
\saythanks, 73, 74
\sbox, 279, 422, 423, 431
\scapmargleftfalse, 203
\scapmarglefttrue, 203
Schnittger, 479
Schusterjungen, *see* orphan
scrbook (class), 91, 105
Script
type, 41
\scriptsize, 45, 46
\scshape, 42
\secheadstyle, 103
secnumdepth (counter), 77, 79
section, *see also* subhead
\section, 5, 76–79, 88, 96, 97, 99, 116, 120, 148, 154, 163, 247, 301, 303, 304, 348, 359, 410
section (chapterstyle), 87–89
section (pagestyle), 87
\section*, 76
sectional division, *see also* subhead
level number, 79
sectionbib (option), 306
\sectionmark, 110, 116, 119
\sectionrefname, 302, 344
sectsty (package), 75
\see, 314, 315, 344
\seealso, 314, 315, 344
\seename, 315, 344
\semiisopage, 29, 32
Serifa, 479
Serific, 479
\setaftersecskip, 358
\setafterSskip, 96
\setarrayelement, 340
\setbeforesecskip), 358
\setbeforeSskip, 94
\setbiblabel, 307
\setbinding, 19, 23
\setbvlinenums, 289
\setcolsepandrule, 22
\setcounter, 107, 145, 295, 331, 442, 463, 466, 473
\setDisplayskipStretch, 51
\setfloatadjustment, 174, 175
\setFloatBlockFor, 182
\setfloatlocations, 181
\setFloatSpacing, 51
\setfootins, 22, 244
\sethangfrom, 97, 98, 359
\setheaderspaces, 21, 23
\setheadfoot, 21, 23
\setlength, 8, 22, 72, 86, 113, 142, 154, 155, 204, 207, 258, 259, 266, 378, 410, 415, 445
\setlist, 140
\setlrmargins, 17, 19, 23
\setlrmarginsandblock, 16, 18, 23
\setlxvchars, 15
\setmarginfloatcaptionadjustment, 175

\setmarginnotes, 21–23, 202, 244, 245, 248, 477
\setmpjustification, 175
\setpagebl, 33
\setpagebm, 33
\setpagebr, 33
\setpagecc, 13, 33
\setpageml, 33
\setpagemm, 33
\setpagemr, 33
\setPagenoteSpacing, 51
\setpagetl, 33
\setpagetm, 33
\setpagetr, 33
\setpageXX, 33
\setparahook, 99
\setpnumwidth, 153, 157
\setrmarg, 153
\setsecheadstyle, 358
\setsecnumdepth, 79, 80, 151, 473
\setsecnumformat, 79, 97–99
\setSheadstyle, 96
\setShook, 97, 98
\setsidcaps, 202
\setsidebarheight, 247, 248
\setsidebars, 248
\setsidecappos, 203, 472
\setsidecaps, 202
\setsidefeet, 250
\setsidefootheight, 249
\setSindent, 96
\setSingleSpace, 51
setspace (package), xxix, xxx, 50, 345
\setstocksize, 8, 12, 22
\setsubsecheadstyle, 359
\setsubsechook, 97
\settocdepth, 145, 151, 164, 361, 414, 473, 476
\settocpreprocessor, 162
\settodepth, 446
\settoheight, 446
\settowidth, 446
\settrimmedsize, 12, 13, 17, 22, 32
\settrims, 13, 22, 32
\settypeblocksize, 16, 17, 23, 333
\settypeoutlayoutunit, 26
\setulmargins, 17, 19, 21, 23
\setulmarginsandblock, 16, 19, 20, 23
\setupmaintoc, 166
\setupparasubsecs, 165–167
\setupshorttoc, 165
\setverbatimfont, 286, 368
\setverselinenums, 270
\setxlvchars, 15
\sffamily, 42, 209
\sfseries, 355
shaded (environment), 280, 281, 284
sheet, 8
sheetsequence (counter), 328
\shipout, 326
\Shook, 97, 98
\shortstack, 421
\shortsubcaption, 202
shortvrb (package), xxix, 286, 345
\showcols, 217
\showheadfootlocoff, 124
showidx (package), xxix, 345
\showindexmarks, 311
showlocs (pagestyle), 124, 326
\showtextblockoff, 124
showtrims (option), xxvi, 5, 326
\showtrimsoff, 326
\showtrimson, 326
side footnote
  text, 250
side note, 245
  adjust position, 245
  text for particular margin, 246
sidebar, 246, 404
  styling, 247
\sidebar, 247, 248
\sidebarfont, 247, 248
\sidebarform, 247
\sidebarhsep (length), 247, 248
\sidebarmargin, 247
\sidebartopsep (length), 247, 248
\sidebarvsep (length), 247, 248
\sidebarwidth (length), 247, 248
sidecap (package), xxx, 189, 345
\sidecapfloatwidth, 204
\sidecapmargin, 203, 204, 474
\sidecapraise (length), 204
\sidecapsep (length), 202, 204
\sidecapstyle, 204

\sidecaption, 474
sidecaption (environment), 202, 203, 205
\sidecapwidth (length), 202
sidecontcaption (environment), 203
\sidefootadjust, 249
\sidefootadjust (length), 249
\sidefootenote, 249
\sidefootfont, 249, 250
\sidefootform, 251
\sidefoothsep, 249
\sidefoothsep (length), 249
\sidefootmargin, 249
\sidefootmarksep (length), 250
\sidefootmarkstyle, 250
\sidefootmarkwidth (length), 250
\sidefootnote, 249
\sidefootnotemark, 249
\sidefootnotetext, 249
\sidefootparindent (length), 250
\sidefoottextfont, 250
\sidefootvsep, 249
\sidefootvsep (length), 249
\sidefootwidth, 249
\sidefootwidth (length), 249
sidelegend (environment), 203
sidenamedlegend (environment), 203
\sidepar, 245, 246, 248
\sideparfont, 245
\sideparform, 245
\sideparmargin, 245–247
\sideparvshift (length), 245
signature, xxxv
simple (pagestyle), 109
single column,
  index, 308, 309
\SingleSpacing, 51
\SingleSpacing*, 51
siunitx (package), 214
\sixt@@n, 443
size,
  page, 8, 12, 22
  paper, 1, 2
  stock, 1, 8, 12, 22
  typeblock, 23
Size substitutions ...., 470
\skip, 443–445, 466
\slashfrac, 339
\slashfracstyle, 339
Slimbach, Robert, 40
slope, 423, 424
  line, 423
  vector, 425
\sloppy, 52, 461
\sloppybottom, 53
sloppypar (environment), 52
\slshape, 42
\small, 45, 46, 128, 140, 174, 229, 259, 290, 355, 388
smalldemyvopaper (option), 2
smallroyalvopaper (option), 2
snugshade (environment), 280, 281
snugshaded (environment), 280
Some shapes ..., 470
Something's wrong ..., 467
Sorry, but I'm not ..., 460
sort, xxxv
source2e.tex (file), 440
\sourceatright, 130
southall (chapterstyle), 92, 404
space,
  at start of paragraph, *see* paragraph indentation
  between lines, *see* leading
  double, *see* (ouble spacing)i
  gobble, 299
  inter-paragraph, 49–50
  interword, 52
\spacefactor, 462
\span, 455, 457
\specialindex, 311
\specialrule, 223
\spinemargin and/or \textwidth and/or \foremargin are too large ..., 473
\spinemargin (length), 23, 24, 136, 473
spread, 12, 17
\Sref, 302, 344
stack
  reference point, 421
stacking
  text, 421
    vertical spacing, 422
standard class, xxv

stanza,
end, 263
indent alternate lines, 269, 273
indent pattern, 269, 277
last line, 266
numbering, 266
line break, 267, 275
indent, 267
long line, 274
number, 270, 276
prevent page break, 266
space, 266
\stanzaskip (length), 266
start new page, 196
statementpaper (option), 2
\stepcounter, 442
stock, xxxv, 5, 7, 8, 12, 13, 21, 22, 27, 356
commercial printing, 8
default, 2
height, 8
paper, 8
size, 1, 8, 12, 22
A3, 1
A4, 1, 8
A5, 1
A6, 1
A7, 1
B3, 1
B4, 1
B5, 1
B6, 1
B7, 1
broadsheet, 2
crown octavo, 2
demy octavo, 2
dollar bill, 2
ebook, 1
executive, 2
foolscap octavo, 2
imperial octavo, 2
large crown octavo, 2
large post octavo, 2
ledger, 2
legal, 2
letterpaper, 2, 8
medium octavo, 2
metric crown octavo, 1
metric demy octavo, 1
metric large crown octavo, 1
metric small royal octavo, 1
old, 2
post octavo, 2
pott octavo, 2
royal octavo, 2
small demy octavo, 2
small royal octavo, 2
statement, 2
super royal octavo, 2
specifying size, 8, 22
specifying trimming, 13
trimmed, 8, 12
width, 8
stock paper size option, 8, 12
\stockaiii, 1
\stockaiv, 1
\stockav, 1
\stockavi, 1
\stockavii, 1
\stockbiii, 1
\stockbiv, 1
\stockbroadsheet, 2
\stockbv, 1
\stockbvi, 1
\stockbvii, 1
\stockcrownvo, 2
\stockdbill, 2
\stockdemyvo, 2
\stockexecutive, 2
\stockfoolscapvo, 2
\stockheight (length), 22, 24, 473
\stockimperialvo, 2
\stocklargecrownvo, 2
\stocklargepostvo, 2
\stockledger, 2
\stocklegal, 2
\stockletter, 2
\stockmdemyvo, 1
\stockmediumvo, 2
\stockmetriccrownvo, 1
\stockmlargecrownvo, 1
\stockmsmallroyalvo, 1
\stockold, 2
\stockpostvo, 2
\stockpottvo, 2

\stockroyalvo, 2
\stocksmalldemyvo, 2
\stocksmallroyalvo, 2
\stockstatement, 2
\stocksuperroyalvo, 2
\stockwidth (length), 22, 24, 356, 473
\stop, 460
stream, 292–300
  check open, 293
  close input, 294
  close output, 293
  input, 292, 294
  limited number, 292
  new input, 292
  new output, 292
  output, 292, 293, 297, 298
Stream ... is not open, 476
\strictpagecheck, 135, 245, 334
\strictpagecheck, 204
\stringtoarray, 340
\strip@pt, 359
Struktor, 479
strut, 422
\strut, 113
subappendices (environment), 78, 79
\subbottom, 180, 199, 200
\subcaption, 180, 199
\subcaptionfont, 201
\subcaptionlabelfont, 201
\subcaptionref, 200, 201
\subcaptionsize, 201
\subcaptionstyle, 202
\subconcluded, 200
subfig (package), xxvii
subfigure (package), xxx, 199, 345
subhead, 75
  moved, 77
  near bottom of page, 77
\subitem, 311
\subparagaph, 343
\subparagraph, 76, 79, 96, 154
\subparaheadstyle, 103
\subsecheadstyle, 103
\subsection, 76, 79, 96, 97, 99, 116, 154, 165, 348, 410
\subsubitem, 311
\subsubsecheadstyle, 103
\subsubsection, 76, 79, 96, 154
\subtop, 180, 199, 200
Suggested extra height ..., 467
superroyalvopaper (option), 2
superscript, 343
\suppressfloats, 182
Symbol, 41, 385
symbol, 343
symbols (environment), 142
\symbolthanksmark, 67, 68
Syntax, 355, 479
syntax (environment), 387

T

t (position argument), 174, 181, 186, 419, 428
Tab overflow, 467
tabbing (environment), 464, 466, 467
\tabcolsep (length), 230
table, 121, 148, 169, 173, 177, 187, 191, 193, 195, 199, 355, 361
  column
    cut-in head, 220
    decked head, 220
    head, 220
    spanner head, 220
    spanner rule, 220
    stub, 220
  half and half, 222
  long, 232
  reference, 302
  row fill, 224–226
  rule, 221–224
    horizontal, 220
    vertical, 220
  subtable, 199–202
table (environment), 173, 175, 193, 194, 232, 465
\tablename, 173, 194, 344
\tableofcontents, 77, 110, 145, 146, 148, 150, 153, 167, 344, 438
\tableofcontents*, 145, 150, 413
\tablerefname, 302, 344
\tabsoff, 286, 287
\tabson, 286
tabular, 226–229, 455

automatic, 233–235
by column, 234
by row, 233
column style, 233
column width, 233, 234
position, 233
table width, 233, 234
column width, 226, 227
continuous, 232
position, 232
controlling width, 226
free, 231–235
caption, 232
intercolumn space, 226, 227, 230
new column type, 228
raggedright, 228
row spacing, 230
rule, 227
vertical position, 230
X column, 226, 229
tabular (environment), 178, 205, 211, 213, 217, 226–228, 230, 232, 237, 387, 455, 457, 458, 463, 465, 466, 471, 472
tabular* (environment), 211, 226, 227, 230
tabularx (environment), 211, 226–229, 476
tabularx (package), xxvii, xxx, 211, 345
\tabularxcolumn, 228
Tahoma, 363, 379
tandh (chapterstyle), 92, 105
tandh (headstyles), 105
tcolorbox (package), 280, 289
\tensunitsep, 338
TeX,
error, 454–463
warning, 454–462
TeX capacity exceeded ..., 460, 462–463
tex (file), 292
text,
Text line contains ..., 460
Text page ... contains only floats, 470
\textasteriskcentered, 138
\textbf, 42
\textbullet, 138
textcase (package), xxx, 54, 109, 115, 482
\textendash, 138
\textfloatsep (length), 185, 187
\textfraction, 185, 186
\textheight (length), 17, 23–25, 248, 474
\textit, 42
\textmd, 42
\textperiodcentered, 138
\textregistered, 343
\textrm, 42
\textsc, 42
\textsf, 42
\textsl, 42
\textsubscript, 339, 340
\textsuperscipt, 343
\textsuperscript, 339
\texttrademark, 343
\texttt, 42
\textup, 42
textwidth, 15
\textwidth, 5
\textwidth (length), 17–19, 23, 24, 113, 136, 231, 473
thanks, 57, 67
styling, 67–70
\thanks, 57, 65, 67, 68, 70, 74, 239, 240, 475
\thanksfootmark, 69
\thanksheadextra, 68
\thanksmark, 68
\thanksmarksep (length), 69
\thanksmarkseries, 67, 68
\thanksmarkstyle, 68
\thanksmarkwidth (length), 68, 69
\thanksrule, 70
\thanksscript, 69
That makes 100 errors ..., 461
thatcher (chapterstyle), 92
Thatcher, Scott, 92
\the, 443, 444, 446
The ... font command is deprecated ..., 476
The 'extrafontsizes' option ..., 473
The combination of argument values ..., 473
The counter will not be printed ..., 476
The file needs format ..., 467

\theauthor, 66, 67
\thebibliography, 344
thebibliography (environment), 110, 305, 306, 322, 467, 475
\thechapter, 295
\thectr, 331
\thedate, 66, 67
\theglossary, 344
theglossary (environment), 319, 322
\theindex, 344
theindex (environment), 110, 309, 311, 322
\thepage, 107, 328
\thepoem, 270
\thepoemline, 270
There were multiply defined labels, 470
There were undefined references, 470
There's no line to end here, 467
\thesection, 99
\thesheetsequence, 328
\thetitle, 66, 67
\theTitleReference, 304
\thicklines, 416
thickness
  line, 416
\thinlines, 416
\thinspace, 342
This can't happen ..., 461
This may be a LaTeX bug, 467
\thispagestyle, 108
\thr@@, 443
\threecolumnfootnotes, 239
\threecolumnfootstyle, 239
tightcenter (environment), 387
\tightlist, 140
\tightlists, 140
\tightsubcaptions, 201
Times, 363
Times New Roman, 40
Times Roman, 15, 16, 40, 379
\tiny, 45, 46
tip in, 108, 170
title, 57
  styling, 63–67
\title, 57, 64, 66, 67, 466, 475
title page, 57, 63, 65–66, 345
title (pagestyle), 64, 109, 110
titledframe (environment), 283
\TitleFrame, 283
titlepage (environment), 65
titlepage (option), 65, 71
\titleref, 302–304
titleref (package), xxix, 302, 304, 345
titlesec (package), xxx, 75, 345
titling,
titling (package), xxix, 63
titlingpage (environment), 65, 66, 70, 107, 108, 110
titlingpage (pagestyle), 65, 66, 109, 110
\titlingpageend, 65, 66
\tmarkbl, 327
\tmarkbm, 327
\tmarkbr, 327
\tmarkml, 327
\tmarkmr, 327
\tmarktl, 327
\tmarktm, 327
\tmarktr, 327
ToC, 72, 76–79, 83, 92–94, 102, 119, 145, 146, 148–157, 159–161, 163–168, 170, 271, 273, 279, 301, 303, 305, 309, 335, 355, 358, 361, 370, 371, 373, 374, 379, 409, 411–414, 454, 463
  controlling entries, 174
toc (file), 93, 145, 146, 162, 251, 292, 438
tocbibind (package), xxix, 145, 345
tocdepth (counter), 145, 151, 165, 168, 174
tocloft (package), xxv, xxix, 145, 345
\tocmark, 151
\tocnameref, 303
tocvsec2 (package), 80, 151
token, 436
Too deeply nested, 467
Too many }'s, 461
Too many columns ..., 467
Too many unprocessed floats, 467
\topfraction, 185–187
\topmargin (length), 19, 24
topnumber (counter), 185
\toprule, 222
\topsep (length), 143
\topskip, 86, 359

\topskip (length), 23, 25, 53
totalnumber (counter), 185
to, 441
\tracingtabularx, 227, 228
trademark, 343
\traditionalparskip, 49, 50
Transitional
  type, 40
\tref, 302, 344
trim,
  specifying, 22
\trimedge (length), 22, 24, 27, 356, 473
\trimFrame, 326
\trimLmarks, 326
\trimmark, 327
\trimmarks, 327
\trimmarkscolor, 326
\trimNone, 326, 327
\trimtop (length), 22, 24, 27, 473
\trimXmarks, 326, 327
trivlist (environment), 143, 144, 291, 387
TrueType, 35, 479
\ttfamily, 42
TUG, xxxi
Turing complete language, 435
Turing, Alan, 435
\tw@, 443
Two \documentclass commands, 468
Two \LoadClass commands, 468
\twocolglossary, 322
\twocolindex, 309
\twocoltocetc, 150, 151
\twocolumn, 74, 470, 475
twocolumn (option), 4, 6, 71–73, 151, 173
\twocolumnfootnotes, 239, 244
\twocolumnfootstyle, 239
twoside (option), 4–6, 26, 34, 84, 246
\TX@verb, 229
type, *see also* typeface
  Blackletter, 479
  Decorative, 480
  Didone, 479
  Free Form, 479
  Geralde, 479
  Mediaeval, 479
  Modern, 479
  Sans Serif, 479
  Script, 41, 480
  Slab Serif, 479
  Symbol, 480
  Transitional, 40, 479
  Uncial, 479
  Venetian, 479
type size, 2
  default, 3
typeblock, 6–8, 17–19, 21, 22, 27, 29, 66, 113, 119, 136, 137, 257, 258, 266, 355, 356, 358–360
  height, 16, 17, 19, 20
  location, 17, 20, 23
  size, 23
  specifying size, 17, 20, 23
  width, 17, 18
typeface, xxxv, 15
  Alte Schwabacher, *see* Alte Schwabacher
  ATF Clearface, *see* ATF Clearface
  Avant Garde Gothic, *see* Avant Garde Gothic
  Barbedour, *see* Barbedour
  Basel, *see* Basel
  Basilia, *see* Basilia
  Baskerville, *see* Baskerville
  Bembo, *see* Bembo
  Bergamo, *see* Bergamo
  Bernhard Modern, *see* Bernhard Modern
  Bodoni, *see* Bodoni
  Bookman, *see* Bookman
  Caslon, *see* Caslon
  Centaur, *see* Centaur
  Century Old Style, *see* Century Old Style
  Chantilly, *see* Chantilly
  Charter, *see* Charter
  Cheltenham, *see* Cheltenham
  Clarendon, *see* Clarendon
  Computer Modern, *see* Computer Modern
  Computer Sans, *see* Computer Sans
  Computer Typewriter, *see* Computer Typewriter

Concrete Roman, *see* Concrete Roman
Courier, *see* Courier
Della Robbia, *see* Della Robbia
Egyptian, *see* Egyptian
English Serif, *see* English Serif
Engravers Litho, *see* Engravers Litho
Engravers Old English, *see* Engravers Old English
Fette Fraktur, *see* Fette Fraktur
Fette Gotisch, *see* Fette Gotisch
Flanders, *see* Flanders
Franklin Gothic, *see* Franklin Gothic
Function, *see* Function
Futura, *see* Futura
Garamond, *see* Garamond
Gill Sans, *see* Gill Sans
Glypha, *see* Glypha
Glytus, *see* Glytus
Goudy Old Style, *see* Goudy Old Style
Helvetica, *see* Helvetica
Jenson Recut, *see* Jenson Recut
Jessica, *see* Jessica
Joanna, *see* Joanna
Lanston Bell, *see* Lanston Bell
Lanston Koch, *see* Lanston Koch
Latin Modern, *see* Latin Modern
Linden, *see* Linden
Litho Antique, *see* Litho Antique
Lydian, *see* Lydian
Melior, *see* Melior
Minion, *see* Minion
Modern, *see* Modern
New Baskerville, *see* New Baskerville
New Century Schoolbook, *see* New Century Schoolbook
News Gothic, *see* News Gothic
Nicholas Cochin, *see* Nicholas Cochin
Olde English, *see* Olde English
Optima, *see* Optima
Opus, *see* Opus
Palatino, *see* Palatino
Rosslaire, *see* Rosslaire
Sabon, *see* Sabon
Savoy, *see* Savoy
Schnittger, *see* Schnittger
Serifa, *see* Serifa
Serific, *see* Serific
Struktor, *see* Struktor
Symbol, *see* Symbol
Syntax, *see* Syntax
Times New Roman, *see* Times New Roman
Times Roman, *see* Times Roman
Typewriter, *see* Typewriter
Unitus, *see* Unitus
Univers, *see* Univers
University Old Style, *see* University Old Style
URW Antiqua, *see* URW Antiqua
URW Latino, *see* URW Latino
URW Palladio, *see* URW Palladio
Utopia, *see* Utopia
Vendome, *see* Vendome
Walbaum, *see* Walbaum
Zapf Chancery, *see* Zapf Chancery
Zapf Dingbats, *see* Zapf Dingbats
`\typeout`, 15
`\typeoutlayout`, 26
`\typeoutstandardlayout`, 26
Typewriter, 15, 229
typewriter, 381
typist, 381

U

`\ucminusname`, 338, 344
`numberline`, 410
Unbalanced output routine, 461
Unbalanced write command, 461
`\unboldmath`, 469
Undefined control sequence, 461
Undefined index file ..., 476
Undefined tab position, 468
Underfull `\hbox` ..., 461
Underfull `\vbox` ..., 461
`underfull`, 441
`\undodrop`, 260
Unicode, 35
unit length, 415
`\unitlength`, 416, 417

\unitlength (length), 137, 167, 260, 359, 415, 416
Unitus, 479
Univers, 479
University Old Style, 479
Unknown document division ..., 473
Unknown mark setting type ..., 473
Unknown numbering type ..., 473
Unknown option ..., 468
Unknown toclevel for ..., 476
\unletcounter, 331
\unnamedsubappendices, 78, 79
Unrecognized argument for \sidecapmargin, 474
Unused global option(s) ..., 470
\upbracefill, 224
\uppercaseheads, 109, 114, 115
\uppermargin and/or \textheight and/or \lowermargin are too large ..., 474
\uppermargin (length), 19, 23, 24, 472, 474
\upshape, 42
url (package), 368, 385
URW Antiqua, 479
URW Latino, 479
URW Palladio, 479
Use of ... doesn't match ..., 461
\usebox, 279, 422
\usepackage, 345, 347, 464, 465, 468
\usepackage before \documentclass, 468
\usethanksrule, 70
Utopia, 16, 40

V

\valign, 455, 458
\vbox, 441, 454, 459, 461
vector, *see also* picture object
  reference point, 425
  restricted slope, 425
  slope, 425
\vector, 425, 426, 464
veelo (chapterstyle), 92, 405, 406
Veelo, Bastiaan, 92, 246, 408
Vendome, 479
\verb, 227, 229, 285, 311
\verb, 468
\verb ended by end of line, 468
\verb illegal in command argument, 468
\verb may be unreliable ..., 476
\verb*, 229, 285, 286
verbatim, 279, 285–292
  changing font, 286
  font, 290
  frame, 287, 294
    styling, 288
  in argument, 279
  new, 289–292
  short, 285
  with tab spaces, 286
  wrap long lines, 287
  write, 294, 297
\verbatim, 286, 291, 292
verbatim (environment), 144, 264, 285–287, 290, 291, 294, 388, 463, 475
verbatim (package), xxix, 286, 345
verbatim* (environment), 285–287
\verbatim@font, 291
\verbatim@line, 291
\verbatim@processline, 292
\verbatim@startline, 291, 292
\verbatimbreakchar, 287
\verbatimindent (length), 287
\verbatiminput, 294
\verbatiminput*, 294
verbatimoutput (environment), 293
\verbfootnote, 238
Verdana, 363, 379
verse, 263–278
  centering, 265, 266
  end a line, 263
  end a stanza, 263
  indent line, 264, 265
  indent pattern, 265, 275
  indent space, 266, 267
  long line, 265, 266
  long lines, 264
  multiple indexes, 265
  prevent page break, 265
  typesetting environments, 263–265
  wrapped line indent, 266

verse (environment), xxviii, 263–266, 268–270
verse (package), xxix, 345
\verselinebreak, 267, 275
\verselinenumbersleft, 270
\verselinenumbersright, 270
\versewidth (length), 265, 266
verso, xxxv
vertical,
  centering, 345
vertical mode, 441
verville (chapterstyle), 92
Verville, Guy, 92
\vfil, 446
\vfill, 446
\vfilneg, 446
\vgap (length), 266, 267, 269
\vin, 266
\vindent (length), 266
\vinphantom, 267
\vleftmargin (length), 266
\vleftofline, 267
\vleftskip, 352
\vleftskip (length), 270
\vline, 230
vminipage (environment), 52
vplace (environment), 345
\vrightskip, 352
\vrightskip (length), 270
\vskip, 51, 446
\vspace*, 86
\vss, 446

W

Walbaum, 479
warning,
  LaTeX, 468–470
  memoir class, 474–476
  TeX, 454–462
\wd, 442
widow
  line, 52–54
\width (length), 281
wilsondob (chapterstyle), 92, 105
wilsondob (headstyles), 105
wrapfig (package), 205
\wrappingoff, 287
\wrappingon, 287
write,
  verbatim, 299
\write, 461
writefigure (environment), 296, 297
writeverbatim (environment), 293, 297

X

X columns too narrow (table too wide), 229
X columns too narrow ..., 476
xcolor (package), 280, 385
\xdef, 440
Xdepth (counter), 174
XeTeX, 35, 342, 474, 479
XeTeX is required to process this document, 474
\Xheadstart, 152
xindy (program), 312
\xindyindex, 312
\xlvchars (length), 15, 16
\Xmark, 152
xtab (package), xxvii, 232

Y

You can't use ... in ..., 461
You can't use '\spacefactor' in vertical mode, 462
You can't use 'macro parameter character #' in ... mode, 462
You have requested release ..., 470
You have requested version ..., 470
You have used the '*pt' option but file ..., 474

Z

\z@, 388, 443, 446
Zapf Chancery, 41
Zapf Dingbats, 41, 385
Zapf, Hermann, 40, 41
zero-sized picture, 124
\zerotrivseps, 143

Index of first lines

Beautiful Railway Bridge of the Silv'ry Tay, 311

Fury said to, 278

His judgement rendered, he dissolved the Thing, 268

I am François, which is unfortunate, 232
I used to love my garden, 263, 273
In a cavern, in a canyon, 276
In mathematics he was greater, 275

Je suis Françoys, dont il me pois, 232

Prince, do not ask in a week, 43
Prince, n'enquerez de sepmaine, 43

Then God created Newton, 274
There was a young lady of Ryde, 276
There was a young man of Quebec, 273
There was an old party of Lyme, 264

What a funny thing is a flea, 274

Colophon

This manual was typeset using the LaTeX typesetting system created by Leslie Lamport and the memoir class. The body text is set 10/12pt on a 33pc measure with Palatino designed by Hermann Zapf, which includes italics and small caps. Other fonts include Sans, Slanted and Typewriter from Donald Knuth's Computer Modern family.